Plan to Learn:
Active Robot Learning by Planning

Shivam Vats

August, 2024
CMU-RI-TR-24-40

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA

Thesis Committee:

Maxim Likhachev	Carnegie Mellon University *(Co-chair)*
Oliver Kroemer	Carnegie Mellon University *(Co-chair)*
Reid Simmons	Carnegie Mellon University
George Konidaris	Brown University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Robotics.

Abstract

Robots hold the promise of becoming an integral part of human life by helping us in our homes, out on farms and in our factories. However, current robots lack the motor skills necessary to perform everyday manipulation tasks, operate outside structured settings and interact with humans. This thesis advocates the principles of active, continual and collaborative learning to allow a robot to autonomously learn the skills necessary to master its domain. We propose a novel *Plan to Learn* (P2L) framework in which the robot solves a meta planning problem to decide which skills it should learn so that it can achieve its long-term objective while minimizing the cost of data collection. We formalize and study this idea from both a practical and a theoretical lens in two challenging scenarios.

First, we explore how robots can plan to learn as part of a collaborative human-robot team. We develop an optimal mixed integer programming-based planner Act, Delegate, or Learn (ADL) to allocate tasks and decide which skills the robot should learn to reduce its teammate's workload. We also provide $\log(n)$-approximation algorithms for ADL by showing that it is an instance of the well-known uncapacitated facility location problem. Next, we explore multi-step tasks, such as opening a door, which require several skills to be sequenced. Our first algorithm MetaReasoning for Skill Learning (MetaReSkill) estimates a probabilistic model of skill improvement to identify and prioritize skills that are both easy to learn and most relevant to the overall task. Finally, we present a hierarchical reinforcement learning formulation to solve the P2L problem for recovery learning. RecoveryChaining learns both where and how to recover by leveraging a hybrid action space consisting of primitive robot actions and nominal options that transfer control to a model-based controller. We demonstrate the effectiveness of our P2L framework on a variety of practically motivated and challenging manipulation tasks both in simulation and in the real world.

This thesis is only a first step towards the ambitious goal of building autonomously learning robots that plan to learn. We sincerely hope that the developed framework and its instantiations on these manipulation tasks will pave the way for further research.

Acknowledgements

First and foremost, I want to express my heartfelt gratitude to my advisors Max and Oliver. I consider myself incredibly lucky to be their student. Max introduced me to research. Max, your commitment to solving fundamental robotics problems with principled algorithms is an inspiration to me. Thank you for your constant support, guidance and patience over the years. You have taught me valuable lessons in research philosophy, problem formulation and having the self-belief to "do what you like" that will stay with me forever. Thank you Oliver for your constant encouragement, kindness and guidance. Thank you for having faith in me to let me pursue my ambitious ideas even when they did not work. There were a number of occasions when I went to you overwhelmed with a complex problem and you showed me how to break it down into manageable chunks. You showed me that research is much more fun with collaborators and friends. Your deep love for robotics has been infectious. I will miss our lively meetings.

I want to thank my thesis committee members Reid and George for their valuable feedback and insightful research discussions. Reid has an incredible ability to cut through complexity and ask the important questions. Thank you for helping me think more clearly about the thesis and for encouraging me to work on human-robot interaction. George's work on skill chaining and learning parameterized skills have been a major influence on this thesis. He taught me to have ambitious research goals and not be easily swayed by what is in fashion. Thank you for your constant encouragement, your confidence in me and for helping me see the big picture. I must also thank Chris Atkeson for being on my qualifier committee and motivating me to work on challenging manipulation problems.

I am greatly indebted to many amazing people who have mentored me over the years. Devesh Jha and Diego Romeres gave me the opportunity to spend a delightful summer at MERL in Cambridge and the intellectual freedom to pursue interesting questions. I am thankful to Partha Pratim Chakrabarti for getting me interested in computer science through his Programming and Data Structures class at IIT Kharagpur and then advising my Masters' thesis on heuristic search. Venkatraman Narayanan mentored me during my time as an intern in SBPL and catalyzed my interest in research. Shushman Choudhury encouraged me to apply to CMU and then to work with Max. Anupam Gupta and Pravesh Kothari taught my favourite course at CMU on Advanced Approximation Algorithms which ended up being quite useful

for my thesis. I am also deeply grateful to my middle school teachers Krishna Sir and Sheo Sir without whose guidance I couldn't have come this far.

I have had the good fortune of working with wonderful collaborators during my time at CMU. Tabitha Lee has been an amazing friend, collaborator, co-author and also my neighbour. I had many fun discussions on human-robot interaction with Pat Callaghan and Michelle Zhao. Jacky Liang and Mohit Sharma taught me much of what I know about robot learning. Their passion for robot learning and commitment to robotics have greatly shaped my research ideas. I couldn't have done any of my real robot experiments without the help of Kevin Zhang. He also happens to be the only person I know with a stronger sweet tooth than me. I thank Aditya Agarwal, Suhail Saleem, Wei Du and Shanshan Xie for working with me on my first research project at CMU. Saumya Saxena kept the lab lively and full of pastries. I thank Alex LaGrassa for teaching me the principles of good collaboration and for their insightful questions. I thank Raghav Sood for collaborating on one of my early projects on search-based planning for navigation. I thank Xinyu Wang and Zilin Si for working on the Punyo gripper with me. I thank Yorai Shaoul for fun discussions on multi-agent planning with diffusion models.

I was fortunate enough to be part of two amazing labs. Both IAM lab and SBPL have been my home away from home. I will miss the camaraderie and the banter. From the IAM lab, in addition to the people I directly collaborated with, I am grateful to Mark Lee, Sarvesh Patil, Ami Sawhney, Steven Lee and Erin Zhang. I want to especially thank Sarvesh and Mark for organizing the most fun lab social events. From SBPL, I thank Ram Natarajan, Tushar Kusnur, Rishi Veerapaneni, Yash Oza, Shohin Mukherjee, Dhruv Saxena, Itamar Mishani, Fahad Islam, Oren Salzman, Anirudh Vemula, Anahita Mohseni-Kabir, Manash Pratim Das and Andrew Dornbush. Thanks to Ram for teaching me trajectory optimization and being an important sounding board. I want to thank Rishi for all his jokes, fun research discussions and for taking the advanced approximation algorithms course with me.

I want to extend my gratitude to all the friends from the broader RI and CMU community who made my time here memorable: Jaskaran Grover, Ashwin Khadke, Mike Lee, Tanya Marwah, Ada Taylor, Abhijat Biswas, Angela Chen, Kate Shih, Thomas Weng, Cherie Ho, Swaminathan Gurumurthy, Pranav Khadpe, Mohamad Qadri, Tejus Gupta, Blake Buchanan, Arkadeep Chaudhury, Abitha Thankaraj, Jeong Hun, Suresh Jayaraman, Nikolaos Gkanatsios, Pragna Mannam, Victoria Dean, Xianyi

Cheng, Murtaza Dalal, Jayanth Krishna Mogali, Joseph Campbell, Simon Stepputtis, Ankit Bhatia, Brian Okorn and Adithya Murali. I am grateful to Leo Keselman for guiding me during the early phase of my PhD. I thank the administrative staff at RI, Peggy, Suzanne and Jean for always being there to help, despite their busy schedules. Special thanks to Rachel Burcin for always greeting me with a warm hug. I will miss her jokes.

I consider myself incredibly lucky to have met Michelle Yeh during the pandemic in Pittsburgh. She made Pittsburgh feel like home with her love and support. Thank you for believing in me. I shared many ups and downs of research and life in Pittsburgh with Kshitij Goel. Thank you for being such a patient listener. Thanks to Harsh Gupta for introducing me to SymPy during my undergrad at IIT Kharagpur and then becoming such a dear friend. From IIT, I thank Abhinav Agarwalla, Ankush Chatterjee, Arnav Jain, Arna Ghosh, Krishna Bagadia, Rishabh Madan, Kumar Krishna Agrawal, Bismaya Sahoo, Tanmay Patil and Samuel Anudeep. I am grateful to my middle school friends from St. Xavier's School, Hazaribagh: Manas Tiwari, Nilesh Rana, Kumar Mantavya and Shriti Priya for being the most supportive friends I could have asked for. No trip to India is complete without meeting them.

Finally, none of this would have been possible without the love, support and sacrifices of my family. My brother Priyanshu is the joy of my life. I am grateful to my parents Kumkum Pandey and Pushkar Pandey for believing in me and leaving no stone unturned to ensure that I get the best education. They instilled in me the value of hard work and the confidence to make my own choices. I owe this PhD and all my accomplishments to them.

For Mummy, Papa and Bauwa

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Figure 1.1: Examples of robots operating in household and collaborative manufacturing settings. Robots need a diverse repertoire of capable motor skills to succeed in such complex and unstructured environments. This thesis advocates that robots should actively and autonomously hone their skills and acquire new ones to achieve their objectives.

It is exciting to live in a day and age where thousands of robots regularly manage massive warehouses and rovers are deployed on Mars to aid in space exploration. Robots hold the promise of becoming an integral part of human life by helping us fold laundry (Miller et al., 2012) in our homes, monitor crops (Lee et al., 2024) out on farms, and collaborate with human workers (Pearce et al., 2018) in factories. To fulfill this promise, we need general-purpose robots that can robustly function in diverse environments, autonomously adapt to novel situations and effectively collaborate with human partners. Despite significant progress, robots today are nowhere close to

achieving these three important capabilities. First, robots lack the physical dexterity required to even perform everyday tasks such as making an omelette and assembling furniture. Second, robots are still largely restricted to highly structured and controlled settings where they are deployed with carefully designed and often hand-engineered skills. Finally, most robots are either separated from humans by a fence in factories or they treat humans as obstacles to be avoided.

Much of recent progress towards general-purpose robots has been powered by data-driven methods such as learning from demonstrations (Ravichandar et al., 2020) and deep reinforcement learning (Lillicrap et al., 2015). Some of the impressive advances made by this approach include generalist robot policies (Brohan et al., 2022) that can learn a variety manipulation skills through imitation and learning dexterous manipulation, such as, solving a Rubik's cube with a human-like hand (Akkaya et al., 2019) using reinforcement learning. However, these methods require massive amounts of data which is a challenge because robot data collection requires expensive physical robots and significant human labor. Consequently, these methods work well only in narrow domains for which sufficient amount of expert data is available and lack the robustness necessary for real-world deployment.

Recent data collection efforts like the Open-X Embodiment dataset (Collaboration et al., 2023) have sought to address this limitation by collecting and consolidating data from multiple institutions. While such endeavours are certainly promising and perhaps necessary to deploy robots in open-world environments (Horvitz, 2008), they are by no means sufficient. A robot operating in unstructured environments has to be prepared to handle situations that are beyond its pre-trained capabilities. When such situations arise, and they are often the norm rather than the exception, an intelligent and autonomous robot ought to take charge of its own learning to go beyond canonical datasets. Such a robot would need to actively collect experiences in its domain to improve and extend its skillset to achieve its long-term objective. This in-domain learning would require careful reasoning to decide which experiences are relevant to the objective and how to efficiently collect those experiences while ensuring that the robot is useful to its human partners.

The goal of this thesis is to make progress towards building such collaborative robots that can operate reliably in less structured settings. Our vision is to build robots that learn autonomously to be useful to their human partners. Ideally (as

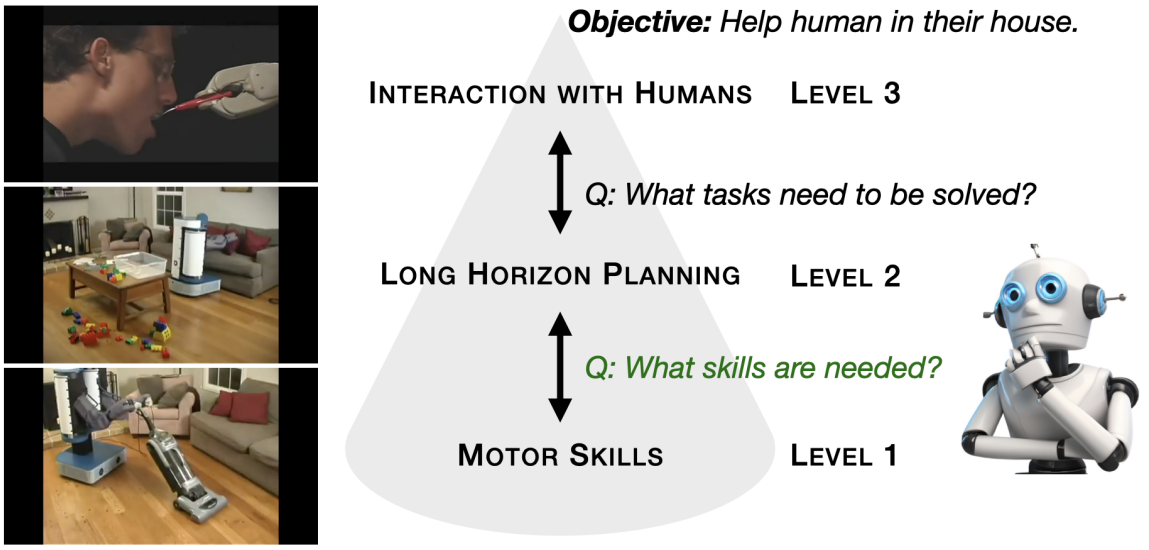


Figure 1.2: We organize robot capabilities into three levels of increasing difficulty. We wish to develop robots that can plan not only how to solve long-horizon tasks but also how to efficiently learn the right set of skills required to achieve a given objective. The real-world images used in this figure are from Eric Berger and Keenan Wyrobek at the Salisbury Robotics Lab, Stanford University, 2007.

visualized in figure 1.2), we would like a user to give only a high-level objective, such as, "help me clean my house" to a robot. The robot should have the long-horizon planning capability to achieve this objective. If the robot realizes that it lacks certain skills, for example, the skill to wipe tables, then it should efficiently learn the missing skills by requesting human demonstrations or in simulation. On the other hand, if the missing skill is difficult to learn, for example, the skill to fold shirts, then the robot should request help from the user. This thesis pursues three broad themes of continual learning, active learning and human-robot collaboration towards achieving this long-term goal.

Continual Learning

The standard train-test paradigm of machine learning is ill-suited to robotics. A robot may not fully know all the situations it may face during deployment. Hence, we adopt the paradigm of continual learning in which learning is done continually during deployment as and when new situations arise. Chapter 3 proposes a human-in-the-loop learning algorithm for learning on the job as new tasks arrive while chapters 4

and 5 take the approach of continually discovering a robot's failures and learning new skills to recover from them. This allows the robot to gradually expand its capabilities to cover the situations it is likely to encounter in its domain.

Active Learning

The high cost of data collection in robotics necessitates careful deliberation to minimize the cost of data. This requires identifying skills that are most relevant to the overall objective. In chapter 3, our discuss an approach that prioritizes skills that cover multiple tasks. In chapter 4, our proposed algorithm identifies the skills that are most relevant to the overall multistep task by using a task planner. While we focus on choosing which skills to learn in this thesis, the active learning idea can be extended to also decide how to learn when multiple sources of data are available, *viz.*, real-world, simulation, human, etc.

Human-Robot Collaboration

Collaborative robots have the advantage of being able to seek help from and assist their human teammates. Hence, they do not need to be fully autonomous to be useful. In fact, a semi-autonomous strategy may often be preferable, where an agent transfers control to a teammate if it improves the probability of achieving the goal (Hexmoor, 2000). For example, in a game of soccer a player may choose to pass the ball to a fellow player if it improves the chances of scoring. Such situated reasoning, wherein a robot's role in the team depends on the situation, can inform whether or not the robot should learn a particular skill. We leverage this insight in chapter 3 to assign tasks that are difficult to teach and one-off to the human teammate. This allows the robot to assist the human by focusing on doing repetitive tasks without needing too many demonstrations.

1.2 P2L: Plan to Learn

> Intelligence is the ability to use optimally limited resources to achieve goals. (Ray Kurzweil)

We integrate these ideas of continual, active and collaborative robot learning in a novel **Plan to Learn** (P2L) framework. We mathematically formalize and study

P2L through both a practical and theoretical lens. The main question P2L seeks to answer is *how should a robot optimally utilize the resources at its disposal, including time, computation and human help, to acquire skills to achieve its objective.* We assume a robot is operating autonomously in a domain with task distribution $\mathcal{D}$. The performance of the robot on a task instance τ is captured by the objective function $J(\tau, \mathcal{L})$ which is a function of the robot's skillset $\mathcal{L} = \{\pi_1, \ldots, \pi_n\}$. Computing the performance may require solving a planning problem. For example, this corresponds to a task allocation problem in chapter 3 and a task planning problem in chapter 4. The expected performance of the robot is

$$\mathbb{E}_{\tau \sim \mathcal{D}} J(\tau, \mathcal{L}) \tag{1.1}$$

Given infinite resources, the robot could fully learn all the skills and maximize its performance. However, in reality data collection and learning is expensive and time consuming. Hence, we assume that every data point collected for a skill has an associated skill-specific cost c_{learn}. It is important that the robot quickly identifies and focuses on the right skills to ensure efficient learning. Let x_i be the amount of data collected for skill π_i and $\mathcal{L}'$ be the resulting improved skillset. Here, we assume a single source of data. Additional sources of data with different cost of collection can similarly be incorporated in this framework. Our P2L framework trades off task performance against the cost of learning by solving the following optimization problem.

$$\max \mathbb{E}_{\tau \sim \mathcal{D}} J(\tau, \mathcal{L}') - \sum_i x_i \cdot c_{learn} \tag{1.2}$$

This is a meta planning problem whose solution provides the robot with a strategy to optimally use its time and resources to learn a skillset that is optimized for its specific domain. We consider a more general human-robot team setting in chapter 3, where the robot also has the option to delegate some tasks to the human teammate. Delegation may useful if it is easier for the robot to delegate a task than to learn how to do it on its own. We develop principled algorithms to solve the P2L problem in two challenging domains of human-robot teaming and multi-step manipulation. This thesis shows that *robots can learn more efficiently and collaborate more effectively by deliberatively choosing what to learn, i.e., by planning to learn.*

1.3 Thesis Overview

Chapter 2 provide useful context and background necessary to understand our contributions. The following chapters present the core contributions of this thesis.

Chapter 3 discusses the P2L problem in a human-robot team scenario. We consider collaborative manufacturing where a human-robot team needs to complete a given sequence of assembly tasks in minimum execution cost. We assume that all future tasks come in a fixed sequence and are known ahead of time. To this end, we propose the Act, Delegate or Learn (ADL) planner (Vats et al., 2022) that determines which tasks to assign to a robot, which tasks to assign to a human and which skills to teach the robot.

Chapter 4 moves away from human-robot collaboration and explores P2L in the context of multistep manipulation tasks. We assume that the robot has a symbolic representation of the task in the form of a skill graph which can be used to plan. We propose Metareasoning for Skill Learning (MetaReSkill) (Vats et al., 2023) to efficiently learn task-relevant skills using this symbolic representation. MetaReSkill uses an estimated skill improvement model in conjunction with a skill planner to actively learn the most task-relevant skills.

Chapter 5 relaxes the assumption of having a symbolic skill graph and introduces a reinforcement learning-based approach RecoveryChaining (Vats et al., 2024) to solve the P2L problem. RecoveryChaining uses a hybrid action space to efficiently learn robust recovery policies that can be chained with model-based controllers. The action space contains temporally extended nominal options that transfer control to a specific nominal controller.

Chapter 6 concludes with a summary of contributions, discussion and directions for future work.

2 Background

This chapter surveys the broader literature on meta-reasoning, active machine learning and hierarchical reinforcement learning. Additional background and related work specific to each chapter is discussed in the relevant chapter.

2.1 Meta-reasoning

The human mind is able to solve a wide range of complex perception and decision making problems despite having access to only a fixed amount of computational resources and limited experience. Griffiths et al. (2019) argue that, "the constraints that characterize human cognition are also intrinsic to developing key components of what we recognize as intelligence". This ability to make the most of limited resources has enabled humans to become efficient and general-purpose learners. One of the key components of this general intelligence of the human brain is *meta-reasoning* (Ackerman and Thompson, 2017). Meta-reasoning refers to the processes in the brain that monitor the progress of human reasoning activities, *i.e.*, metacognitive monitoring and allocate time and effort devoted to different cognitive tasks, *i.e.*, metacognitive control. In particular, *rational metareasoning* (Russell and Wefald, 1991) treats computations as actions to be selected from based on their expected utilities. For example, an agent could compute the value of computation for every computation that could be executed and pursue the computation with the highest value. We adopt a similar approach in this thesis, wherein some of the robot actions are meta-actions that correspond to spending computational resources or querying humans for data collection.

Meta-reasoning has been used in artificial intelligence to design agents that can uti-

lize computation efficiently to operate in resource constrained environments. Robots often use anytime motion planning for acting under tight time constraints. Anytime algorithms (Zilberstein, 1996) are algorithms that can improve the quality of solutions with additional computation time. Meta-reasoning has been used in this context to decide when to stop planning and start executing the current best plan (Svegliato et al., 2018; Sung et al., 2021). In many practical situations, it is preferable to execute a sub-optimal plan instead of waiting to compute a better plan. This trade-off between solution quality and computation time is typically encoded in a utility function which can be optimized through meta-reasoning.

Budgeted Machine Learning

Many machine learning applications require good classification performance under budget constraints and hence can take advantage of meta-reasoning. In particular, budgeted machine learning (Gao and Koller, 2011; Benbouzid et al., 2012; Karayev et al., 2013) is a relatively new field that studies the trade-off between prediction accuracy and prediction cost in resource constrained settings. In some settings, the constraint is during training. For example, Li et al. (2019) propose to adjust the learning rate schedule to maximize performance of a neural network. In some medical applications, the agent has access to labels but needs to pay for each attribute of a sample. Here, Deng et al. (2013) propose multi-armed bandit-based and other heuristic ways to pick training examples for training. In other applications, there are resource constraints during inference. Here, prior works seek to learn a classifier that utilizes inexpensive sensing and classification modalities as much as possible. Techniques include learning (Wang et al., 2014; Trapeznikov and Saligrama, 2013) feature acquisition rules using empirical risk minimization and formulating (Benbouzid et al., 2012; Karayev et al., 2013) the acquisition problem as an MDP.

2.2 Active Learning

The standard machine learning (ML) paradigm focuses on learning the most performant model from a given dataset. However, it is difficult to obtain sufficient data in many domains in which data collection is expensive, for example, if a human needs to be queried to generate the label for a training instance. An important observation

is that not all data points are equally important from the perspective of model training. For example, in the case of classification, data that lie close to the separation boundary is more informative than data that clearly belongs to a specific class. Active learning (AL) (Aggarwal et al., 2014; Settles, 2009) is a sub-field of ML that leverages this observation to achieve the best model performance while spending the minimum effort on data collection. Every active learning system has access to an oracle that generates the label for a query. The oracle could be a human or a computationally expensive algorithm. The role of the active learning system is to intelligently pose queries to the oracle.

A number of strategies have been developed for different active learning scenarios. We highlight a few here.

1. **Uncertainty Sampling** (Lewis, 1995). The learner maintains an estimate of its confidence in different parts of the input space. It then queries instances that it is most uncertain about. For example, when training a Bayesian classifier, a learner should label instances whose predicted probability is close to 0.5.

2. **Expected Error Reduction** (Roy and McCallum, 2001). In contrast to the uncertainty sampling approach that maximizes the label uncertainty of its queries, this approach minimizes the uncertainty in prediction of the remaining instances. The learner computes the expected reduction in its error if it is retrained with an additional data-point for every instance and queries the label of the instance that promises the highest reduction.

3. **Variance Reduction** (Cohn, 1993). The error reduction approach is expensive as it computes the error reduction in terms of all the remaining instances. When using variance reduction, the learner tries to reduce future expected error by minimizing the variance of the model as per the bias-variance decomposition (Geman et al., 1992) of the expected error. This can be expressed in closed form and is hence computationally cheaper.

4. **Expected Model Change** (Settles et al., 2007). This is a decision-theoretic approach in which the learner selects an instance that would impart the greatest change to the model if the learner knew the label.

While typical AL algorithms select one query at a time, neural networks require batches of data. Recent works on deep active learning use core-set selection (Sener

and Savarese, 2018; Yehuda et al., 2022) to select batches of samples for training neural networks. They formulate the selection problem either as a facility location or a covering problem (Vazirani, 2013). While both these problems are NP-hard, these approaches use their polynomial-time approximation algorithms to solve them. We will use a similar idea in chapter 3 to develop approximation algorithms for planning to learn. AL has also been extended to the multi-task setting by Harpale and Yang (2010) by using a *utility gain* that not only improves performance on one task but also performance on other tasks.

Active learning has been an important area of research in the robot learning community because of the high cost of data collection. AL methods have been proposed for learning parameterized robot skills from fewer samples (Fabisch and Metzen, 2014; Da Silva et al., 2014). AL techniques are particularly relevant to robot learning from demonstrations as each demonstration involves significant amount of human effort. AL can allow the robot to be an active participant of its learning process and reduce human effort (Chernova and Veloso, 2009). Recent methods like INQUIRE (Fitzgerald et al.) even allow the robot to actively choose the appropriate query type to learn from humans.

2.3 Hierarchical Reinforcement Learning

Hierarchical reinforcement learning (HRL) (Pateria et al., 2021) is an approach for solving long-horizon decision making problems by decomposing them into smaller sub-tasks to enable easier policy learning and better generalization. This decomposition allows decisions to be made at higher levels of abstraction without having to deal with the low-level details. Each sub-task may itself be a reinforcement learning problem which can be solved independently to learn a low level policy. A higher-level policy is then learned on top of these low level policies. In general, HRL can have a multi-level hierarchy of sub-tasks. However, it is more common in robotics to have a bilevel hierarchy. We use bilevel hierarchies in this thesis as well in which temporally extended actions, called *options*, are used as an abstraction over low level robot actions for efficient learning.

Options Framework

We model each robot skill as an option as per the options framework (Sutton et al., 1999; Konidaris et al., 2018). Each option consists of three components:

1. robot *control policy* π which chooses the robot's low level actions.
2. *initiation set* $\mathcal{I}$, also sometimes called a *precondition*, which defines the states from which the option can be executed.
3. *termination condition* β which defines the states in which the option must terminate.

Skill Preconditions

We use a probabilistic version of initiation sets, where the precondition (Konidaris et al., 2015) of a skill is a classifier $\rho : \Theta \rightarrow [0, 1]$ that takes in features describing a state and returns the probability that the skill can be successfully executed from that state. This classifier is usually trained by executing the skill from various start states to generate success labels (Kroemer and Sukhatme, 2016).

3 P2L for Human-Robot Teams: Act, Delegate or Learn

The vast majority of industrial robots today are pre-programmed by domain experts to do highly specific tasks. These robots lack the flexibility to recover from errors and adapt to even small changes in their task or environment. While robot learning (Kroemer et al., 2021) promises to reduce pre-programming and improve generalization, full robot autonomy for challenging tasks in unstructured environments is likely to be out of reach for the foreseeable future (Kragic et al., 2018). Even when full autonomy is technically feasible, it may not always be affordable as teaching robots to do new tasks requires expensive data collection and extensive tooling. In this chapter[1], we consider the middle ground of collaborative robotics wherein robots operate alongside humans. Collaborative robotics has been of great interest for applications in manufacturing and service industries because it combines the complementary strengths of humans and robots (Vicentini, 2021).

Consider a manufacturing facility with collaborative robots that gets its orders at the start of each day. How should the human and robot workers work together to efficiently fulfil all the orders? Traditionally, planners for task allocation (Gombolay et al., 2013; Shannon et al., 2016) have been used to allocate tasks to humans and robots based on their capabilities. However, they assume that robots comes pre-trained or pre-programmed with a fixed set of capabilities. This static view of robots misses the opportunity to have robots learn on the job from their human teammates. In particular, techniques from learning from human demonstrations(LfD) (Argall et al., 2009; Chernova and Thomaz, 2014) can be used to teach the robot new skills that empower it to better assist the human on future tasks. This leads to a

[1]This chapter is based on material from Vats et al. (2022).

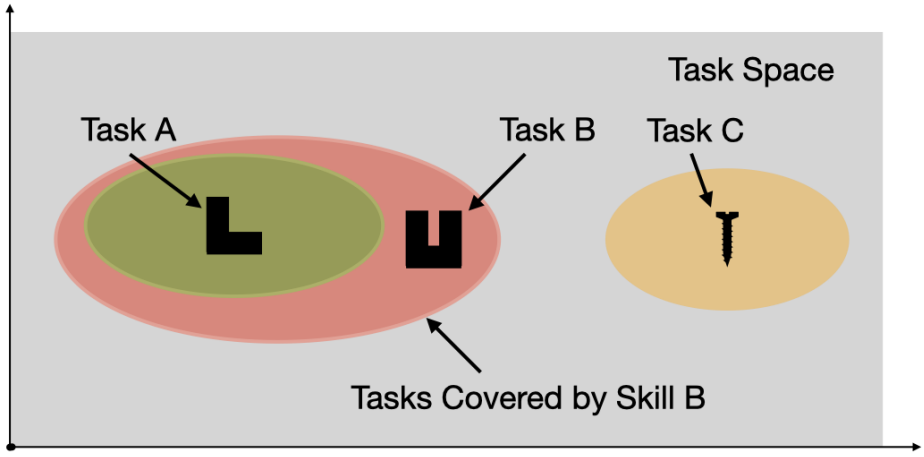


Figure 3.1: Consider three assembly tasks visualized in a 2D task state-space. Each colored oval covers tasks that can be solved by a specific robot skill. Note that skill B covers more tasks than the other two skills while task C remains uncovered even after learning skill B. Our framework schedules teaching of only those skills that cover enough future tasks to offset the cost of robot teaching. Remaining tasks are delegated to a human for completion.

number of important questions: Should additional robot teaching be done? If so, on which tasks? What tasks should be done by robots and what tasks by humans?

To this end, we propose a decision making framework **A**ct, **D**elegate, or **L**earn (ADL) that jointly reasons about task allocation and robot learning. Learning extends the capabilities of a robot, which is useful when similar tasks are expected to be encountered again in the future. One-off tasks that are beyond the capability of the robot, on the other hand, are allocated to the human. We focus on a setting where tasks come in a *known fixed sequence*. This is motivated by time and cost critical domains like agile assembly lines in factories where a diverse but known set of orders need to be fulfilled with minimum human and robot effort. While human help is available, it is at a premium. Hence, we would like to use it optimally so as to minimize the overall effort. We show the ADL framework provides significant reduction in human workload compared to task allocation and robot learning approaches.

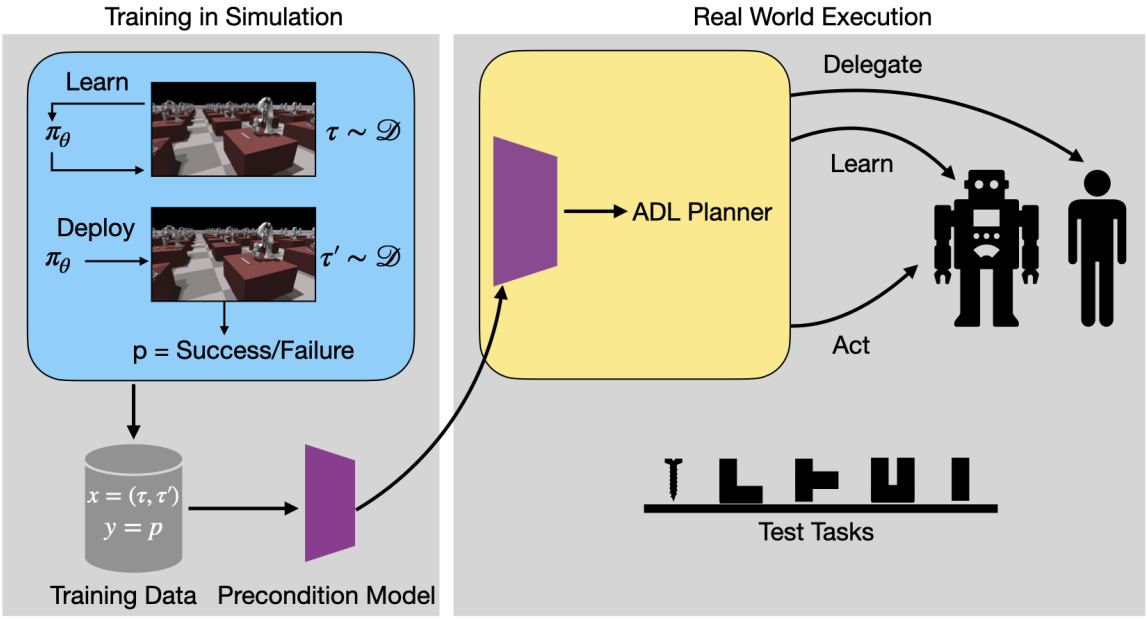


Figure 3.2: Overall approach: (i) **Training in Simulation** Skills are learned (using RL) and deployed on tasks $\tau \sim \mathcal{D}$ to collect data on what other tasks can be solved by a skill learned for a particular task. A precondition prediction model is trained using this data. (ii) **Real World Execution** Our planner makes use of the learned model to decide when the robot should attempt a task, when it should delegate to a human and when it should learn a new skill for a task.

3.1 Related Work

Task Allocation

Task or function allocation is the decision making problem of determining which functions should be performed by machines and which by humans (Inagaki et al., 2003; Fitts et al., 1951). While a number of strategies have been proposed, the one closest to our work is *economic allocation* (Inagaki et al., 2003; Dearden et al., 2000) which finds an allocation that ensures economic efficiency.

Adaptive Automation

Adaptive automation can accommodate changes in the environment or the human for function allocation. A number of frameworks have been proposed over several decades (Rouse, 1976; Scerbo, 1996; Kaber et al., 2001; Sheridan, 2011) which focus on optimizing operator workload, attention and efficiency. Consequently, their focus

has been on modeling the human (Pew, 1969; Chen and Barnes, 2014; Shannon et al., 2017). (Basich et al., 2020) recently propose an interactive model of autonomy, where a system learns a model of its competence online. All these strategies assume that the robot has certain fixed capabilities

Learning from Demonstrations

There are three main categories(Ravichandar et al., 2020) of LfD– kinesthetic teaching, teleoperation and passive observation. Kinesthetic teaching is the most common approach for providing demos in manufacturing and health-care (Ravichandar et al., 2020), while tele-operation does not require the user to be copresent with the robot. Passive observation usually requires multiple demos (Hayes and Scassellati, 2014), special instrumentation (motion capture, force-torque sensors) depending on the task and is complicated to solve due to the need for retargeting. Despite recent progress, teaching robots generalizable skills still requires significant human effort.

Interactive Robot Learning

Interactive robot learning (Chernova and Veloso, 2009; Cakmak et al., 2010; Gribovskaya et al., 2010) seeks to make it easier for humans to teach robots, for example, by minimizing the number of demos. A popular approach is for the robot to *ask for help* when it is not confident or uncertain. In particular, Confidence-Based Autonomy (Chernova and Veloso, 2009) uses classification confidence to choose between autonomous execution and request for a demo and Xie et al. proactively ask for help from a human when the agent enters irreversible states. ThriftyDAgger (Hoque et al., 2021) uses estimated probability of task success to determine when to solicit human interventions. Different from these approaches, our framework takes into account the cost of human intervention which depends on the type of intervention, such as, demonstration or recovery from failures. Rigter et al. (2020) propose a multi-armed bandit approach to explicitly minimize a cost function corresponding to human exertion. However, their decision making framework is myopic and does not take into account the effect of a demonstration on improving robot performance on future tasks.

Multi-task Learning

Deisenroth et al. (2014) look at learning a single policy in a multi-task setting with a continuous set of tasks. Kupcsik et al. (2017) learn a two level policy where the low level policy controls the robot for a given context and the high level policy generalizes among contexts. In contrast, we take a library of independent skills approach, where generalization happens only at the lower level.

3.2 Act, Delegate or Learn Framework

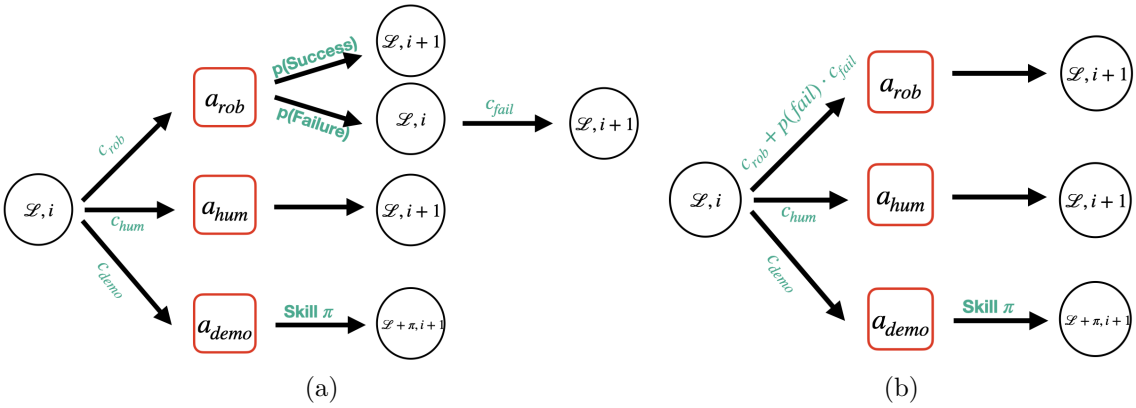


Figure 3.3: (a) **Transition Model.** Our MDP has three actions: a_{rob}, a_{hum} and a_{demo} with associated costs of c_{rob}, c_{hum} and c_{demo} corresponding to the options act, delegate and learn. A human intervenes to complete a task if robot execution fails. We assume that a human can complete all the tasks and is available at all times to teach the robot. (b) **Simplified Transition Model.** We can replace the two stochastic outcomes due to a_{rob} with a single outcome whose cost is an expectation over them.

We are interested in completing a known sequence of tasks with minimum total expected human and robot effort. Before deployment, we are provided an approximate distribution $\mathcal{D}$ of tasks that are expected to be encountered. The robot may be pre-trained with a set of skills based on this knowledge. The actual tasks and the order in which they need to be done are revealed only at test time. At this stage, a decision needs to be made for every task: should the robot do the task, should it delegate the task to a human or should it ask to be taught how to do the task? We require that every task be completed. Hence, each robot failure incurs additional cost due to

human intervention to complete the task and correct the setup. Finally, we assume that a human is available at all times to intervene if needed - either to correct a robot failure or to teach it, for example, by providing demonstrations.

A popular approach for solving related tasks is to learn a parameterized skill (da Silva et al., 2012), that adapts the policy based on changes in the task. This approach is practical if only some aspects of the task can change. Adapting to various changes in the tasks requires a more complex skill parameterization that makes the learning problem harder and more sample complex. An alternative approach, which we take in this work, is to have the robot maintain a *library* $\mathcal{L} = \{\pi_1, \cdots, \pi_n\}$ of skills, each of which is learned on a narrow task distribution from demonstrations. Given a task τ, the robot picks an appropriate skill for it. by selecting a skill with the highest probability of success: $\text{argmax}_{\pi \in \mathcal{L}} \rho_\pi(\tau)$. This representation has a number of advantages over learning a monolithic skill, chiefly, modularity, allowing local updates and providing alternatives in case of execution failure.

3.2.1 Problem Definition

The ADL planning problem is a stochastic shortest path (SSP) problem $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G})$ (Kolobov, 2012) where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition model, $\mathcal{C} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is a cost function and $\mathcal{G} \subset \mathcal{S}$ is a set of goal states. We define each of these components of the MDP for our problem:

State Space

Each state $s \in \mathcal{S}$ is a tuple $\langle \mathcal{L}, k \rangle$, where $\mathcal{L}$ is the skill library of the robot at that state and k refers to the tasks completed so far.

Action Space

$\mathcal{A} = \{a_{rob}, a_{hum}, a_{demo}\}$, where a_{rob} implies that the robot attempts to solve the task, a_{hum} implies that the human solves it and a_{demo} implies that the human teaches the robot a new skill for it in addition to solving it.

Transition Function

$\mathcal{T}$ models whether the skill library got updated or a task was completed after an action. Though the outcome of robot execution is stochastic, we can convert it into

a deterministic MDP by taking an expectation over the two outcomes (see figure 3.3 for details). We will be using the resulting simplified transition model in the rest of the paper. The transition model makes it clear that a_{rob} and a_{hum} do not affect the skill library in any way. On the other hand a_{demo} updates the library by adding a new skill π to its repertoire.

Cost Function

The cost function is defined as

$$\mathcal{C}(s_i, a) = \begin{cases} c_{rob}(i) + \Pr(fail) \cdot c_{fail}(i) & a = a_{rob} \\ c_{hum}(i) & a = a_{hum} \\ c_{demo}(i) & a = a_{demo} \end{cases} \tag{3.1}$$

where, $\Pr(fail) = 1 - \max_{\pi \in \mathcal{L}} \rho_\pi(\tau_i)$. The cost of a robot execution includes the cost of a potential failure and hence depends on the robot's skill library. c_{rob}, c_{hum} and c_{demo} are domain and task dependent costs specified by a domain expert. For example, in manufacturing, where minimizing the *economic cost* of production is crucial, c_{rob} could reflect the cost of operating a robot, while c_{hum} and c_{demo} could depend on the efficiency of a human collaborator. There exist a number of approaches (Chen and Barnes, 2014; Shannon et al., 2017; Gombolay et al., 2017) to model human performance. c_{fail} corresponds to the difficulty of fixing a mistake made by the robot. In some domains, this could be as simple as asking a human in the factory to complete the remaining task, while in others, it may be high if there is a risk of damage due to a failure.

A goal state is reached once all the tasks have been completed. Let $\{\tau_i\}_{i=1}^n$ be the sequence of tasks and $\eta = \{\eta_i\}_{i=1}^n$ be the sequence of actions taken. Then, the expected cost of execution is: $J(\eta) = \sum_{i=1}^n \mathcal{C}(s_i, \eta_i)$, where $s_{i+1} = \mathcal{T}(s_i, \eta_i)$ and our goal is to find an optimal plan $\eta^* = \text{argmin}_{\eta \in \mathcal{A}^n} J(\eta)$.

3.3 Approach

3.3.1 Mixed Integer Programming Formulation

The standard techniques used to solve a deterministic SSP are graph search algorithms like Dijkstra's algorithm and A*. Unfortunately, the search graph induced by our problem has *exponentially* many states in the number of tasks to be done. Though A*-like algorithms can leverage heuristics to speed-up search, their performance is highly dependent on the quality of the heuristic and hence incur substantial overhead for designing good heuristics.

Motivated by this, we propose a mixed integer programming (MIP) formulation of the SSP which can be solved using off-the-shelf solvers without the need to design heuristics. These solvers provide high quality solutions (with sub-optimality bounds) and are highly scalable.

We introduce decision variables for every task: $x_i, y_i, z_i \in \{0, 1\}, w_i \in [0, 1], \forall i \in \{1, \cdots, n\}$. Let binary decision variable x_i be 1 if a demo is sought on task τ_i, y_i be 1 if a human is asked to solve it and z_i be 1 if the robot is asked to attempt the task. As the robot may fail in its attempt, we model the probability of human intervention with a continuous decision variable w_i– note that it is non-zero only if robot execution is chosen for a task. We exercise indirect control over w_i via the probability of failure of the action taken.

Our overall objective is:

$$\min \sum_{i=1}^{n} c_{demo}(i)x_i + c_{hum}(i)y_i + c_{rob}(i)z_i + c_{fail}(i)w_i$$

where, $z_i = 1 - x_i - y_i$ as we allow exactly one of these three actions for a task. Hence, the objective can be simplified.

$$\min \sum_{i=1}^{n} c'_{demo}(i)x_i + c'_{hum}(i)y_i + c_{fail}(i)w_i \tag{3.2}$$

where $\forall i \in \{1, \cdots, n\}$

$$
\begin{aligned}
c'_{demo}(i) &= c_{demo}(i) - c_{rob}(i) \\
c'_{hum}(i) &= c_{hum}(i) - c_{rob}(i) \\
w_i &= 1 - \max\{\rho_0, \rho_1(\tau_i)x_1, \cdots, \rho_i(\tau_i)x_i, y_i\}
\end{aligned}
$$

The max term in the last equation is a maximization over the success probabilities of the available ways to solve the task – using pre-trained skills (with precondition ρ_0), learning new skills (with preconditions $\rho_1 \cdots, \rho_n$) and delegating to a human (represented by y_i). y_i is 1 if the robot delegates the task to a human, in which case we are assured of task completion. In its current form this program is not linear due to the max operation. However, we can easily convert it into a linear MIP by introducing additional binary decision variables. Some solvers like Gurobi (Gurobi Optimization, 2021) can directly take this program and do the linearization under the hood.

3.3.2 Precondition Prediction Model

A key requirement of our planner is the ability to foresee the benefit of robot teaching *before* committing to it. Past works (Kroemer and Sukhatme, 2016; Sharma and Kroemer, 2020) have looked at the problem of precondition learning, wherein a classifier is trained for an existing skill to predict what other tasks can be solved by it. By contrast, we need to predict the preconditions of a skill that *will* be learned if we choose to teach the robot– a precondition prediction problem. Our proposed solution is to learn a classifier $P : (\tau_{train}, \tau_{test}) \rightarrow [0, 1]$ (figure 3.4) that takes as input a train task and a test task and predicts whether a robot trained on the former will be able to solve the latter. Intuitively, this can be thought of as learning a similarity metric between tasks.

We collect training data for the precondition model using algorithm 1. This can be prohibitively expensive as we need to learn robot policies to generate labels. We get around this limitation by observing that we do not need to transfer robot skills from sim2real but only task relationships— the former requires high fidelity simulation while the latter does not. It is often the case that a lower dimensional state representation is sufficient to discriminate between tasks. The key is to simplify the problem such that inter-task relationships remain intact– tasks that are similar/dissimilar in

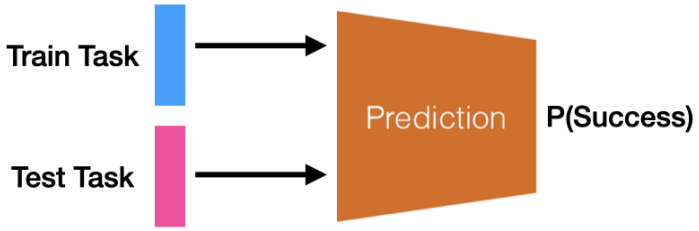


Figure 3.4: Precondition prediction model predicts the probability of success on a test task τ' after the robot has been trained on a task τ.

the real world should remain so in simulation and vice versa. Concretely, we define an *abstraction* (Li et al., 2006; Konidaris and Barto, 2009a) M as a pair of functions (f, g) such that $f : S \to S'$ maps the original problem state space S to a smaller state space S' and $g : A \to A'$ maps the full action space to a smaller action space. The specific state and action abstraction to be used in training are provided as domain knowledge.

Algorithm 1 Data collection in abstract simulation.

```
1:  procedure GetTrainingData(m, n)
2:      X ← φ, Y ← φ
3:      S ← Sample m tasks from D
4:      for i ∈ {1, ··· , n} do
5:          Sample τ from D
6:          π ← Learn policy for τ
7:          for τ' ∈ S do
8:              x ← (τ, τ')
9:              y ← Evaluate π on τ'
10:             X.insert(x), Y.insert(y)
        return X, Y
```

3.4 Approximation Algorithm

The formulation presented in the previous section is intractable when the number of tasks is large as MIP is NP-hard. In this section, we show that ADL is an instance of the facility location problem (Vazirani, 2013). While solving the facility location problem optimally is NP-hard, it has $O(\log n)$ approximation algorithms which can

compute a bounded-suboptimal solution in polynomial time. This reduction scales our framework to large batches of tasks without compromising on efficiency and solution quality.

We first briefly introduce the uncapacitated facility location (UFL) problem. Please refer to Williamson and Shmoys (2011, chapter 4) for a more detailed treatment. The facility location problem has a set of demands $D = \{1, \ldots, m\}$ and a set of facilities $F = \{1, \ldots, n\}$. There is *facility cost* f_i associated with opening each facility $i \in F$ and an *assignment* or *service cost* c_{ij} of serving demand j by facility i. The goal is to serve all the demands by opening a subset of facilities $F' \subseteq F$ such that the overall cost of opening the facilities in F' and the cost of assigning each demand $j \in D$ to the nearest facility $i \in F'$ is minimized:

$$\min \underbrace{\sum_{i \in F'} f_i}_{\text{facility cost}} + \underbrace{\sum_{j \in D} \min_{i \in F'} c_{ij}}_{\text{service cost}} \tag{3.3}$$

Theorem 3.4.1. *ADL is an instance of the uncapacitated facility location problem.*

Proof. Given an instance of ADL with m tasks, we construct an instance of UFL with m demands and $2m$ facilities as follows. Let M be a large number such that $M > c_{rob} + c_{fail} + c_{hum}$. Define the set of demands as

$$D = \{1, \ldots, m\}$$

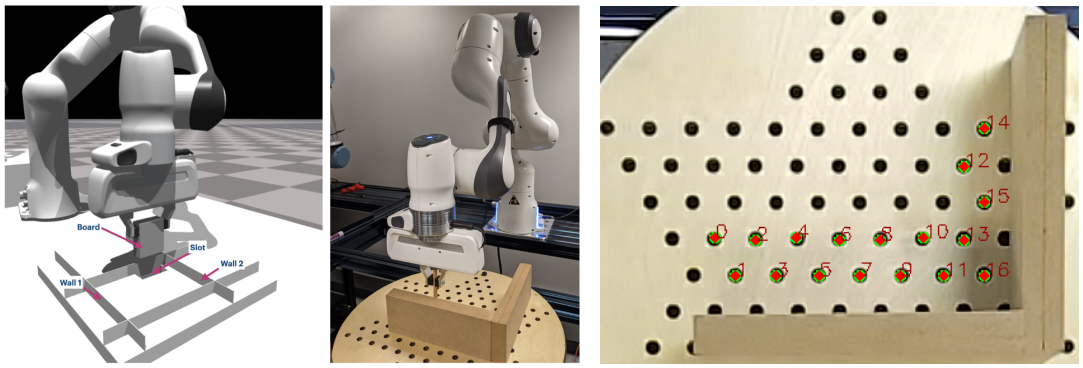
corresponding to all the tasks. For every task, define a delegate facility and a learn facility. The delegate facility corresponds to delegating that task while the learn facility corresponds to learning the task. Hence, the set of facilities is

$$F = F_{del} \cup F_{learn} = \{1_{del}, \ldots, m_{del}\} \cup \{1_{learn}, \ldots, m_{learn}\}$$

A delegate facility i_{del} has a facility cost of $f_i^{del} = c_{del}(i)$. The service cost is 0 for task i and M for all the other tasks. A learn facility i_{learn} has a facility cost of $f_i^{learn} = c_{learn}(i)$. The service cost is

$$c_{ij} = c_{rob} + (1 - \rho_i(\tau_j)) \cdot c_{fail}$$

for all subsequent tasks τ_j and M for all preceding tasks. This is necessary as the



(a) (b)

Figure 3.5: (a) Peg insertion under uncertainty in simulation and the real world (b) Shown here is an overhead image of the task setup. In our experiments, we focused on the holes (highlighted in red) close to the two wooden walls so that the robot could always use one of the walls to localize the peg with respect to the target hole. The task-id corresponding to each hole is written next to it.

tasks come in a fixed order.

The solution to this UFL instance is a subset $F' \subseteq F$ of facilities that should be opened to minimize the objective 3.3. A solution to ADL can be extracted as follows. If facility $i_{del} \in F'$, then task τ_i should be delegated. If $i_{learn} \in F'$, then the robot should be taught τ_i. Otherwise the robot should attempt the task. □

3.5 Experiments

We evaluate our approach, both in simulation and in the real world, on two challenging problems of insertion under uncertainty and Lego stacking. Our first objective is to understand the benefits of the ADL framework as compared to baselines that are myopic or reason about only a subset of the three options. Second, we evaluate our hypothesis that the precondition model can be transferred to a real robot after training in simulation. In both these experiments, we use a 2-layer fully connected neural network as our precondition prediction model and we are able to solve our mixed integer program optimally in well under a second using Gurobi (Gurobi Optimization, 2021).

Baselines

We compare our approach against three baselines:

1. **Act Delegate (AD)**. The robot chooses between acting and delegating based on the expected costs of these two actions.
2. **Confidence-Based Autonomy (CBA)** (Chernova and Veloso, 2009). Given a fixed threshold θ, the robot attempts a task if its confidence in success is greater than θ and asks for demonstrations to learn the task, otherwise.
3. **Act-Learn Myopic (ALM)**. Similar to the strategy used by (Rigter et al., 2020), the robot chooses between attempting a task and asking for human demos by comparing the immediate expected costs of both the actions.

Evaluation Metrics

The main evaluation metric is the total cost of completing all the given tasks. We also compare the methods based on the number of demonstrations and human interventions and the number of failures.

Skill Representation

In both our experiments, the robot end-effector is controlled using Cartesian-space impedance control which commands torques at the end-effector based on errors in the Cartesian space using a spring-damper system. A skill is a sequence of waypoints in the robot's end-effector frame, where each waypoint is defined by a 6D pose and the stiffness to be used in the corresponding spring-damper system.

3.5.1 Insertion under Uncertainty

We first look at insertion of blocks and pegs into a hole under uncertainty. Each task involves inserting a block of dimensions 1 cm x 1 cm x 6 cm into a slot of dimensions 1.2 cm x 1.2 cm x 2 cm in a known environment with a noisy estimate of the slot location $\sim \mathcal{N}(0, 0.3^2 cm^2)$. We generate four different environments of dimensions 20 cm x 20 cm each, with different numbers of walls arranged in a grid. We use the Nvidia Isaac Gym simulator (Nvidia, 2020) to simulate the tasks and to train the precondition model. Due to uncertainty in the position of the hole, the

robot needs to take uncertainty reducing actions for successful insertion. We visualize such a strategy in figure 3.6 which first localizes the wall next to the hole.

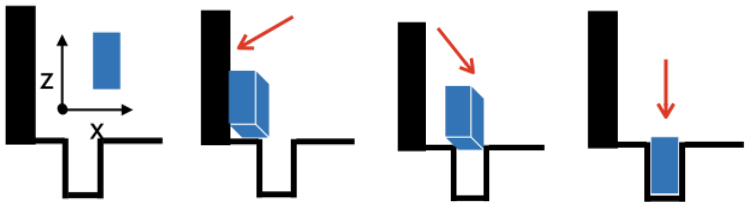


Figure 3.6: Shown here is a side-view of a 4-step block insertion strategy. The block is shown in blue, the wall next to the slot is shown in solid black and the red arrow points along the force vector applied on the board by the robot. The robot knows the location of the slot with respect to the wall and hence first pushes against it to reduce uncertainty in the x direction.

Precondition Prediction Model

We generate a set of 100 tasks from the same four environments for learning the precondition prediction model. For 30 tasks out of these, we learn block insertion policies using Relative Entropy Policy Search (REPS) (Peters et al., 2010). These 30 skills are evaluated on all the 100 tasks to generate success binary labels. Using this data we train a 2-layer fully connected neural network as the precondition prediction model. Given training and test task features τ_{train} and τ_{test}, we feed $\tau_{train} - \tau_{test}$ as input to the model to encode translational invariance.

Evaluation in Simulation

We compare ADL with AD, CBA($\theta = 0.5$) , CBA($\theta = 0.2$) and ALM in figure 3.7, where 0.2 is the optimal CBA threshold found using grid-search. ADL outperforms all the baselines at every level of pretraining. However, the improvement provided by ADL drops with increase in pretraining as the robot can complete more of the tasks autonomously without seeking additional demos or delegating. Also note that CBA outperforms AD at low levels of pretraining but the opposite holds at higher levels as demos sought by CBA are not cost-effective for the task set.

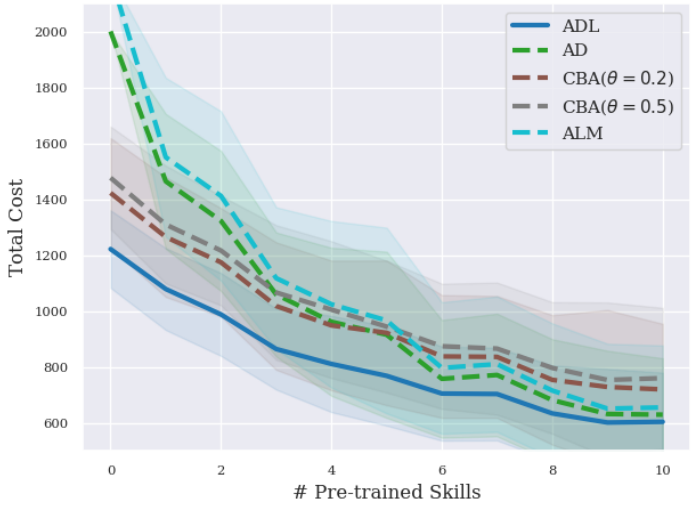


Figure 3.7: Comparison of ADL vs baselines in total cost for solving 20 block insertion tasks at different levels of skill pretraining. Pretraining is done by teaching the robot randomly sampled tasks from the task distribution. ADL is strictly better than all baselines at every level of pre-training. However, after pre-training with 8 skills, both ADL and AD converge to full autonomy as the robot is able to solve most of the tasks with pre-trained skills. We use $c_{rob} = 10$, $c_{hum} = c_{fail} = 100$ and $c_{demo} = 200$.

Evaluation on a Real Robot

We do a qualitative evaluation of our approach on peg insertion in the real world using the 7 DoF Franka Emika Panda arm. In our approximate model, we assume that there is always only one hole in the task area and each new task is a different set of walls and hole. In the real world we use a single Chinese checkers board as shown in figure 3.5. We attach two pieces of wood next to the holes to act as fixtures for localization. Each task here is fully specified by its location on the board. From a set of 17 tasks, we create 5 sets of 10 tasks each for evaluating our approach.

The results from our experiment are summarized in in table 3.1. In task set 1 (table 3.1), all the tasks are close to the bottom wall and hence can be solved using the same skill. The precondition prediction model is able to capture this relationship, which is why ADL asks for only 1 demo and doesn't delegate to a human. In task set 2 (table 3.1), tasks 12 and 14 lie close to the right wall and hence need a skill different from the one that can solve the other 8 tasks in that set. Because the precondition model is able to predict this (a) it does not ask the robot to attempt these two tasks and (b) it delegates them to the human as it is not cost-effective to seek additional

	Order of Tasks	#demo	#hum
1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	1	0
2	0, 1, 2, 12, 3, 4, 14, 5, 6, 7	1	2
3	1, 0, 2, 12, 3, 4, 14, 5, 6, 7	1	3
4	1, 3, 2, 12, 0, 4, 14, 5, 6, 7	1	4
5	1, 3, 5, 7, 0, 12, 4, 14, 2, 6	2	2

Table 3.1: Our parameterized preconditions model learned in simulation transfers to the real world as it is able to capture the relative distances of walls from the hole. We assume $c_{rob} = 10, c_{hum} = 100, c_{fail} = 100, c_{demo} = 300$.

demos for just two tasks.

3.5.2 Lego Stacking

In our second domain of Lego stacking we seek to evaluate how well our method works in the real world. In particular, we want to understand whether a precondition model learned using an abstract simulation is able to reduce effort in real world. Each task involves picking up a part made up of Lego bricks from a table and stacking it firmly onto a Lego base plate. A robot execution fails if two or more corners of the part are not locked onto the plate or the robot hits the base at any point. The robot is provided a bounding box around the part, a grasp location and a target location by the user. We use a 66D feature vector for each task– binarized and resized image (to 8×8) along with its original size. Before running the experiments, we record 5 demos for each task in the ground set. Every time the robot requests a demo for a task, one of the 5 pre-recorded demos is provided by sampling randomly.

Skills

Each stacking skill consists of three sub-skills executed in sequence: pickup, place-and-wiggle and robust-tapping.

1. **Pickup.** Picks up the part given a grasp location.
2. **Place-and wiggle.** Places the part at the target location and pushes it down while perturbing the part randomly to align it with the studs on the base plate. For all parts, except the smallest ones, this step alone is not sufficient for firm stacking as it only ensures stacking within a small region around the grasp.

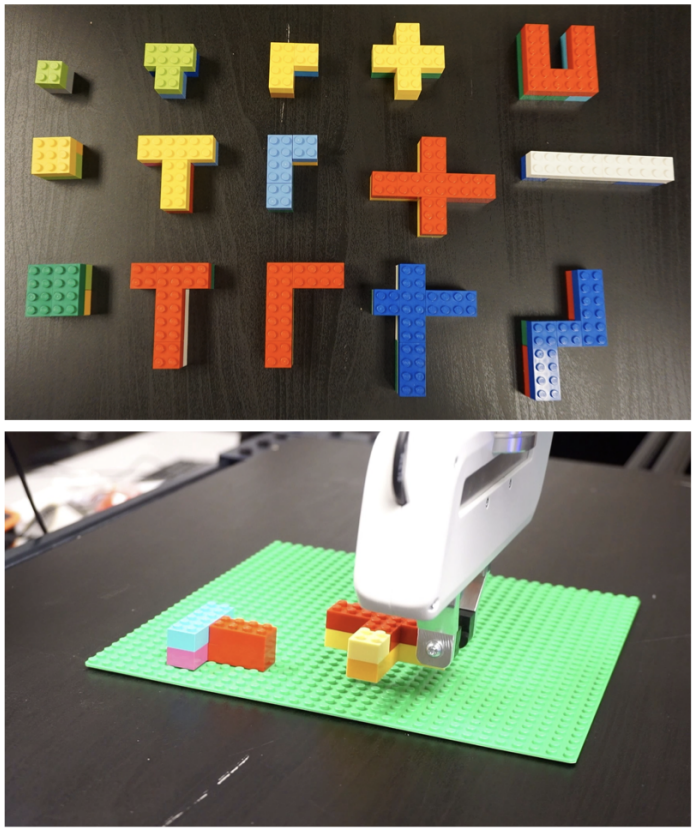


Figure 3.8: (*Top*) The ground set of 15 tasks from which test sets of 10 tasks each are sampled uniformly randomly. (*Bottom*) The Franka-Emika Panda robot stacking one of the parts onto the base plate.

3. **Robust-tapping.** Pushes down on the part at different locations to make sure that the part is stacked firmly.

The first two are hand-designed and common across all tasks, while robust-tapping needs to adapt the number and location of taps based on the geometry of the part. The latter is learned in the grasp-frame and scaled based on the size of the part. This allows the skill to generalize to different locations and across parts of similar shape but different sizes.

Precondition Prediction Model

Physics-based simulators struggle to simulate interactions among multiple Lego bricks and the interference fit mechanism used in them. Consequently, we use a custom simulation based on our observation that the primary reason for variability in

skills is the geometry of the parts. We can afford to ignore physics and robot dynamics as we do not transfer the learned skills to the real world. Our coverage-based simulation takes in a 2D image of a part and identifies only the number and location of taps needed to cover the whole part by randomly sampling points on the image. Experimentally, we found that a single tapping action has an effect upto about 3cm from the tapping location. We use this knowledge in the simulation to determine whether a part is covered or not after a sequence of taps. We capture 10 images of each of the 15 tasks, along with a bounding box around the part and the grasp location. After training skills for each of the resulting 150 tasks in our coverage-based simulation, we evaluate them on all the tasks to generate binary success labels Using this data we train a 2-layer fully connected neural network as the precondition prediction model. Given training and test task features τ_{train} and τ_{test}, we feed $[\tau_{train}, \tau_{test}]$ as input to the model.

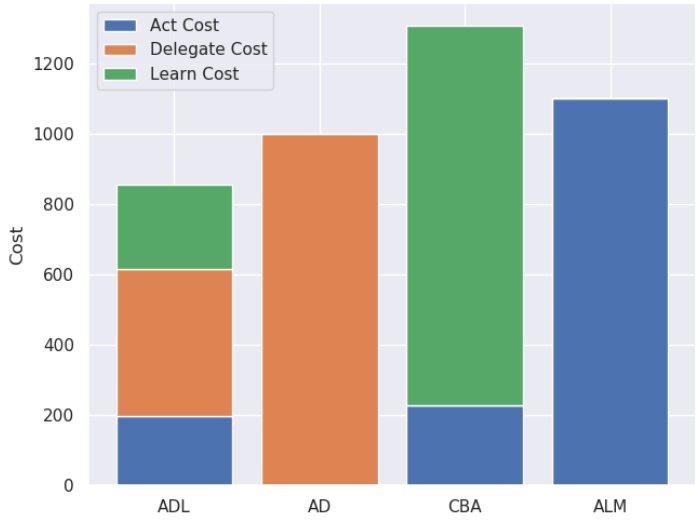


Figure 3.9: Comparison of ADL vs baselines in the Lego stacking domain using $c_{rob} = 10, c_{hum} = c_{fail} = 100$ and $c_{demo} = 200$. ADL is the only planner that leverages synergy among acting, delegating and learning to complete tasks at minimum cost.

Evaluation on a Real Robot

We evaluate all the approaches on 10 sets of 10 Lego-stacking tasks each using $c_{rob} = 10, c_{hum} = c_{fail} = 100$ and $c_{demo} = 200$. We choose $c_{demo} > c_{hum}$ as it takes much more time to provide a demo than for the human to stack the Lego themself,

while $c_{hum} = c_{fail}$ as a failed robot execution can be fixed quickly by a human. c_{rob} is the smallest cost as we value human time much more than robot time in this domain. Figure 3.9 shows the total cost of completing all tasks using each of the methods. AD delegates all tasks as the skill library is empty at the beginning, CBA asks for too many human demos as it doesn't take into account their relevance to the task set and ALM doesn't ask for any demos as its upfront cost is higher than failing at a task. In contrast, ADL finds the optimal synergy among all the three options to solve the tasks with minimum cost. The full results are summarized in table 5.1.

	Total Cost	# demo	# hum	# fail
ADL	**856** (± 120.9)	1.2 (± 0.4)	4.2 (± 1.7)	1.5 (± 1)
AD	1000 (± 0)	0 (± 0)	10 (± 0)	0 (± 0)
CBA	1306 (± 207.2)	5.4 (± 1.2)	0 (± 0)	1.8 (± 1.5)
ALM	1100 (± 0)	0 (± 0)	0 (± 0)	10 (± 0)

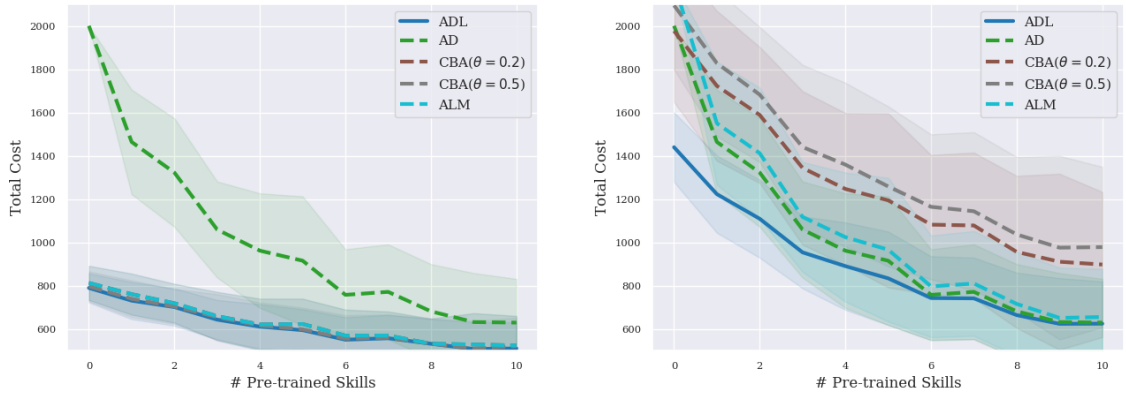
Table 3.2: We report the mean and standard deviation of the results averaged over 10 different planning problems with 10 tasks each. Baselines ALM, AD and CBA($\theta = 0.5$) cost 28.5%, 16.8% and 52.6% more than ADL as they don't plan ahead. We use $c_{rob} = 10$, $c_{hum} = 100$, $c_{fail} = 100$, $c_{demo} = 200$ and no skill pretraining.

Effect of Costs on Plans

To better understand the effect of costs on ADL, we compare ADL with baselines in two settings (see figure 3.10):

1. **Low cost of teaching**, *i.e.*, $c_{demo} \leq c_{hum} < c_{rob} + c_{fail}$. When it is easier to teach a new skill to the robot than to assign it to a human, the cost optimal strategy is independent of future tasks and is to simply ask for demos if the robot is not confident. Unsurprisingly, both CBA and ALM perform as well as ADL in this setting as planning is not required.

2. **High cost of teaching**, *i.e.*, $c_{hum} < c_{rob} + c_{fail} < c_{demo}$. Teaching a robot how to do a new task is typically harder than assigning it to a human. In this setting, it is important to plan ahead to minimize the number of demos since the cost-optimal strategy depends on future tasks. Experimentally we find that ADL does indeed perform substantially better than baselines.



(a) $c_{demo} = 90 \leq c_{hum} < c_{rob} + c_{fail}$ (b) $c_{hum} < c_{rob} + c_{fail} \leq c_{demo} = 300$

Figure 3.10: We compare ADL with baselines with different amounts of pre-training at the start of deployment and two different costs of learning. (a) The cost-optimal strategy is to ask for demos if the robot is not confident. Both CBA and ALM perform as well as ADL in this setting. (b) It is important to plan ahead to be able to minimize the number of demos. ADL performs substantially better than baselines in this setting. $c_{rob} = 10$ and $c_{hum} = c_{fail} = 100$ in both the settings.

3.6 Discussion

In this chapter, we proposed a P2L framework for collaborative robots that operate as part of a human-robot team. Planners for task allocation in human-robot teams have traditionally assumed robots possess a fixed set of capabilities. We argued that this is a missed opportunity as collaborative robots can be taught new skills on the job by their human teammate through demonstrations. However, this additional teaching has to be done strategically to avoid over-burdening the human worker with queries for demonstrations. Motivated by the assembly domain, we considered the setting of fulfilling a known sequence of n tasks. In such settings, the cost of teaching a robot via demonstrations is typically much higher than the cost of a human taking over the task. However, it is beneficial to teach a new skill to the robot if the time spent by the human on teaching is offset by the time saved by allocating additional tasks to the robot in the future. This leads to a number of questions, such as, which additional skills should be taught to the robot and how should tasks be allocated?

Our proposed planning framework called Act, Delegate or Learn (ADL) answers these questions cost-optimally. ADL has two key components: (1) a *general* mixed

integer programming formulation and (2) a learned *domain-dependent* precondition prediction model to predict the benefits of learning a new skill. Simulated and real world evaluations on two challenging manipulation domains indicate that our approach saves significant human and robot effort compared with approaches that do not plan ahead.

3.6.1 Limitations

A major limitation of our approach is that it assumes knowledge about the effort involved in teaching a robot and doing tasks in the form of costs. It can be challenging to acquire these costs accurately as they are highly situational and may depend on human preferences. For example, kinesthetic teaching is more challenging with heavy robot arms than with lighter arms. Estimating costs that are aligned human preferences is hence a promising direction for future work. That said, our approach can be useful even when the costs are not precisely known as long as they capture the relative difficulty of the different actions.

Second, we focus on a setting in which the order of all future tasks is known. In some settings it may be possible to re-order the tasks. This can allow the robot to learn all the important skills early on so that it can apply them to the remaining tasks. We believe our mixed integer programming approach can be extended to such settings. We also currently do not account for uncertainty in future tasks which is likely to be present in many domains.

Another limitation of our approach is that it does not reason about sequencing multiple skills to solve multistep tasks. This limits ADL to relatively short horizon tasks that can be completed using a single skill. For our approach to be more broadly applicable, it would need to decide not only which tasks the robot should learn but also which skill in the sequence to seek demonstrations for. A skill that is reused across multiple tasks is evidently more important than a task-specific skill. We will consider active skill learning for such multistep tasks in the next chapter.

4 P2L for Multistep Tasks: Metareasoning for Skill Learning

In this chapter[1], we move away from human-robot collaboration to focus on multistep tasks that require sequencing multiple skills. We focus on skill learning to improve a robot's robustness to failures due to noise in state estimation and actuation. For example, a robot may miss the handle or drop the key while trying to open a door due to an incorrect handle pose estimate. In practice, such mistakes are often handled with hand-engineered or heuristic behaviors and state machines. While practical for relatively simple tasks in controlled environments, this approach cannot scale to systems deployed in the real-world which can fail in a variety of different ways. Hence, there is a need for an algorithmic way to (1) discover potential failures and (2) quickly improve the robot when new failures are discovered.

To this end, we propose an approach to incrementally improve a robot's robustness by discovering potential failures in simulation and learning *recovery skills* that allow the robot to recover. While our approach can robustify against failures due to uncertainty in both execution and state estimation, we focus on the latter as it is more challenging. We assume the robot is given a nominal set of policies which can complete the task under ideal conditions. These could be hand-designed controllers or policies learned from human demonstrations. In reality, the state is rarely known perfectly but is estimated using an online state estimation module. Consequently, the robot may make mistakes during execution and enter a state from which it cannot continue— a *failure state*. We discover such failures in simulation by executing the nominal policies under a simulated state estimation model. Next, we cluster similar

[1]This chapter is based on material from Vats et al. (2023).

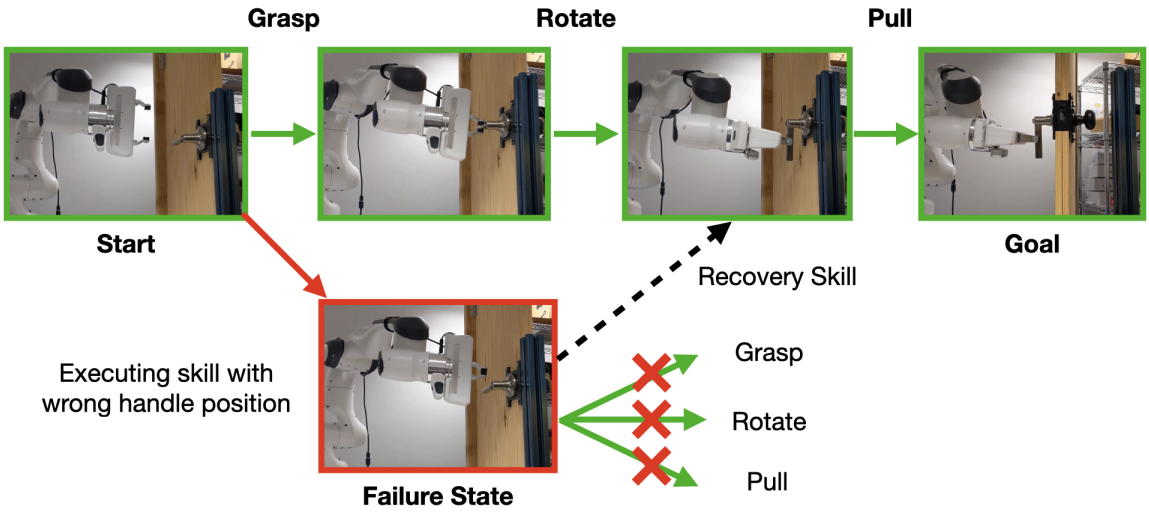


Figure 4.1: To open a door, the robot has to (1) grasp the handle (2) rotate it and (3) pull the door. However, it can fail during any of these three stages due to incorrect state information and erroneously enter a failure state from which none of its skills can be applied. We propose an approach that (1) discovers such failures in simulation and (2) uses meta-reasoning to efficiently learn recoveries to the preconditions of the robot's existing skills.

failures and learn recovery skills for each of the clusters that allow the robot to recover to the *precondition* of one of its nominal policies.

There are multiple potential recoveries from every failure cluster, each corresponding to a precondition the robot could recover to. For n failure clusters and m preconditions, this results in a total of $n \times m$ potential recovery skills. Since attempting to learn all of these recoveries is computationally expensive and redundant, prior works use heuristics to choose where to recover. Recoveries generated in such a way can be sub-optimal as these heuristics don't reason about the quality of the recovery. For example, a common heuristic is to recover to a previous state upon detecting a failure. However, it is preferable to recover closer to the goal in terms of execution cost. On the other hand, it is not known *a priori* if recovering close to the goal is feasible. To this end, we propose **Meta-Re**asoning for **Skil**l **L**earning (MetaReSkill), an algorithm that builds a predictive model of improvement in skill performance and decides online which skills to devote training resources to such that the overall task performance improves maximally.

4.1 Related Work and Background

Recovery from Failures

Robotic systems are usually deployed with hand-designed recovery behaviors in the anticipation of failures. Common recovery strategies include retrying the previous step (Ebert et al., 2018; Matsuoka et al., 1999), backtracking (Wang and Kroemer, 2019) and hand-designed corrective actions (Hsiao et al., 2010; Sundaresan et al., 2021). To execute a recovery, it is important to first detect (Rodriguez et al., 2010; Park et al., 2016; Luo et al., 2021; Zachares et al., 2021) what kind of failure has happened or is about to happen. Pastor et al. (2012) propose Associative Skill Memories which associate stereotypical sensory events with robot movements. Parashar and Goel (2021) propose an architecture for robot assembly which uses meta-reasoning to identify the cause of a failure and repair the knowledge that caused the failure. In all these works, the recovery behaviors are either manually designed, which limits their scalability, or are generated using a heuristic, which limits their complexity and quality. By contrast, we learn recovery behaviors with reinforcement learning which offers the possibility of learning complex recoveries. Pacheck et al. (2019) encode the robot's capabilities in linear temporal logic (LTL) which allows them to suggest additional skills that would make an infeasible task feasible. Similarly, Niekum et al. (2015) learn a finite-state task representation from demonstrations and improve it using corrective demonstrations. While we too leverage an abstract state representation, our focus is on actively choosing which skill to learn. If human help is available during deployment, then robots can use inverse semantics (Knepper et al., 2015) to seek help through language requests such as "Please give me the white table leg that is on the black table". However, we do not assume help during deployment. More recently, there has been interest in learning a policy to recover to a safe state (Thananjeyan et al., 2021; Wilcox et al., 2022). However, these works learn a single recovery policy, assume full observability and do not focus on the efficiency of learning.

Skill Chaining

Skill chaining (Konidaris and Barto, 2009b; Bagaria and Konidaris, 2019) is a popular approach to solve long horizon RL problems by decomposing them into shorter sub-problems. A sequence of options is learned backwards from the goal, such that

in each iteration a new option is learned to reach the precondition of the previously learned options. The precondition is usually learned using binary classification and describes the states from which the option policy can be successfully executed. During the *initiation period* of an option it is executed from different states in the environment to collect data for training the precondition. The precondition classifier is then frozen and used to generate success or failure rewards for the option that is trying to reach it.

Acting under Uncertainty

The problem of acting under partial state information is optimally solved by formulating it as a Partially Observable Markov Decision Process (POMDP) (Åström, 1965; Kaelbling et al., 1998). A POMDP is defined by the tuple $\langle S, A, T, R, \Omega, O, \gamma \rangle$, where S is the underlying state space and Ω is the observation space. In this formulation, the robot acts on its state belief b, which is a probability distribution over all possible current states. However, solving a POMDP exactly is intractable in manipulation (Kaelbling and Lozano-Pérez, 2013). Instead, we use a common heuristic technique of using a state estimator to maintain a belief over world states based on observations and actions, while the robot acts on the most likely state (Spaan, 2012).

4.2 Problem Setup

We are interested in solving a manipulation task defined by a distribution of start states $\mathcal{D}$ and a goal indicator function $f_{goal} : \mathcal{S} \rightarrow \{0, 1\}$. The robot incurs costs based on its actions and a penalty of c_{fail} if it ends up in a dead-end or is unable to complete the task in T timesteps. We are given a set of nominal control policies $\{\pi_1, \cdots, \pi_k\}$ and a high level policy Π which chooses among these control policies. We assume that they can reliably complete the task under no uncertainty. Our goal is to improve the robustness of the robot by discovering and learning additional recovery skills that can handle failures due to state uncertainty. Formally, we seek to maximize the expected return of the high-level policy on the task distribution:

$$\mathbb{E}_{\tau \sim \mathcal{D}} \sum_{\substack{t=0, s_0=\tau \\ a_t \sim \pi(s_t), \pi \sim \Pi(s_t)}}^{T} R(s_t, a_t) \tag{4.1}$$

where $R(s,a) = -c(s,a) - c_{fail}1_{failure}$ and s_t is the most likely state at time t as determined by a state estimator.

Recovery Skill

A recovery skill is an option whose goal is to bring the system to a state from which its nominal control policies can take over. Formally, let $\Pi = \{\pi_1, \cdots, \pi_k\}$ be the nominal set of control policies of the robot and let $\{\rho_1, \cdots, \rho_k\}$ be their preconditions. We say that the robot has reached a failure state if none of the preconditions ρ_i are satisfied. A recovery skill (figure 4.1) drives the robot to a safe state where at least one of the preconditions is satisfied so that the robot can complete the task.

4.2.1 Symbolic Skill Graph

Instead of reasoning with low-level ground states, which are high dimensional, we build a compact symbolic skill graph $G = (V, E)$. Each vertex in this graph is a symbolic state corresponding to a set of continuous states defined by its precondition $\rho : \mathcal{S} \to \{0, 1\}$. There exists an edge between two vertices $u, v \in V$ if there is a skill whose precondition contains u and its effect is contained in v. We initialize this graph as a chain with vertices $V = \{\rho_1, \cdots, \rho_k, \rho_{goal}\}$ corresponding to the preconditions of the nominal skills and edges E corresponding to the nominal policies. Let π_{ij} be the skill from ρ_i to ρ_j and q_{ij} be its probability of success. If π_{ij} fails then we assume it ends up in an absorbing failure state $Fail$ incurring a penalty of c_{fail}. π_{ij} can be learnt using off-the-shelf RL algorithms where ρ_i is the initial state distribution and ρ_j is the goal condition. While we could use ρ_j to define a binary reward function for RL, this is usually impractical for high-dimensional domains. Fortunately, ρ_j can also be used to define a dense reward, for example, by computing the distance to the decision boundary or using the probability $\rho_j(s)$ as the reward. Finally, the high level policy Π for choosing which skill to execute at every symbolic state can be computed using Value Iteration as the skill graph is discrete.

Precondition Chaining

The preconditions of the nominal skills can either be hand-designed or learnt. We use precondition chaining (described in alg. 2) to learn the preconditions backwards

Algorithm 2 Learning the preconditions of nominal skills via precondition chaining.

procedure PRECONDITIONCHAINING($\{\pi_1^0, \cdots, \pi_k^0\}$)
  Execute skill chain and collect N successful trajectories of size $k+1$
  $\mathcal{D}_{1:k}^+ \leftarrow$ Learn positive distribution of the start of every skill
  goal $\leftarrow f_{goal}$ ▷ Overall task goal
  for $i \in \{k, k-1, \cdots, 1\}$ **do** ▷ backwards from goal
    $X \leftarrow \phi, Y \leftarrow \phi$
    for $1 \leq j \leq M$ **do**
      Sample τ from $\mathcal{D}_i^+$
      Set state to τ and execute π_i^0
      y $\leftarrow 1$ if goal is satisfied, 0 otherwise
      $X = X \cup \tau$, $Y = Y \cup y$
    $\rho_i \leftarrow$ Train classifier using (X, y)
    goal $\leftarrow \rho_i$
  return $\{\rho_1, \cdots, \rho_k\}$

from the goal. The main difference from skill chaining (Konidaris and Barto, 2009b; Konidaris et al., 2012) is that we are not interested in discovering new skills at this point but only in learning the preconditions of the given skills. This involves two steps:

1. We collect successful trajectories by executing the nominal policies in simulation. Let $\{s_1, \cdots, s_k, s_{goal}\}$ be one such trajectory consisting of only the start and end states of each policy and s_{goal} is a state that satisfies the goal function. For every policy π_i, we learn a corresponding positive distribution $\mathcal{D}_i^+$ over its start states S_i^+.

2. We train the precondition classifiers backwards from the goal. To learn the precondition ρ_i, we sample states in the vicinity of $\mathcal{D}_i^+$ and execute π_i from there. We verify its success using ρ_{i+1} (ρ_{goal} for ρ_k). This helps us gather informative negative samples S_i^- and additional positive samples which are crucial for learning a tight decision boundary. The precondition classifier ρ_i is trained using S_i^+ and S_i^-.

4.3 Approach

Our approach consists of two steps: failure discovery and learning recovery skills using meta-reasoning.

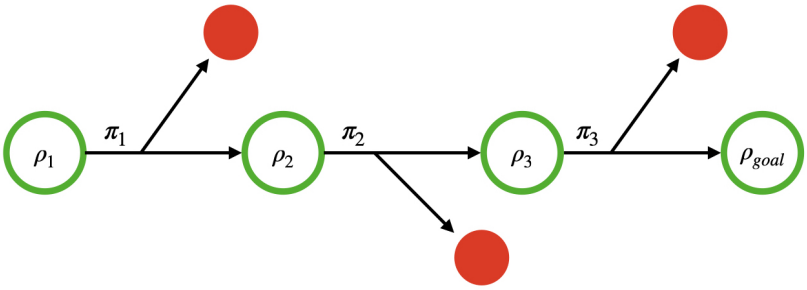


Figure 4.2: **Failure Discovery.** We execute the nominal skills under a simulated state estimator to induce failures (shown in red). These failure states $\in S$ are clustered into failure modes using a Gaussian mixture model (GMM). This GMM is used as a failure classifier during execution.

4.3.1 Failure Discovery

We procedurally generate failure states in simulation by executing the nominal skills under noisy state information. Concretely, let s be the true current state and o be a noisy observation. While the true state is known to us in simulation, we provide only the observation to the nominal skills. Because of the mismatch between o and s, the skill may not work as intended and the robot may end up in a new state s' with observation o'. If none of the existing skills is applicable at s', we record s' as a failure state as the robot would not be able to recover from it even if it could observe the true state. Note that we do not record o' as it may not even be a valid world state. While a recovery for s' does not allow the robot to deal with its current observation o', it will be useful when the robot observes $o \approx s'$. We propose two failure discovery strategies:

1. **Pessimistic Discovery.** The robot executes its nominal policies open-loop under simulated high state uncertainty. This strategy discovers a larger and more diverse set of failures than what may actually be encountered during execution. While this makes recovery learning computationally more expensive, it doesn't require a model of the state estimator.

2. **Early Termination.** The robot executes its nominal policies using observations from a simulated state estimator and terminates if none of the preconditions are satisfied. This strategy discovers a more accurate failure distribution and is preferable if a model of the state estimator is available.

Let S_{fail} be the set of failures discovered. We cluster S_{fail} into n failure clusters $\{\rho_1^f, \cdots, \rho_n^f\}$ and add them as states to our symbolic skill graph. For failures discovered using the early termination strategy, the size of a failure cluster corresponds to the likelihood that the robot will end up in that failure. Both of these failure discovery strategies lead to recoveries that provide significant improvement in performance over heuristic recovery strategies in our experiments.

4.3.2 Meta-Reasoning for Skill Learning (MetaReSkill)

The robot may recover to one of the $k+1$ preconditions $\{\rho_1, \cdots, \rho_k, \rho_{goal}\}$ from every failure cluster. However, many of these recoveries are redundant or infeasible. Instead of trying to learn all of them, our algorithm identifies and prioritizes the most promising recoveries.

4.3.3 Objective

We define the Value of Failures to measure the performance of the robot at its failure states. Consider a failure cluster ρ_i^f in figure 4.3 at the start of recovery learning. ρ_i^f is not connected to any precondition as the success probability q_{ij} of all the recoveries is 0. Hence, the value of this failure cluster is $V(\rho_i^f) = -c_{fail}$. With further training, the value improves to

$$\max_j q_{ij} V(\rho_j) - (1 - q_{ij}) c_{fail}$$

ρ_j's are high value states as nominal skills can be executed reliably from there. To take into account multiple failure modes, we take a weighted sum of the values of all the failure clusters:

$$VoF = \sum_i \frac{|\rho_i^f|}{\sum_j |\rho_j^f|} V(\rho_i^f) \tag{4.2}$$

where $|\rho_i^f|$ is the number of ground states in the cluster ρ_i^f.

Intuitively, a high VoF implies that failures during execution are less problematic as the robot is highly confident of recovering. Learning recoveries that optimize the VoF improves robustness to failures. A good first meta-strategy is to train all the recoveries in a round-robin manner as we do not know *a priori* what the best recovery

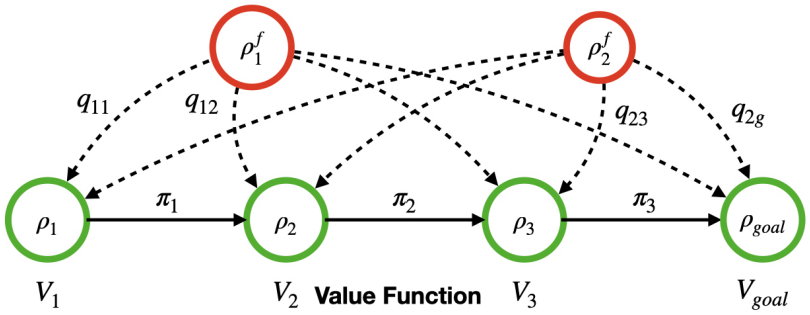


Figure 4.3: **Optimistic Recovery Learning.** We learn recoveries using the most likely state heuristic, i.e., we optimistically assume the state becomes fully observable after the robot ends up in a failure state. We can potentially learn a recovery (shown in dotted lines) to each of the preconditions. At the start of recovery learning, none of these recoveries have been learnt and the robot always incurs a penalty c_{fail} for failing. After some training, the recoveries are partially learnt and have success probabilities q_{ij}. Note that it is preferable to recover to preconditions closer to the goal as they have a higher value than those away from the goal.

from every failure mode will be. While this works well in the initial stages of learning, we observed in our experiments that it is quite inefficient as the VoF quickly saturates (figure 4.6). To address this issue, we propose a meta-reasoning algorithm that tracks the progress of all the recoveries and chooses which failure modes to focus on and which precondition to recover to such that the VoF improves maximally with high probability in every training episode.

4.3.4 Model of Task Performance

The key idea of MetaReSkill is to build a model of improvement in task performance by estimating the *rate of improvement* (ROI) of individual skills. We use confidence interval estimation to compute optimistic upper bounds Δq_{ij}^U of the ROI Δq_{ij} of the success probabilities q_{ij} of all the recoveries. As shown in figure 4.4, this provides an optimistic estimate of how much a recovery could improve after another round of training.

Let θ be a parameter we wish to estimate. An α-confidence interval (Hines et al., 2008) for θ is an interval (L, U) such that θ is contained in the interval with confidence α. This also implies that U is an upper bound on θ at least with confidence α. Let

Algorithm 3 Meta-Reasoning for Skill Learning

procedure TRAIN
  $Q_{ij} \leftarrow$ queue of max size $w, \forall i, j$
  for $1 \leq t \leq B$ **do**
    if all policies have been trained $\geq K$ times **then**
      for all i, j **do**
        $T \leftarrow$ transition matrix of skill graph
        $T(i, j) \leftarrow q_{ij}^U$
        $J_{ij}^U \leftarrow$ OBJECTIVE(T, R)
      $(i^*, j^*) \leftarrow$ argmax J^U
    else
      $(i^*, j^*) \leftarrow$ least trained policy
    train recovery (i^*, j^*) for η episodes
    $q \leftarrow$ estimate new success rate of $\pi_{i^*j^*}$
    $q_{best} \leftarrow \max(q, q_{i^*j^*})$
    $q_{i^*j^*} \leftarrow q_{best}$, Q_{ij}.insert(q_{best})
    $q_{i^*j^*}^U \leftarrow$ COMPUTEUCL(i^*, j^*)
  return $\pi_{ij}, \forall i, j$
procedure OBJECTIVE(T, R)
  $V \leftarrow$ VALUEITERATION(T, R)
  return $\sum_i \frac{|\rho_i^f|}{\sum_j |\rho_j^f|} V(\rho_i^f)$
procedure COMPUTEUCL(i, j)
  $\Delta Q_{ij} \leftarrow$ compute forward differences of Q_{ij}
  $n \leftarrow |\Delta Q_{ij}|$
  $\Delta q_{ij}^U \leftarrow \overline{\Delta Q_{ij}} + t_{(1-\alpha)/2, n-1} \frac{s}{\sqrt{n}}$ ▷ UCL of ROI
  $q_{ij}^U \leftarrow q_{ij} + \Delta q_{ij}^U$ **return** q_{ij}^U

$\theta_1, \cdots, \theta_w$ be a random sample of the parameter. Under the assumption that the underlying population is normally distributed, the mean μ of the distribution lies in the following interval with probability α:

$$\overline{\theta} - t_{(1-\alpha)/2, w-1} \frac{s}{\sqrt{w}} \leq \mu \leq \overline{\theta} + t_{(1-\alpha)/2, w-1} \frac{s}{\sqrt{w}}$$

where $t_{\alpha,w}$ is the upper α percentage point of the student's t-distribution with w degrees of freedom and s is the standard error.

We compute the upper confidence limit (UCL) of Δq_{ij} using only the w most recent forward differences of q_{ij} as the rate of improvement is a non-stationary quantity (w

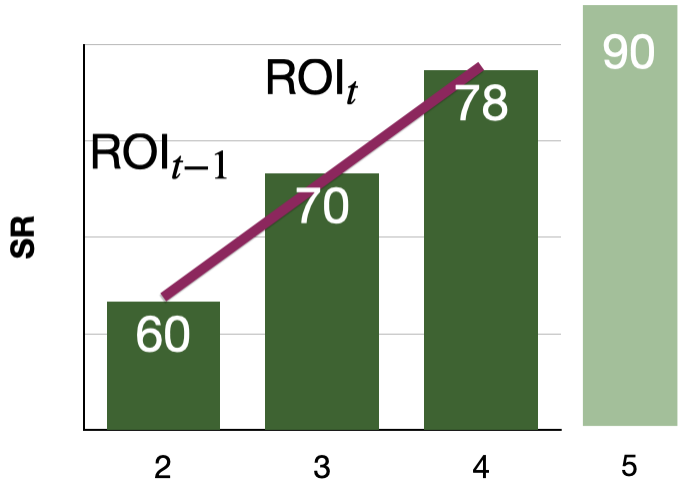


Figure 4.4: Using a skill's past rate of improvement in training rounds 2-4, we predict its success rate (SR) in the future round 5.

is a domain-dependent hyper-parameter). For every recovery with current success probability q_{ij}, $q^U_{ij} = q_{ij} + \Delta q^U_{ij}$ is an optimistic upper bound on its success probability after an additional round of training. Let J^U_{ij} be the VoF computed by replacing q_{ij} with q^U_{ij} in the transition matrix of the recovery learning graph 4.3. J^U_{ij} is then an optimistic prediction of the VoF if we were to train π_{ij} for another round. Our algorithm greedily picks a recovery for training that promises the highest VoF in the next round. We initialize MetaReSkill with K rounds of round-robin to estimate the UCL. Priors on Δq_{ij}, if available, can further speed up learning.

4.4 Experiments

We evaluate our approach on the task of door opening under noisy handle position information both in simulation and in the real world. The goal is to open a door by at least 0.3 rad with the Franka Panda robot under high initial state uncertainty.

Simulation Environment

We adapt the door environment from the MuJoCo-based robosuite (Todorov et al., 2012; Zhu et al., 2020) framework to match our real door. The world state is 18 dimensional and includes the robot's joint angles and poses of the door and the

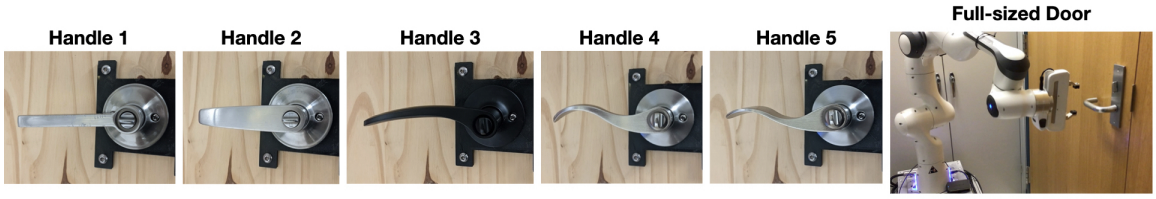


Figure 4.5: **Handles used in evaluation.** We compare our approach with open-loop execution on 5 different lever latch handles and a full-sized door with a real robot. Only handle 1 and a small door was used during training; handles 2-5 and the full-sized door are unseen.

	Success Rate (%)	Cost (m)
Recovery-skills (Ours)	**92.4**	0.95 ($\pm$ 0.34)
Retry	66.9	0.80 ($\pm$ 0.02)
Recover-to-prev	75.5	0.86 ($\pm$ 0.19)
Recover-to-start	73.6	0.87 ($\pm$ 0.26)
No-recovery	64.4	0.80 ($\pm$ 0.02)
Open-loop	71.0	0.80 ($\pm$ 0.02)

Table 4.1: **Simulation results.** We compare recovery skills trained with our approach using 150 REPS queries with heuristic recovery strategies. Our approach significantly improves the success rate. The statistics are averaged over 5 sets of recovery skills learnt with different seeds, each evaluated 200 times using a simulated state estimator and a limit of 10 skills per evaluation.

handle. The initial state uncertainty is sampled from $\mathcal{N}(0, \sigma = 2cm)$ each in the x, y and z positions of the handle. We design 3 open-loop skills to be executed in sequence - ReachAndGraspHandle, RotateHandle and PullHandle- as the nominal skills. Each skill consists of one or more 7D waypoints that the robot tries to reach using task space impedance control, where each target consists of a gripper open/close state and a 6D end-effector pose. These skills are able reliably to open doors in simulation and the real world if accurate state information is available.

Symbolic Skill Graph

We train the preconditions of the nominal skills by precondition chaining using a total of 1223 positive and negative samples. Each precondition is a generative classifier with the positive distribution $\mathcal{D}^+$ learnt as a Gaussian distribution and the

negative distribution $\mathcal{D}^-$ learnt as a Gaussian Mixture Model. The 3 nominal skills result in 4 symbols for the start, subgoals, and the goal.

Recovery Skill

Each recovery skill π_{ij} is a parameterized skill (da Silva et al., 2012) that uses a regression model to predict robot actions $\theta \in \Theta$ based on the start state s. We use k-nearest neighbours regression to predict a 21D vector, a sequence of three 6D poses with respect to the initial end-effector pose and gripper open/close states, as robot actions. Collecting data of the form (s, θ) for training this regression model involves sampling a start state s from the failure mode ρ_i^f and computing the robot action parameters θ for recovery to ρ_j using Relative Entropy Policy Search (REPS) (Peters et al., 2010). For learning a recovery to precondition ρ, REPS uses the reward function $R(s) = 0.1 \log f_{\mathcal{D}^+}(s) + 10\rho(s)$, where $f_{\mathcal{D}^+}$ is the probability density function of the corresponding $\mathcal{D}^+$. Each REPS query takes about 2 minutes to solve on our Intel® Core™ i7-9700K CPU.

4.4.1 Evaluation of Learnt Recovery Skills

We first evaluate the effectiveness of our overall approach using the pessimistic failure discovery strategy we described earlier. We execute the nominal skills 1000 times for failure discovery to collect a total of 1400 failure states which we group into 6 clusters using the Gaussian Mixture Model (GMM) (Pedregosa et al., 2011). Common failure modes include the robot missing the handle and the robot slipping while pulling the handle due to an improper grasp. We learn recovery skills in simulation using a budget of just 150 REPS queries. With 24 potential recovery skills, this means that each recovery policy can get only 6 data-points on average.

Evaluation in Simulation

We simulate a state estimator by assuming that the standard deviation of the noise distribution halves after every robot action. As we show in table 5.1, learnt recoveries are significantly better than heuristic recovery strategies in improving success rate. Recovering from failures during door opening often requires the robot to (1) carefully move the handle so as not to weaken the grasp and (2) avoid collision with the environment. Heuristic recoveries are unable to account for this and hence perform

poorly. Compared to open-loop execution, our approach substantially improves task success rate from 71% to 92.4%. This indicates that (1) the failures discovered using our pessimistic failure discovery do cover a number of failures encountered by the robot when using a state estimator and (2) recoveries can be reliably learnt with a generic reward function defined using preconditions which promises better scalability than reward shaping.

	Open-loop (%)	Recovery (%)
Handle 1 (Train)	75 (15/20)	**90** (18/20)
Handle 2 (Test)	50 (5/10)	**80** (8/10)
Handle 3 (Test)	80 (8/10)	**80** (8/10)
Handle 4 (Test)	80 (8/10)	**90** (9/10)
Handle 5 (Test)	60 (6/10)	**90** (9/10)
Full-sized door (Test)	30 (3/10)	**50** (5/10)

Table 4.2: **Success Rate on a Real Robot.** Learnt recovery skills significantly outperform open-loop execution on a real robot across 5 different handles. Open-loop fails almost half the time on handles 2 and 5 which are, respectively, the smallest and the thinnest of the 5 handles. By contrast, the learnt recoveries use a caging grasp to re-grasp the handle close to the handle's axis of rotation and are robust to these variations. We also test recovery skills on a full sized door where the success rate of our approach drops to 50% due to the slip-prone cylindrical handle of the door.

Evaluation on a Real Robot

We transfer the preconditions, failure classifier, nominal skills and recovery skills learnt in simulation to a real Franka Panda robot and run experiments with (a) 5 different lever latch handles on a small door and (b) a full-sized door in our building with a cylindrical handle (figure 4.5). We fine-tune only the gains of the impedance controller on the real robot by increasing their values to achieve similar tracking as in simulation. As in simulation, the robot is controlled by a Cartesian-space impedance controller that executes each skill open-loop. We evaluate the recovery skills learnt in simulation under an *idealized state estimator*. The ground-truth handle position is known to us but at $T = 0$, we only provide a noisy handle position estimate to the robot, where, noise $\sim \mathcal{N}(0, \sigma = 2cm)$. The robot executes its nominal skill using this noisy state information. At $T = 1$, by the time the robot finishes executing the first

skill, we assume that the state estimator has converged to the ground-truth. Hence, the robot has access to the accurate handle position at this point. The robot uses its preconditions to check if any of its nominal skills can be executed. If so, it executes the remaining nominal skills. If not, it uses the failure classifier to identify the failure mode and execute the best recovery from that mode.

We compare our approach (RECOVERY) with open-loop execution (OPEN-LOOP) of nominal skills (table 5.2). The success rate of OPEN-LOOP is sensitive to the handle and varies from $50 - 80\%$ By contrast, RECOVERY improves the success rate to $80 - 90\%$ consistently across all of the 5 handles even though it was trained only for handle 1. Both OPEN-LOOP and RECOVERY struggle at the full-sized door due to the slippery cylindrical handle. However, our approach still does significantly better than OPEN-LOOP which only succeeded in 3/10 attempts. We expect recovery performance to improve with further training and by the use of a good state estimator which will enable closed-loop behavior. Importantly, our approach did not induce any additional failures which indicates good transfer of the preconditions and failure classifier learnt in simulation.

4.4.2 Evaluation of MetaReSkill

In this evaluation, we assume that we have access to an accurate model of the state estimator so that we can estimate the failure distribution accurately. We discover failures using our early termination discovery strategy along with a simulated state estimator that halves the standard deviation of the noise distribution after every robot action. We discover a total of 2000 failure states which we group into 5 clusters using the GMM. These failures are less diverse than the failures discovered by pessimistic discovery and are more concentrated near the door handle. We compare round-robin learning of recoveries with MetaReSkill in figure 4.7 using 5 different seeds. We use $\alpha = 0.95$ to compute the confidence interval, window size w of 3 and query REPS for a new data-point in every round, i.e. $\eta = 1$. We initialize the UCL estimates by training every recovery twice in a round-robin order. Not only does MetaReSkill improve significantly faster than round-robin, but also it converges to a better objective in all the trials. In 3/5 trials, MetaReSkill used only **70%** of the training budget to achieve the best objective achieved by round-robin, i.e., 1 hour earlier. This shows that it

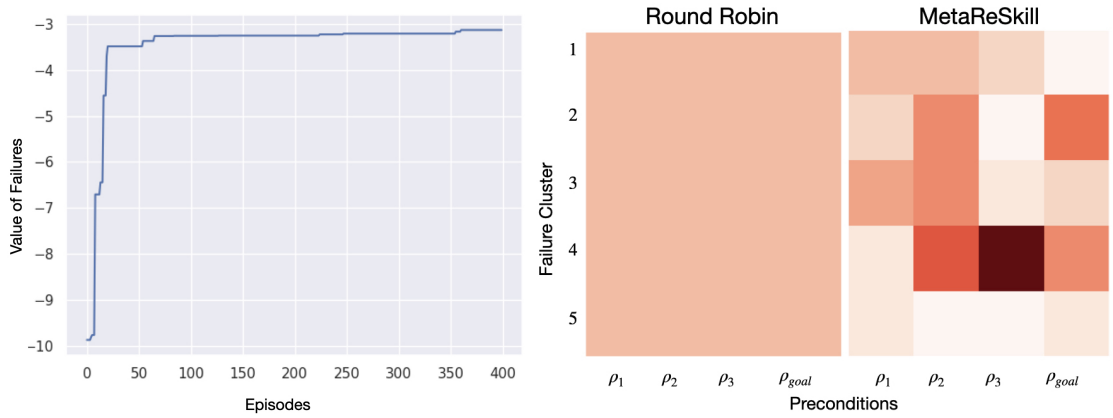


Figure 4.6: *(left)* **VoF Saturates.** After quick initial improvement, the objective saturates when recoveries are trained in a round-robin order. *(right)* **Allocation.** We show how many rounds each of the 20 recoveries were trained for by Round-robin and MetaReSkill. Each recovery is identified by the pair (i, j), where, i is the failure cluster it is meant for and j is the precondition it recovers to. Round-robin trains all of them equally while MetaReSkill prioritizes a small number of promising recoveries that improve the VoF by the most.

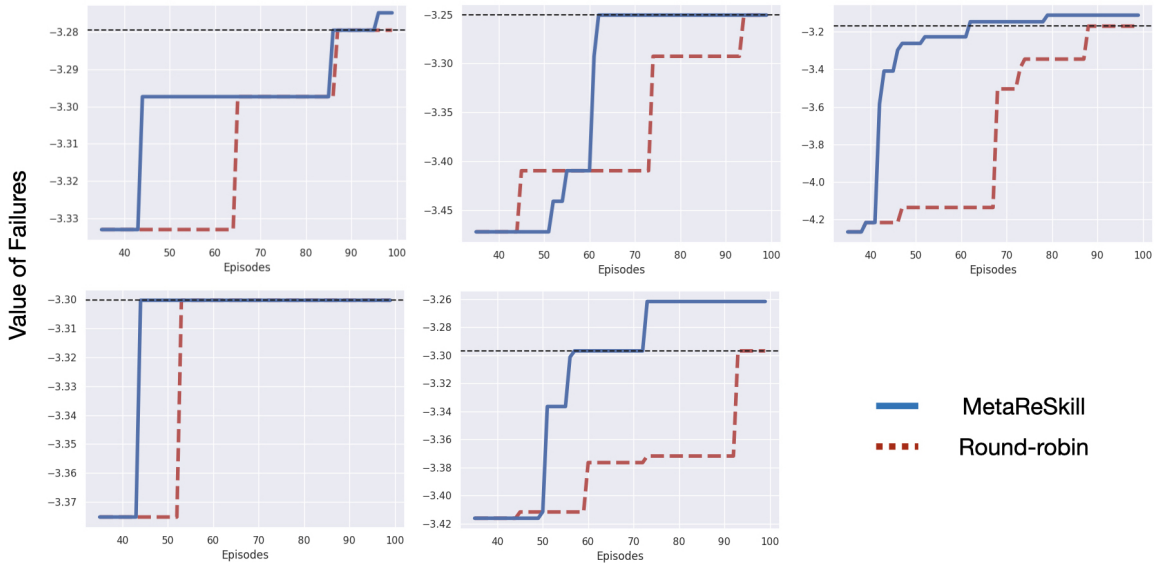


Figure 4.7: We compare MetaReSkill with round-robin learning of recoveries over 100 REPS training episodes with 5 different seeds. MetaReSkill is initialized by training each recovery twice initially in a round-robin manner. Hence, we see a difference in performance from episode 40 when MetaReSkill kicks in. MetaReSkill immediately focuses on the most promising recoveries which allows it to optimize the VoF significantly faster, also converging to a better VoF in every trial.

can make better use of training resources to improve robustness.

4.5 Discussion

In this chapter, we considered the problem of active skill learning for multistep tasks motivated by the goal of sample efficient and scalable learning. We defined the active learning problem as an online decision problem of selecting skills for training that are going to improve the robot's overall performance most quickly. This decision depends on a number of factors, *viz.*, the overall task distribution, the difficulty of learning each skill and its relative importance to the overall task.

We proposed a P2L algorithm called Meta-Reasoning for Skill Learning, *i.e.*, MetaReSkill that directly solves this decision problem. MetaReSkill makes use of a symbolic skill graph, wherein, each vertex corresponds to the precondition of a skill while an edge between two vertices u and v corresponds to the skill that is initiated in u and terminates in v. Each edge in this symbolic graph has an associated transition probability which reflects the success rate of the skill in reaching its intended goal v. MetaReSkill estimates a probabilistic *skill improvement model*, *i.e.*, the difficulty of learning skills online by monitoring their progress and computing their rate of improvement in each round. The skill improvement model is combined with the symbolic skill graph to build a *task improvement model* which can predict the improvement in the overall task if a certain skill is further trained. The task improvement model is then used online to choose the skill that is likely to best improve the robot's performance. We evaluated MetaReSkill on the problem of recovery learning for multistep manipulation problems. First, we discovered failures of given nominal skill in simulation by evaluating them under simulated state uncertainty and then used MetaReSkill to actively train recovery skills. Compared to baselines, we found that the learnt recovery skills significantly improve task success both in simulation and in the real world. We also found that MetaReSkill made a much better use of computation than round-robin skill learning.

4.5.1 Limitations

While MetaReSkill can in principle be used for learning in the real world, we limited ourselves to learning skills in simulation in this chapter. There are a number

of benefits to learning in simulation, such as, access to privileged information, the ability to reset to failures and scalability. On the other hand, learning in simulation requires setting up a reasonably accurate model of the task being solved. This can be infeasible for certain problems which are difficult to simulate, *e.g.*, soft objects. Hence, an exciting direction for future work is to apply MetaReSkill to learn skills in the real world. Kumar et al. (2024) have recently proposed a plan to learn approach that is similar to MetaReSkill for practising parameterized skills in the real world.

Second, MetaReSkill requires a reasonably accurate symbolic skill graph to build the task improvement model. While estimating the success rates of skills is relatively easy, we found it challenging to accurately learn skill preconditions that correspond to vertices in the skill graph. Training a skill precondition accurately requires a diverse dataset of positively and negatively labeled samples which was challenging to collect. We relax this requirement by proposing a model-free reinforcement learning approach for the active recovery learning problem in the next chapter.

5 P2L for Recovery Learning: RecoveryChaining

Figure 5.1: Due to uncertainty in the grasp of the object, the robot ends up making contact with the shelf during execution resulting in a collision as well as an in-hand slip failure. However, using the proposed recovery chaining framework, the robot recovers from the collision state and then hands over control to the nominal place skill. The learnt recovery also allows the robot to correct the slip by intentionally making contact with the shelf wall. Note that the policy is trained entirely in simulation.

So far in this thesis, we have taken a model-based approach for planning to learn. Both in chapters 3 and 4, we first learned a symbolic representation of the tasks by learning skill preconditions. The effectiveness of our algorithms is hence heavily reliant on the accuracy of these preconditions. This chapter[1] relaxes that requirement by learning recovery skills for multistep manipulation without relying on a symbolic representation of the task.

Consider a robot trying to tidy up a human's house. A robot trying to tidy up a

[1]The work presented in this chapter was done in collaboration with Devesh Jha and Diego Romeres at Mitsubishi Electric Research Laboratories (MERL).

house is confronted with a myriad of possible failures. It might pick up a half-read *'Planning Algorithms'* book from the table and try to place it in the top shelf but fail to see the objects already inside. How does it respond when the book bumps into the clutter and starts slipping out of its hand? One potential recovery behavior would be to first fix the slip by pushing the book against the shelf and then repositioning it for another attempt. Humans can quickly come up with robust strategies to deal with such failures, often with just partial information. For example, we often use our sense of touch to extract objects from a bag when we don't have a clear view of its contents, and we use our quick reflexes to recover from slipping on a patch of ice.

However, popular model-based planning approaches (Mason, 2001; Kroemer et al., 2021) struggle to generate such behaviors on-the-fly because of their reliance on an approximate model of the environment. This reliance leads to failures when the robot is faced with large inaccuracies in the model during deployment (Lagrassa et al., 2020). Hence, robots are usually deployed with recovery behaviors to gracefully handle potential failures. Common recovery strategies include retrying the previous step (Ebert et al., 2018), backtracking (Wang and Kroemer, 2019) and hand-designed corrective actions (Sundaresan et al., 2021). These heuristic strategies can be sub-optimal and require significant manual engineering effort. In this paper, we propose to use reinforcement learning (RL) (Sutton and Barto, 2018; Lillicrap et al., 2015) to automate the discovery of robust recovery behaviours for multi-step manipulation. RL is capable of discovering and learning complex robot skills (Haarnoja et al., 2023; Zhou and Held, 2023) but is limited by the twin problems of (1) high sample complexity and (2) significant reward shaping.

To address this, we propose a novel hierarchical reinforcement learning (HRL) formulation **RecoveryChaining** for recovery learning that is much more sample efficient than flat RL and can solve challenging manipulation problems even with a sparse reward. Our main idea is to use a hybrid action space which consists of primitive robot actions and temporally extended nominal options that transfer control to one of the model-based controllers. During exploration, when the agent takes a nominal action at a state, it verifies in simulation whether or not the task can be solved reliably by transferring control to the nominal controllers from that state. The verification is then used as a binary reward signal for the recovery policy. RecoveryChaining not only learns how to recover from the immediate source of failure but

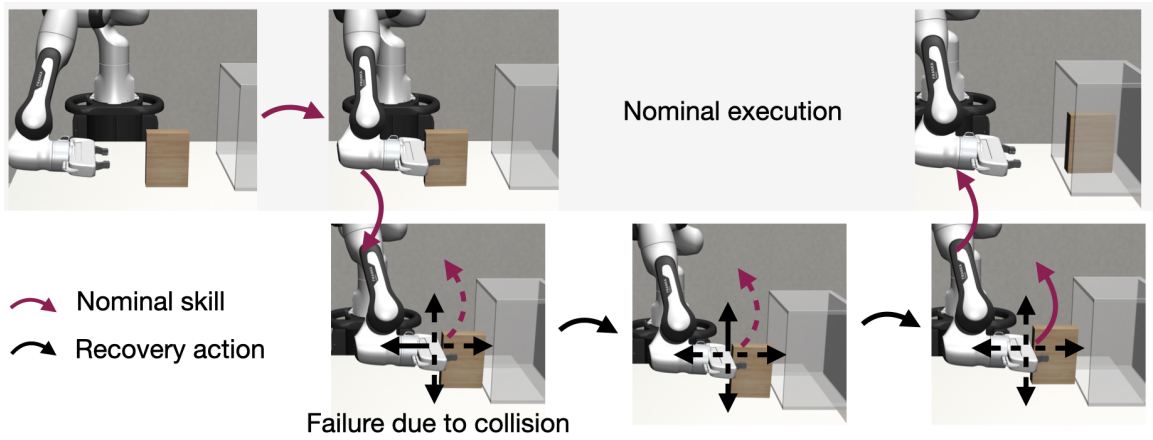


Figure 5.2: We propose an approach to learn robust recovery behaviors on top of given nominal controllers using reinforcement learning that works even with sparse rewards. Here, the robot is trying to place a box on a shelf but accidentally collides with the shelf due to an imprecise grasp. Using our approach, the robot learns a recovery policy from the failure state in a *hybrid* action space consisting of primitive robot actions and temporally extended nominal options that trigger a sub-sequence of the nominal controllers. The recovery policy is trained to quickly take the robot to the precondition of one of the nominal controllers so that it can transfer control to the nominal controllers to complete the task. Solid arrows indicate actions taken by the robot and dashed arrows other available actions.

also which nominal controller to recover to.

We evaluate our approach in three multi-step manipulation tasks of pick-place, shelf, and cluttered-shelf. The results show that our approach is able to learn significantly more robust recoveries than prior methods. In some of the shelf domain scenarios, the robot is able to learn to leverage contact with the environment to reduce uncertainty, adjust for slip, and avoid further collisions. We also show that our approach is suitable for sim-to-real by transferring the learned skills to a physical Mitsubishi Electric Assista arm without the need of any real-world fine-tuning. (see figure 5.1).

Assumptions. Our approach makes two key assumptions: (1) *Training in simulation.* We learn recovery policies in a physics-based simulator, like MuJoCo (Todorov et al., 2012) with the capability to reset to a previously observed state. (2) *Failure detector.* We assume access to a failure detector module that can detect undesirable states to prevent the robot from ending up in an irrecoverable failure.

5.1 Related Work

Recovery Learning

Recent works have explored the idea of learning recovery policies using offline datasets for safe exploration (Thananjeyan et al., 2021; Wilcox et al., 2022) and for recovery from execution failures (Reichlin et al., 2022). For example, (Wilcox et al., 2022) learn a safe set using an offline dataset. The learned preconditions (or safe sets) are then used as the goal for recovery learning. Similarly, the approach presented in chapter 4 also relies on a symbolic representation learned offline. This approach is highly sensitive to the quality of the learned preconditions and is pessimistic as it tries to stay close to the offline dataset. To address these issues, the approach presented in this chapter uses online RL with temporally extended actions to better explore the state space and discover robust policies.

Another common approach for learning reactive policies is to learn from human demonstrations (Chernova and Thomaz, 2014; Chi et al., 2023). However, these policies are prone to failure if the robot visits out-of-distribution states during execution. Online data collection (Ross et al., 2011) by an expert human is often required to learn recoveries which makes this approach quite expensive. By contrast, our approach does not rely on human demonstrations. (Wang et al., 2022) propose to recover by modulating dynamical systems learned from segmented demonstrations. However, this assumes accurate detection of manually specified modes which we do not require.

Hierarchical Reinforcement Learning

Our recovery learning approach utilizes the hierarchical reinforcement learning (HRL) framework, wherein recovery policies from failures are learned to connect them to the nominal policies. Our approach is also inspired by previous works which show that manipulation policies can be learned more efficiently by using structured action spaces (Sharma et al., 2020; Nasiriany et al., 2022; Rosen et al., 2022) such as, object-centric controllers and parameterized primitives. We discuss HRL in section 2.3.

Hybrid Methods

Hybrid methods combining the model-free and model-based techniques for RL have also been explored in past research. Chebotar et al. (2017) combine model-

based and model-free updates for time-varying linear Gaussian controllers. There has been prior work by Lagrassa et al. (2020) to learn a model-free policy from expert demonstrations to complete the task in regions where the model-based planner failed. Uncertainty has also been to guide switching of controllers between model-based and model-free policies (Lee et al., 2020). The model-based policy acts as a funnel to guide the robot towards the area of interest while the model-free policy is used when the uncertainty is high. Similarly, residual RL has been explored in past (Johannink et al., 2019) where the problem of feedback control of complex tasks is solved by decomposing them into a part that is solved efficiently by conventional feedback control methods, and the residual which is solved with RL.

5.2 Problem Statement

Consider a long-horizon contact-rich manipulation task defined by a distribution of start states and a binary goal function f_{goal}. We are given a sequence of k nominal model-based controllers (or nominal policies) $\Pi^{nom} := (\pi_1^{nom}, \ldots, \pi_k^{nom})$ capable of solving the task with non-zero probability. The nominal policies can form a chain, as in figure 5.3, if they are applied sequentially but they can form a graph more generally. Due to state and actuation uncertainty or model inaccuracy, the system may enter a state (shown in red) in which none of the nominal policies are applicable. We utilize an anomaly or failure detector module that monitors the execution of the system and raises a flag `fail-condition` if unsafe or unexpected conditions are met, for example, high end-effector forces, dropping an object, or slip. Collision while moving in free space is a failure but collision when pushing an object on the table is expected. The failure detector can be hand-designed or learned from demonstrations (Pastor et al., 2012). Our goal is to robustify the system by efficiently learning a separate *recovery policy* that allows the robot to complete the task even from these failure states. In the following section, we describe our proposed approach and present solutions to some of the key challenges associated with this learning problem.

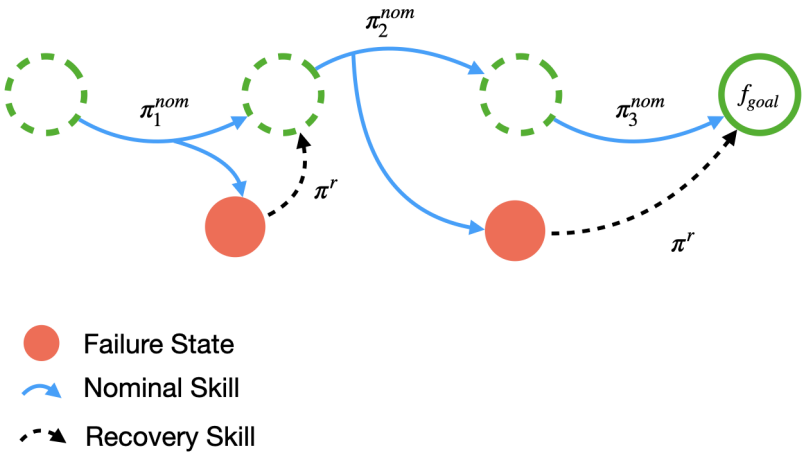


Figure 5.3: Representation of a sequence of nominal policies that solve a task specified by a binary function f_{goal}. Due to model inaccuracies and stochastic dynamics, the system may end up in a failure state from which the nominal policies are not applicable. The recovery policy π^r is learned to take the system back on the nominal chain.

5.3 Approach

We model the system as a MOMDP (Ong et al., 2010), wherein the robot maintains an estimate $\hat{s} := (x, \hat{y})$ of the true state s and acts based on $(\hat{s}, o)$, where $o \in O$ corresponds to sensory observations such as proprioception. Our approach, visualized in fig. 5.2, involves two steps: (1) **Failure Discovery.** Nominal policies are executed under various conditions in simulation to induce and record failures. We leverage privileged information in simulation to record both the true state s and the corresponding observations associated with a failure. This allows us to directly reset to the failure in simulation during recovery learning. (2) **Recovery Learning.** Recovery policies are learned in simulation to handle the failures collected in the previous step using reinforcement learning.

The final robot policy Π is composed of the nominal policies $\{\pi_i^{nom}\}_{i=1}^k$ and the recovery policy π^r.

$$\Pi(s, a, t) = \begin{cases} \pi_t^{nom}(s, a) \text{ if } \texttt{fail-condition} \text{ is false} \\ \pi^r(s, a) \text{ if } \texttt{fail-condition} \text{ is true} \end{cases} \tag{5.1}$$

5.3.1 Failure Discovery

We discover and record potential failures $\mathcal{D}_{fail}$ of the nominal policies by executing them from various initial conditions. Once a failure is detected, execution is terminated and the resulting state $s = (x, y)$ is recorded. Though, the robot may not have access to the true world state at test time, we can do so during training either in simulation or with extra sensors in the physical world. This is straightforward in simulation but would require extra sensing on a real system during training, for example, using AprilTag markers.

5.3.2 Recovery Learning

Our goal is to learn a policy that reliably takes the robot to the precondition of one of the nominal policies from all the failures $\mathcal{D}_{fail}$. For example, if the book in a robot's hand slips while putting it on a shelf, recovery should regrasp the book such that the place controller can be executed. Prior works (Thananjeyan et al., 2021; Wilcox et al., 2022; Reichlin et al., 2022) in recovery learning explicitly learn the nominal precondition $\mathcal{I}_i^{nom}$ offline and compute actions to satisfy it for recovery. The agent computes actions to maximize the precondition $r(s, a) = \mathcal{I}_i^{nom}(s)$ and hence relies on accurate estimation of the precondition. This is often difficult in practice and highly dependent on the quality of the offline dataset making these approaches brittle and pessimistic. We discuss these issues in more detail in section 5.3.3.

To address these drawbacks, we propose an online RL approach that does not rely on a learned reward function and works even with sparse rewards. Our main observation is that we can compute a monte-carlo estimate of the precondition of a nominal controller π_i^{nom} by executing the chain of nominal controllers in simulation $(\pi_i^{nom}, \ldots, \pi_k^{nom})$ and verifying whether the goal was achieved or not. Thus, the precondition can be estimated as $\mathcal{I}_i^{mc}(s) = r(s, (\pi_i^{nom}, \ldots, \pi_k^{nom}))$.

A Straw-man Approach

A straw-man RL approach to leverage this observation would be to replace the learned $\mathcal{I}_i^{nom}$ with $\mathcal{I}_i^{mc}$. However, this is highly inefficient as the environment would have to execute long sequences of nominal controllers, potentially multiple times, in every step to compute the reward. Ideally the environment need not have to repeat-

edly compute the monte-carlo estimates from states well outside the precondition.

RecoveryChaining

Our key insight is *to let the RL agent decide when to estimate the precondition.* We provide the RL agent with temporally extended *nominal* options o_i^{nom} which simulate the transfer of control to the nominal policy π_i^{nom}. The sequence of nominal policies $o_i^{nom} = (\pi_i^{nom}, \ldots, \pi_k^{nom})$ is then executed from that state and the resulting task success is used as a reward. We choose to make the nominal options terminal so that the corresponding reward is independent of the policy being learned and hence is stationary. A single simulation is sufficient in deterministic MDPs but additional simulations may be required in highly stochastic domains. We use only a single simulation in our experiments. Monte-carlo simulations are computationally more expensive than querying a learned precondition but provide a more reliable reward estimate in our experience.

RecoveryChaining MDP

Let A be the original action space of the agent consisting of primitive actions and S be the state space. Then, the RecoveryChaining MDP is defined by the tuple $(S, A^{rc}, T, r^{rc}, \gamma, \mu^{rc})$ where

- A^{rc} is a hybrid action space $A \cup \{o_1^{nom}, \ldots, o_k^{nom}\}$ consisting of primitive actions and terminal nominal options that transfer control to the nominal policies.
- $r^{rc}(s, a, s') = f_{goal}(s')$
- $\mu^{rc} = \mathcal{D}_{fail}$

The hybrid action space is visulialized in figure 5.4. Intuitively, when the agent is far away from the precondition $\mathcal{I}_i^{nom}$ and executes o_i^{nom}, it gets no reward. After a few trials and errors, the agent identifies states that lie outside the precondition and stops executing the nominal option from those states. On the other hand, as the agent gets closer to the precondition, it receives higher rewards upon executing o_i^{nom} and implicitly learns that it is inside the desired precondition.

This approach has three key advantages:

1. The agent implicitly learns the nominal preconditions through trial and error and stops computing monte-carlo estimates for nominal controllers that are not applicable. This makes it much more computationally efficient than the straw-man approach.

2. The ability to try all the nominal policies allows the agent to potentially discover novel ways to reuse them.

3. The agent is not obligated to recover to any precondition. If the correct recovery is to completely avoid the nominal policies then it can discover such strategies.

Early Termination

We use an additional trick to further optimize the computation of reward for the nominal actions. Failure states are designated as absorbing states in the RecoveryChaining MDP. This allows us to terminate a monte-carlo simulation of the nominal policies early if the `fail-condition` is met. In practice, this speeds up the simulations as the application of a controller without satisfying its precondition often quickly results in collisions and slip.

State Representation

To recover from failures due to inaccurate and partial state information, it is necessary that the robot has access to multiple sensing modalities, such as vision, proprioception and tactile sensing. Proprioception, in particular, is commonly available and provides reliable information about how the robot is making contact with the world. We use an asymmetric learning approach, where we leverage full state information available in simulation for failure discovery but train the recovery policy to use only those state variables that will be available at deployment. To bridge the sim-to-real gap, we convert end-effector force signals into a binary signal using a threshold. Further, we also provide the recovery policy with a limited amount of history in partially observable domains which is critical for it to learn a robust recovery.

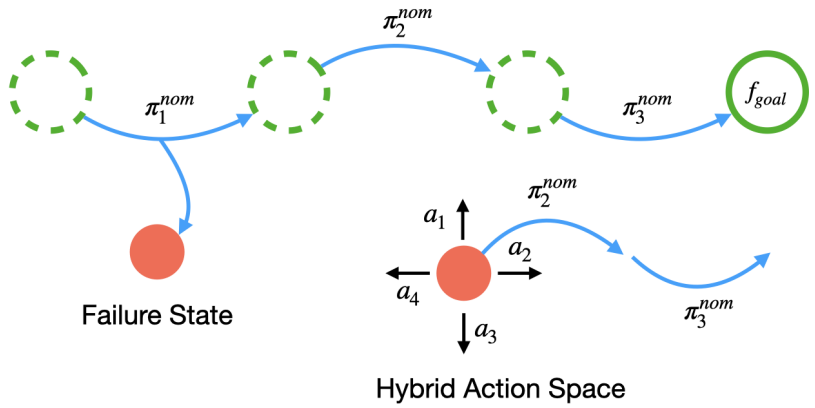


Figure 5.4: We use a hybrid action space for reinforcement learning. It consists of both primitive robot actions and nominal options that transfer control to a sequence of nominal policies that can take it to the goal if applied successfully.

5.3.3 RecoveryChaining vs Pretrained Preconditions

Most previous recovery learning works learn the nominal precondition (Vats et al., 2023) or the safe set (Wilcox et al., 2022) using an offline dataset. The recovery policy is then trained to satisfy the leaned precondition. While this approach is efficient as it decouples recovery learning from the overall task, it suffers from two issues in practice:

Brittleness

Recovery learning is highly sensitive to the quality of the learned preconditions. RL is a reward maximizer that is known to find and exploit loopholes even in carefully hand-designed reward functions (ope, 2016). A learned reward function aggravates this issue as the agent is likely to explore states that are quite different from the states where data for learning the reward function was collected. It is a challenge in manipulation, in particular, because the complex and non-linear dynamics induced by contact make generalization from limited data difficult. Hence, there is a risk that the agent may learn an undesirable behavior using a pre-trained

Pessimistic Bias

Learning with pretrained preconditions is pessimistic as the agent needs to stay close to the offline data. While this is effective at dealing with small perturbations it cannot discover complex recovery behaviors that require the agent to visit an out-of-

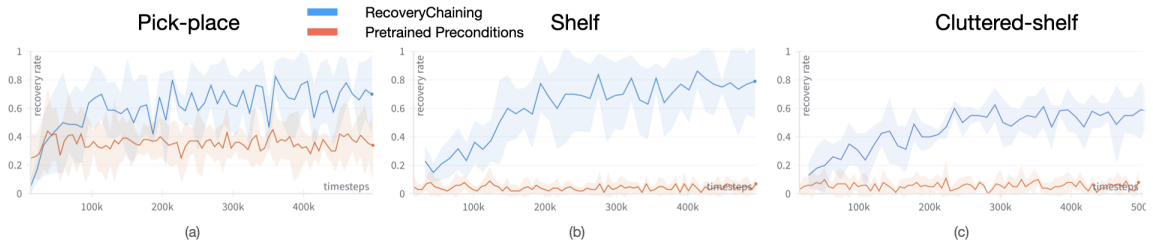


Figure 5.5: Comparison of the learning curves between RecoveryChaining (RC) and Pretrained Preconditions (PP) in pick-place, shelf and cluttered-shelf domains. RC makes consistent progress in learning while PP hits a local optimum early in training. It is not able to further improve its policy as it is limited by a learned reward model. PP does quite poorly on the shelf task due to its partially observable nature. Results are averaged over 5 runs.

distribution state. Recovery sometimes requires novel use of existing policies which is not possible with a pessimistic bias in learning.

5.4 Experiments

We evaluate our proposed approach in three challenging multi-step manipulation environments. The purpose of these experiments is to understand (1) whether our approach can learn robust and composable recoveries and (2) how well the hybrid approach of combining model-based and model-free policies works.

Baselines

We compare our method RecoveryChaining (RC) with three baselines. *Nominal:* completes the task using just the model-based controllers. *Pretrained Preconditions (PP):* learns a recovery policy to reach preconditions learned from offline data. *RL:* learns a model-free policy for the entire task using RL with a hand-designed dense reward and just the primitive actions.

RL Training

We use a discrete action space to learn all the policies. Each action primitive is defined in the robot's end-effector frame and operates only on a single dimension. Each translation action primitive moves the end-effector by ± 2 cm along x, y or

z axis and the rotational primitives apply a roll, pitch or yaw of $\pm\pi/2$. We train all the RL approaches for 0.5 million timesteps using Proximal Policy Optimization (PPO) (Schulman et al., 2017) from stable-baselines (Raffin et al., 2019). Most of the default hyper-parameters worked well us so we didn't have to do much tuning. We run all the methods using 5 different seeds and report the average.

5.4.1 Pick-Place Domain

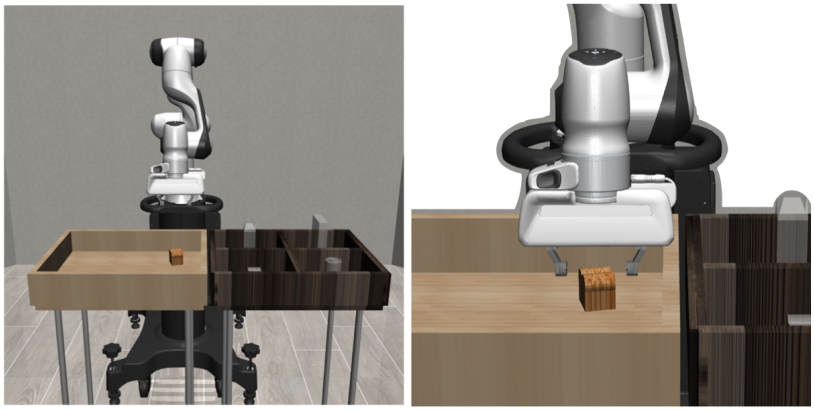


Figure 5.6: The pick-place task requires the robot to pick a small bread from the source bin and place it in the target bin. The nominal controllers do not account for the sides of the bin because of which the end-effector collides with them when the bread is close to the walls. One such situation is shown in the right figure.

Our first domain is the pick-place task from robosuite (Zhu et al., 2020). To solve this task, the robot needs to pick a small loaf of bread from the source bin and place it in the target bin. The robot gets a 46 dimensional observation consisting of object poses, end-effector pose, etc. The bread is initialized in a different location in the source bin in every episode.

Nominal Skills

We designed four Markovian nominal controllers: GoToGrasp skill takes the robot to a pre-grasp pose over the object, Pick skill picks up the object, GoToGoal moves the robot to the drop location and Place skill places the object at the drop location. Each controller is implemented as a state machine that selects a target pose for the robot end-effector in the Cartesian space based on its current state. The

target end-effector pose is then achieved using task-space impedance control with fixed impedance. The nominal skills achieve a success rate of 70% on their own in this task.

Failures

Most of the failures in this task occur when the initial location of the bread is close to one of the walls. The robot needs to reach inside the bin to grasp the object due to its small size. This causes the end-effector to collide with the wall leading to a failure that the nominal skills aren't capable of handling. Collision is detected using a threshold on the end-effector forces. We collect a total of 100 failures for learning recovery.

Recovery

The learned policy finds two different strategies to recover from these failures. The first strategy rotates the end-effector along the z axis so that it does not collide with the wall when the robot reaches inside the bin. The second, and perhaps more interesting, strategy uses the gripper fingers to push the object away from the walls before picking it up. RC uses a combination of these two strategies to improve the success rate from 70% to 90%. On the other hand, PP quickly plateaus and converges to a locally optimal policy (fig 5.5). It uses preconditions learned from around 300 nominal trajectories as the reward function. RL is not able to learn the task at all. We summarize the success rates in table 5.1.

5.4.2 Shelf Domain

Our second domain is a shelf environment with state uncertainty also implemented using robosuite. In this task, (fig. 5.9) the robot needs to pick up a box from a table and place it inside a shelf in an upright position. The robot observes a noisy estimate of the position of the box, where, the noise is sampled from a zero-mean Gaussian distribution with standard deviation of 1 and 2 cm along the y and z axes, respectively. We also provide the robot the number of actions taken so far as an observation. This allows it to learn open-loop policies if needed. The dimensions of the shelf and the box and the position of the shelf are sampled randomly in every episode.

Nominal Skills

We designed three nominal skills for the task assuming that the observations are accurate. PICK skill: the robot goes to the observed position of the box, closes the gripper, and picks up the box; MOVE skill: the robot moves to a pre-placement position conditioned on the given position of the shelf; PLACE skill: the robot places the box on the shelf and retracts. These nominal skills can complete the task reliably if the state estimates are accurate but can fail when the state estimates are wrong. All the nominal skills control the robot using task-space impedance control with fixed impedance.

Failures

We collect failures by executing the nominal policies in simulation under state uncertainty. The failure conditions are given by the end-effector forces F_e exceeding a predefined threshold. Note that these thresholds are different for simulation and real-world and need to be tuned appropriately. The nominal controllers assume perfect pose of the box. However, the robot may grasp the box with an offset due to a wrong position estimate. This leads to mainly two types of failures: (1) *collision*: robot collides with the shelf or the table (2) *collision-slip*: collision with the shelf leads to in-hand rotation of the object if there is a delay in stopping the robot.

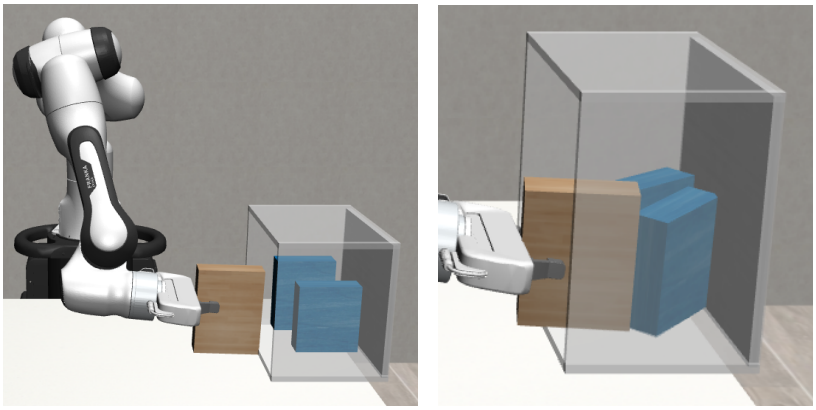
Recovery

The recovery policy learns to move up and inside the shelf before switching to the nominal skill because most collisions happen below the center of mass of the object. We found that providing the number of past actions was crucial to learn a recovery in this task because of unreliable state estimates. In our ablation studies with different amounts of state uncertainty, we found that the policy tends to be more conservative and learns relatively open-loop policies under high uncertainty. Overall, RC did significantly better than PP on this task by improving task success from 46% to 83.2%. RC was not able to recover from failures involving significant in-hand rotation of the box as it was not provided with its orientation observation. Additional sensing, for example, slip detection using a tactile sensor can further improve the recovery policy. The reward model for PP was learned using 232 nominal trajectories.

	Nominal	**Nominal + RC**	Nominal + PP	RL
Pick-place	70	**90**	75.6	0
Shelf	46	**83.2**	56	0
Cluttered-shelf	39.2	**57.2**	43.2	0

Table 5.1: Comparison of the overall success rate. RecoveryChaining (RC) significantly robustifies the nominal controllers in both the domains and is the best performing method. The improvement is more pronounced in the shelf domain as it has state uncertainty which makes learning more challenging. RL was not able to solve the tasks at all despite a dense reward.

5.4.3 Cluttered Shelf Domain



(a) Cluttered shelf domain. (b) Failure state.

Figure 5.7: (a) The robot needs to place a box on a cluttered shelf with two objects. In addition to avoiding collision, successful task completion requires the robot to avoid rotating the objects on the shelf. (b) The robot collides and rotates the objects during execution leading to a failure.

We consider a more complex version of the shelf domain with two objects randomly placed on the shelf. The robot needs to avoid them while putting the box on the shelf. We use the same nominal skills used in the shelf domain.

Failures

In addition to monitoring end-effector forces for collision detection, we also use vision-based failure detection to avoid toppling objects on the shelf. The failure de-

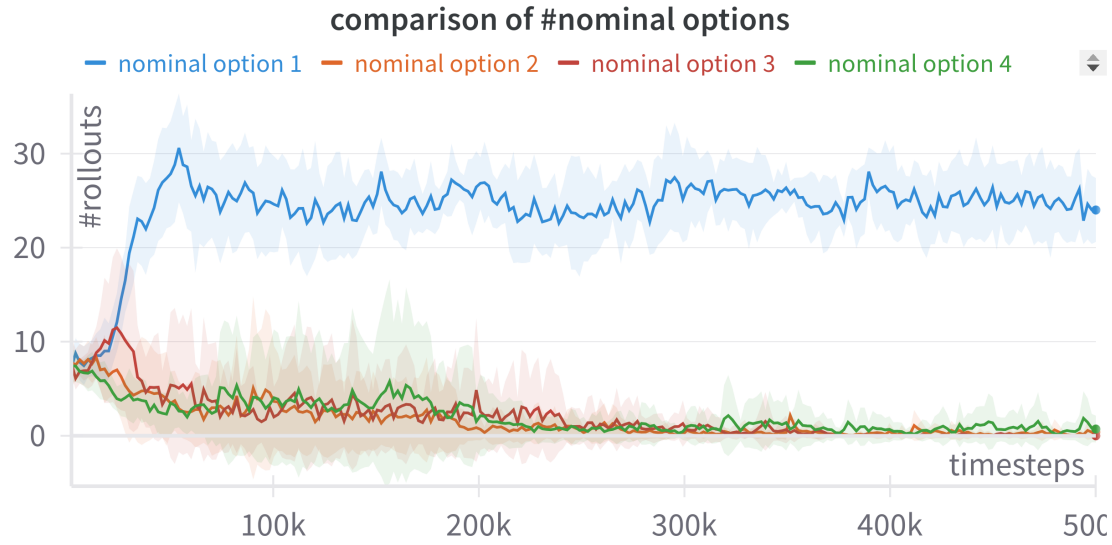


Figure 5.8: A comparison of the number of different nominal options taken by the agent in the pick-place task in every round of exploration consisting of 120 actions. The agent explores all the nominal options initially but quickly identifies and commits to the best nominal controller to recover to.

tector is triggered if the robot topples or rotates any object by more than a predefined threshold. Such failures can be detected in the real world by using object detectors and pose estimation while we use privileged state information in simulation. We show an example of a failure in figure 5.7.

Recovery

We compare the learning curves of RC with PP in figure 5.5. RC learns a significantly more reliable recovery skill than that learned by PP. However, we observe a drop in performance compared with that on the simpler shelf task. We believe performance can be improved with longer training, a dense reward function and a better action space.

5.4.4 Analysis of Learned Recoveries

RC can identify the best nominal controller to recover to.

We do not provide the agent any prior knowledge about where it should recover to and which nominal controller it should switch to. To understand the exploration behavior of RC we plot the number of times the agent chooses each nominal option in a round of exploration in figure 5.8. Each round consists of 120 actions. We observe that the agent initially explores all the nominal controllers but quickly identifies the most suitable one and commits to recovering to its precondition. The RC policy *implicitly* learns the preconditions of all the nominal controllers through trial and error and avoids trying to switch to a nominal controller at states in which it is unlikely to succeed.

RC can reuse the nominal controllers in novel ways.

Prior approaches that use pretrained preconditions, like skill chaining, provide a pessimistic bias to the RL agent by freezing the preconditions after the initiation period. This prevents the agent from discovering novel ways to use the nominal controllers that may be quite different from the trajectories used to train the precondition. Sometimes, the robot needs to use the nominal controllers in novel ways not seen during nominal execution. We describe one such novel reuse of the PLACE skill discovered by our agent for the shelf task in figure 5.9. In this example, the box undergoes significant in-hand rotation due to a collision with the shelf below its center of mass. The robot needs to re-grasp the box before placing it on the shelf as it may fall over otherwise. The RC agent discovers that switching to the nominal PLACE skill from inside the shelf fixes the in-hand slip by aligning the box with the back of the shelf. This behavior is well outside the distribution of states visited by the nominal skills as the PLACE controller is designed to be triggered outside the shelf.

5.4.5 Transfer to a Physical Robot

We transfer the nominal and recovery policies to a real robot. Our real-world setup includes a 6-DoF Mitsubishi Electric Assista robot arm with a WSG-32 parallel-jaw gripper and a Mitsubishi F/T sensor 1F-FS001-W200 mounted on the wrist of the

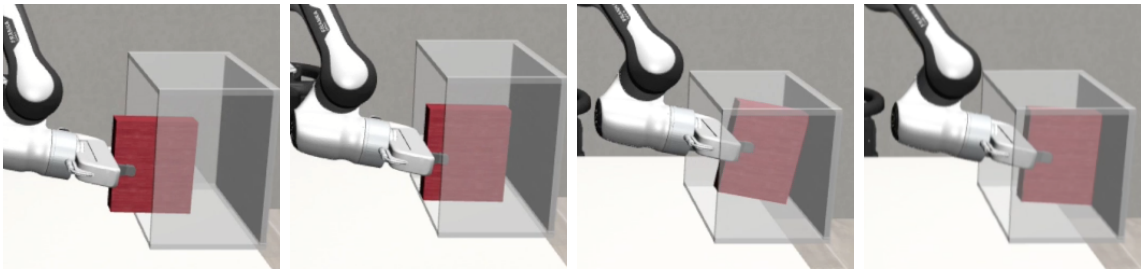


Figure 5.9: While trying the nominal controllers from different states during exploration, the agent discovers a novel application of the PLACE controller. *(top)* The PLACE skill was designed to gently place the box assuming it is upright. *(bottom)* To fix the slip due to a prior collision, RC learns to move deeper inside the shelf than nominal execution before switching to the PLACE skill. This allows the robot to fix the orientation of the box by pushing against the back of the shelf to ensure stable placement.

robot. We evaluate the feasibility of sim-to-real transfer of recoveries learned using our approach. We train the recovery in a simulated Assista robot with a box and then evaluate it on a real Assista with one box and two unseen objects- a mustard bottle and a can. Failures are induced on the real robot by providing incorrect position estimates of the shelf. Similar to the simulation, we implement a collision detector on the real robot by using observations from the F/T sensor on the robot. The robot was able to recover from both collision and slip (fig. 5.1) by using our recovery. The policy generalized well to the mustard bottle but performed slightly worse on the can because it is smaller than the box that was used for training and its curved surface is more prone to slip.

	Recovery Rate (%)
Box	100 (5/5)
Mustard bottle	100 (5/5)
Can	80 (4/5)

Table 5.2: Summary of real-robot experiments on a real robot. Our approach generalizes to a mustard bottle and a can despite having been trained only on a box.

5.5 Discussion

In this chapter, we proposed a hierarchical reinforcement learning approach to learn model-free recovery policies for robustifying nominal model-based controllers. Our approach, called RecoveryChaining, uses a hybrid action space to efficiently learn robust recovery policies that can be chained with model-based controllers. The action space contains temporally extended nominal options that transfer control to a specific nominal controller. These nominal options reduce the effective horizon of the task and enable recovery learning for multi-step manipulation. We evaluated our approach in three challenging domains and found that our approach can significantly improve task success using just a sparse reward. We also successfully transferred a recovery trained in simulation to a physical robot to demonstrate the feasibility of sim-to-real transfer.

5.5.1 Limitations

Similar to the previous chapter, the work presented in this chapter also relies on a physics-based simulator for recovery learning. This limits its applicability to tasks that can be modeled well in simulation and introduces the challenge of sim-to-real transfer. Furthermore, RecoveryChaining does require a large number of timesteps for learning a performant recovery policy despite being more efficient than flat RL. Hence, it may be challenging to apply this approach to learning in the real world. One middle ground is to first learn in simulation using our approach and then finetune in the real world using an adaptive framework like Xiong et al. (2024). There is scope for further improving the sample complexity of our method by using off-policy RL. The main challenge is that off-policy RL tends to be more unstable and hence harder to train than on-policy RL. Finally, more research is needed on efficiently learning recoveries in partially observable environments. Partial observability is a fundamental and ubiquitous challenge in open-world manipulation. In structured and lab settings, this can often be addressed via careful sensor placement and task design to make the environment fully observable to the robot. However, this is not feasible in unstructured settings in the open world where the robot has to intelligently reason about partial observability to operate autonomously.

6 Conclusion

6.1 Summary of Contributions

This thesis made the following key contributions:

- **Plan to Learn Framework.** This thesis advocated the principles of continual, active and collaborative robot learning to achieve the goal of general-purpose robots. Based on these principles, we introduced a novel plan to learn (P2L) framework for robot learning in which the robot decides which skills to learn according to the constraints it has to operate under and resources it has access to. P2L formalized active robot learning in complex situations as a meta planning problem which traded off the cost of learning against the improvement in overall performance.

- **P2L for Human-Robot Teams.** We mathematically formalized and solved the P2L problem for collaborative human-robot teams. We considered a human-robot team that had to fulfill a given sequence of orders at minimum expected human and robot effort. We proposed the Act, Delegate or Learn (ADL) planner to solve this problem. *To the best of our knowledge, ADL is the first optimal planner that jointly optimizes robot learning and task allocation.* We also showed that ADL was an instance of the uncapacitated facility location problem. This reduction provided us with $\log(n)$-approximation algorithms for ADL.

- **P2L for Multistep Tasks.** We formalized the P2L problem for skill learning in multistep tasks, such as assembling furniture as an active skill learning problem. We proposed Metareasoning for Skill Learning (MetaReSkill) which is the *first*

active skill learning algorithm for sequential decision problems. MetaReSkill estimated a probabilistic model of skill improvement to predict how individual skills would improve with additional training. This model was then used in conjunction with a task planner to identify skills which enable the most improvement in the overall objective. Compared to baselines, the learnt recovery skills significantly improved task success both in simulation and in the real world. We also demonstrated that MetaReSkill made much better use of computation than round-robin skill learning.

- **P2L for Recovery Learning.** We proposed a hierarchical reinforcement learning-based approach RecoveryChaining to efficiently learn recovery skills. To recover from a failure during a multistep task, a robot needs to decide where and how to recover. RecoveryChaining learned both where and how to recover by leveraging a hybrid action space consisting of primitive robot actions and temporally extended nominal options that transfer control to a model-based controller

- **Rigorous Real Robot Evaluation.** We demonstrated the effectiveness of the P2L framework by rigorously evaluating it on practically motivated and challenging manipulation tasks such as, opening doors, placing objects on a shelf under uncertainty and Lego assembly.

6.2 Directions for Future Work

We believe that this thesis is only the start of what could be a fruitful exploration of planning to guide robot learning. We hope that the developed framework and its instantiations on these manipulation domains pave the way for further research. We discuss some promising avenues for future work next.

6.2.1 ADL for Unordered and Uncertain Tasks

The ADL framework discussed in chapter 3 is designed for a fixed and known sequence of tasks. However, there are many domains, both in manufacturing and household settings, where the order of tasks can be altered or where there is uncertainty about future tasks. For example, consider a human-robot team that is asked to

assemble 100 chairs. The team can assemble the chairs in any order but the sub-tasks involved in assembling a specific chair have a fixed order. We hypothesize that the mixed integer programming approach used in ADL can be extended to decide the optimal ordering of tasks whenever possible. This would give robots the flexibility to identify the most useful skills and learn them early.

Handling uncertainty in future tasks may require more work. The current framework assumes a known set of future tasks, while uncertainty in tasks is usually represented by a probability distribution over tasks. One potential approach to adapt ADL is to plan with a set of tasks sampled from the task distribution. This could be a good proxy for the distribution if the sample size is sufficiently large. Another possibility is to plan greedily by picking the best action in expectation over the task distribution. However, this would not have the strong theoretical guarantees of ADL.

6.2.2 Learning Human Preferences

Human-robot collaboration calls for robots to assist humans while adapting to their personal preferences and learning new skills on the job. However, this assumption does not hold in many domains. For example, a household robot may not be sure where it should put the dishes after unloading a dishwasher. Users may have differing preferences about where they like to put different types of dishes. Similarly, many users may not be comfortable with the robot holding a knife. It is crucial for effective collaboration that the robot learns these preferences without overburdening the human with queries about their preferences. Prior work in interactive robot learning like Fitzgerald et al. and Bıyık et al. (2022) select informative queries by optimizing across multiple query modalities to maximize information gain. While this approach does allow a robot to learn human preferences more efficiently, it assumes that all preferences are equally important. In particular, these works make assumptions about the role of a robot in a human-robot team. The role of a robot depends on the robot's capabilities, the human's preferences and can change as the robot acquires new capabilities. Furthermore, preference queries should also depend on the frequency of tasks. It is much more important for a robot to know how to handle a human's daily mug than to know how to handle an expensive dinner plate that the human doesn't trust the robot with. We believe that our idea of goal-oriented skill learning proposed in chapter 3 can be generalized to the problem of goal-oriented preference learning.

6.2.3 Learning Skills in the Real World

The reliance on physics-based simulators in this thesis to discover failures and learn skills is a major limitation. This not only imposes the challenge of sim-to-real transfer but also it requires significant effort to design a reasonably accurate simulation of the environment. Learning in the real world is important since simulators cannot accurately capture the full range of situations a robot may face during deployment. Hence, a natural idea is to both discover failures and learn skills in the real world. However, this would require modifications in the approach presented in chapter 4 which assumes access to privileged state information and the ability to reset the simulator to arbitrary states. This poses interesting questions about how to actively select failures to recover from when the robot is not guaranteed to be able to recreate them. Failures that occur later are likely to be harder to recreate than failures that occur early during execution. Perhaps it is preferable to learn skills to handle failures closer to the start before moving on to those that occur later in the task. Furthermore, learning in the real world is cumbersome and expensive. This can be ameliorated by combining learning in the real world with that in simulation.

6.2.4 Partially Observable Environments

We explored failures due to partial observability in a number of our experiments. While we found that reinforcement learning can often learn useful Markov policies to deal with a limited amount of partial observability, a more principled solution is to learn policies that adapt to the history of past actions and observations. There are two popular approaches to incorporate history in the policy, *viz.*, using recurrent neural networks (RNN) to represent the policy and learning in the belief space. Both of these approaches have unique challenges. The use of RNN is more popular in the RL community due to its generality. However, this representation has to deal with the dual burden of memory, i.e., remembering important information and credit assignment. This makes it computationally much more challenging to train than training Markovian policies. In our experiments, we were unable to scale this approach to realistic robot skill learning. We believe that learning in the belief space is a more promising direction as this representation has been successfully used in robotics. However, the belief space is still higher dimensional than the state space and hence has higher sample complexity. To address this, we conjecture that POMDP planners

can be used in our RecoveryChaining framework to guide the reinforcement learning agent. We have already seen that model-based planners can drastically simplify RL in contact-rich tasks. Hence, it is conceivable that they may be similarly helpful in partially observable tasks by generating information gathering demonstrations that the RL agent may struggle to discover on its own.

Bibliography

Faulty reward functions in the wild, Dec 2016. URL `https://openai.com/research/faulty-reward-functions`. 5.3.3

Rakefet Ackerman and Valerie A Thompson. Meta-reasoning: Monitoring and control of thinking and reasoning. *Trends in cognitive sciences*, 21(8):607–617, 2017. 2.1

Charu C Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and S Yu Philip. Active learning: A survey. In *Data classification*, pages 599–634. Chapman and Hall/CRC, 2014. 2.2

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 1.1

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483, 2009. 3

Karl Johan Åström. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965. 4.1

Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019. 4.1

Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. Learning to optimize autonomy in competence-aware systems, 2020. 3.1

Djalel Benbouzid, Róbert Busa-Fekete, and Balázs Kégl. Fast classification using sparse decision dags. *arXiv preprint arXiv:1206.6387*, 2012. 2.1

Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022. 6.2.2

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 1.1

Maya Cakmak, Crystal Chao, and Andrea L Thomaz. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2 (2):108–118, 2010. 3.1

Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *International conference on machine learning*, pages 703–711. PMLR, 2017. 5.1

Jessie Y. C. Chen and Michael J. Barnes. Human–agent teaming for multirobot control: A review of human factors issues. *IEEE Transactions on Human-Machine Systems*, 44(1):13–29, 2014. doi: 10.1109/THMS.2013.2293535. 3.1, 3.2.1

Sonia Chernova and Andrea L Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014. 3, 5.1

Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009. 2.2, 3.1, 2

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL `https://doi.org/10.15607/RSS.2023.XIX.026`. 5.1

David Cohn. Neural network exploration using optimal experiment design. *Advances in neural information processing systems*, 6, 1993. 3

Open X.-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alexander Herzog, Alex Irpan, Alexander Khazatsky, Anant Raj, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Gregory Kahn, Hao Su, Haoshu Fang, Haochen Shi,

Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, and et al. Open x-embodiment: Robotic learning datasets and RT-X models. *CoRR*, abs/2310.08864, 2023. doi: 10.48550/ARXIV.2310.08864. URL `https://doi.org/10.48550/arXiv.2310.08864`. 1.1

Bruno Da Silva, George Konidaris, and Andrew Barto. Active learning of parameterized skills. In *International Conference on Machine Learning*, pages 1737–1745. PMLR, 2014. 2.2

Bruno Castro da Silva, George Dimitri Konidaris, and Andrew G. Barto. Learning parameterized skills. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL `http://icml.cc/2012/papers/826.pdf`. 3.2, 4.4

Andy Dearden, Michael Harrison, and Peter Wright. Allocation of function: scenarios, context and the economics of effort. *International Journal of Human-Computer Studies*, 52(2):289–318, 2000. 3.1

Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3876–3881, 2014. doi: 10.1109/ICRA.2014.6907421. URL `http://arxiv.org/abs/1307.0813`. 3.1

Kun Deng, Yaling Zheng, Chris Bourke, Stephen Scott, and Julie Masciale. New algorithms for budgeted learning. *Machine learning*, 90(1):59–90, 2013. 2.1

Frederik Ebert, Sudeep Dasari, Alex X Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In *Conference on Robot Learning*, pages 983–993. PMLR, 2018. 4.1, 5

Alexander Fabisch and Jan Hendrik Metzen. Active contextual policy search. *Journal of Machine Learning Research*, 15:3371–3399, 2014. ISSN 15337928. 2.2

P. Fitts, M. S. Viteles, N. L. Barr, D. R. Brimhall, G. Finch, Eric Gardner, W. F. Grether, W. E. Kellum, and S. S. Stevens. Human engineering for an effective air-navigation and traffic-control system, and appendixes 1 thru 3. 1951. 3.1

Tesca Fitzgerald, Pallavi Koppol, Patrick Callaghan, Russell Quinlan Jun Hei Wong, Reid Simmons, Oliver Kroemer, and Henny Admoni. Inquire: Interactive querying for user-aware informative reasoning. In *6th Annual Conference on Robot Learning*. 2.2, 6.2.2

Tianshi Gao and Daphne Koller. Active classification based on value of classifier. *Advances in neural information processing systems*, 24, 2011. 2.1

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992. 3

Matthew Gombolay, Ronald Wilcox, and Julie Shah. Fast scheduling of multi-robot teams with temporospatial constraints. 2013. 3

Matthew Gombolay, Anna Bair, Cindy Huang, and Julie Shah. Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences. *The International journal of robotics research*, 36(5-7):597–617, 2017. 3.2.1

E Gribovskaya, Florent d'Halluin, and Aude Billard. An active learning interface for bootstrapping robot's generalization abilities in learning from demonstration. In *RSS Workshop Towards Closing the Loop: Active Learning for Robotics*, volume 86, 2010. 3.1

Thomas L. Griffiths, Frederick Callaway, Michael Chang, Erin Grant, and Falk Lieder. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29:24–30, 2019. 2.1

LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021. URL `http://www.gurobi.com`. 3.3.1, 3.5

Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *arXiv preprint arXiv:2304.13653*, 2023. 5

Abhay Harpale and Yiming Yang. Active learning for multi-task adaptive filtering. In *ICML*, 2010. 2.2

Bradley Hayes and Brian Scassellati. Discovering task constraints through observation and active learning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4442–4449. IEEE, 2014. 3.1

Henry Hexmoor. A cognitive model of situated autonomy. In *Pacific Rim International Conference on Artificial Intelligence*, pages 325–334. Springer, 2000. 1.1

William W Hines, Douglas C Montgomery, and David M Goldman Connie M Borror. *Probability and statistics in engineering*. John Wiley & Sons, 2008. 4.3.4

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021. 3.1

Eric Horvitz. Artificial intelligence in the open world. *AAAI Presidential Address.(cited on page 11)*, 2008. 1.1

Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E Gil Jones. Contact-reactive grasping of objects with partial shape information. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1228–1235. IEEE, 2010. 4.1

Toshiyuki Inagaki et al. Adaptive automation: Sharing and trading of control. *Handbook of cognitive task design*, 8:147–169, 2003. 3.1

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029, 2019. doi: 10.1109/ICRA.2019.8794127. 5.1

David B Kaber, Jennifer M Riley, Kheng-Wooi Tan, and Mica R Endsley. On the design of adaptive automation for complex systems. *International Journal of Cognitive Ergonomics*, 5(1):37–57, 2001. 3.1

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013. 4.1

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 4.1

Sergey Karayev, Mario J Fritz, and Trevor Darrell. Dynamic feature selection for classification on a budget. In *International conference on machine learning (ICML): Workshop on prediction with sequential models*. Citeseer, 2013. 2.1

Ross A Knepper, Stefanie Tellex, Adrian Li, Nicholas Roy, and Daniela Rus. Recovering from failure by asking for help. *Autonomous Robots*, 39:347–362, 2015. 4.1

Andrey Kolobov. Planning with markov decision processes: An ai perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012. 3.2.1

George Konidaris and Andrew Barto. Efficient skill learning using abstraction selection. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009a. 3.3.2

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22:1015–1023, 2009b. 4.1, 4.2.1

George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research*, 31(3):360–375, 2012. ISSN 02783649. doi: 10.1177/0278364911428653. 4.2.1

George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Symbol acquisition for probabilistic high-level planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. 2.3

George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018. 2.3

Danica Kragic, Joakim Gustafson, Hakan Karaoguz, Patric Jensfelt, and Robert Krug. Interactive, collaborative robots: Challenges and opportunities. In *IJCAI*, pages 18–25, 2018. 3

Oliver Kroemer and Gaurav S Sukhatme. Learning spatial preconditions of manipulation skills using random forests. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 676–683. IEEE, 2016. 2.3, 3.3.2

Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *The Journal of Machine Learning Research*, 22(1):1395–1476, 2021. 3, 5

Nishanth Kumar, Tom Silver, Willie McClinton, Linfeng Zhao, Stephen Proulx, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Jennifer Barry. Practice makes perfect: Planning to learn skill parameter policies. *arXiv preprint arXiv:2402.15025*, 2024. 4.5.1

Andras Kupcsik, Marc Peter Deisenroth, Jan Peters, Ai Poh Loh, Prahlad Vadakkepat, and Gerhard Neumann. Data-Efficient Generalization of Robot Skills with Contextual Policy Search. *Artificial Intelligence*, 247:415–439, 2017. ISSN 00043702. doi: 10.1016/j.artint.2014.11.005. 3.1

Alex Lagrassa, Steven Lee, and Oliver Kroemer. Learning skills to patch plans based on inaccurate models. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9441–9448. IEEE, 2020. 5, 5.1

Michelle A Lee, Carlos Florensa, Jonathan Tremblay, Nathan Ratliff, Animesh Garg, Fabio Ramos, and Dieter Fox. Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7505–7512. IEEE, 2020. 5.1

Moonyoung Lee, Aaron Berger, Dominic Guri, Kevin Zhang, Lisa Coffey, George Kantor, and Oliver Kroemer. Towards autonomous crop monitoring: Inserting sensors in cluttered environments. *IEEE Robotics and Automation Letters*, 2024. 1.1

David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995. 1

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4:5, 2006. 3.3.2

Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. *arXiv preprint arXiv:1905.04753*, 2019. 2.1

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. *arXiv preprint arXiv:1509.02971*, 2015. 1.1, 5

Shuangqi Luo, Hongmin Wu, Shuangda Duan, Yijiong Lin, and Juan Rojas. Endowing Robots with Longer-term Autonomy by Recovering from External Disturbances in Manipulation Through Grounded Anomaly Classification and Recovery Policies. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 101(3), 2021. ISSN 15730409. doi: 10.1007/s10846-021-01312-6. 4.1

Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001. 5

Takeshi Matsuoka, Tsutomu Hasegawa, and Kyuhei Honda. A dexterous manipulation system with error detection and recovery by a multi-fingered robotic hand. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 1, pages 418–423. IEEE, 1999. 4.1

Stephen Miller, Jur Van Den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31(2):249–267, 2012. 1.1

Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7477–7484. IEEE, 2022. 5.1

Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2): 131–157, 2015. 4.1

Nvidia. Isaac sim, 2020. URL `https://developer.nvidia.com/isaac-gym`. 3.5.1

Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. POMDPs for Robotic Tasks with Mixed Observability. In *Robotics: Science and Systems V*. The MIT Press, 07 2010. ISBN 9780262289801. doi: 10.7551/mitpress/8727.003.0027. URL `https://doi.org/10.7551/mitpress/8727.003.0027`. 5.3

Adam Pacheck, George Konidaris, and Hadas Kress-Gazit. Automatic encoding and repair of reactive high-level tasks with learned abstract representations. In *Robotics Research: the 18th Annual Symposium*, 2019. 4.1

Priyam Parashar and Ashok K Goel. Meta-reasoning in assembly robots. *Systems Engineering and Artificial Intelligence*, pages 425–449, 2021. 4.1

Daehyung Park, Zackory Erickson, Tapomayukh Bhattacharjee, and Charles C Kemp. Multimodal execution monitoring for anomaly detection during robot manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 407–414. IEEE, 2016. 4.1

Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, and Stefan Schaal. Towards associative skill memories. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 309–315. IEEE, 2012. 4.1, 5.2

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021. 2.3

Margaret Pearce, Bilge Mutlu, Julie Shah, and Robert Radwin. Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes. *IEEE transactions on automation science and engineering*, 15(4):1772–1784, 2018. 1.1

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 4.4.1

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010. 3.5.1, 4.4

Richard W Pew. The speed-accuracy operating characteristic. *Acta Psychologica*, 30: 16–26, 1969. 3.1

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019. 5.4

Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020. ISSN 2573-5144. doi: 10.1146/annurev-control-100819-063206. 1.1, 3.1

Alfredo Reichlin, Giovanni Luca Marchetti, Hang Yin, Ali Ghadirzadeh, and Danica Kragic. Back to the manifold: Recovering from out-of-distribution states. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8660–8666. IEEE, 2022. 5.1, 5.3.2

Marc Rigter, Bruno Lacerda, and Nick Hawes. A Framework for Learning from Demonstration with Minimal Human Effort. *IEEE Robotics and Automation Letters*, 5(2):2023–2030, 2020. ISSN 23773766. doi: 10.1109/LRA.2020.2970619. 3.1, 3

Alberto Rodriguez, David Bourne, Mathew Mason, Gregory F Rossano, and JianJun Wang. Failure detection in assembly: Force signature analysis. In *2010 IEEE International Conference on Automation Science and Engineering*, pages 210–215. IEEE, 2010. 4.1

Eric Rosen, Ben M Abbatematteo, Skye Thompson, Tuluhan Akbulut, and George Konidaris. On the role of structure in manipulation skill learning. In *CoRL 2022 Workshop on Learning, Perception, and Abstraction for Long-Horizon Planning*, 2022. 5.1

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. 5.1

William B Rouse. Adaptive allocation of decision making responsibility between supervisor and computer. In *Monitoring behavior and supervisory control*, pages 295–306. Springer, 1976. 3.1

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. int. conf. on machine learning. 2001. 2

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395, 1991. 2.1

M. Scerbo. Theoretical perspectives on adaptive automation. 1996. 3.1

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5.4

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=H1aIuk-RW`. 2.2

Burr Settles. Active learning literature survey. 2009. 2.2

Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. *Advances in neural information processing systems*, 20, 2007. 4

Christopher J Shannon, Luke B Johnson, Kimberly F Jackson, and Jonathan P How. Adaptive mission planning for coupled human-robot teams. In *2016 American Control Conference (ACC)*, pages 6164–6169. IEEE, 2016. 3

Christopher J Shannon, David C Horney, Kimberly F Jackson, and Jonathan P How. Human-autonomy teaming using flexible human performance models: An initial pilot study. In *Advances in human factors in robots and unmanned systems*, pages 211–224. Springer, 2017. 3.1, 3.2.1

Mohit Sharma and Oliver Kroemer. Relational learning for skill preconditions. *arXiv preprint arXiv:2012.01693*, 2020. 3.3.2

Mohit Sharma, Jacky Liang, Jialiang Zhao, Alex LaGrassa, and Oliver Kroemer. Learning to compose hierarchical object-centric controllers for robotic manipulation. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 822–844. PMLR, 2020. URL `https://proceedings.mlr.press/v155/sharma21a.html`. 5.1

Thomas B Sheridan. Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: Distinctions and modes of adaptation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(4):662–667, 2011. 3.1

Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement Learning*, pages 387–414. Springer, 2012. 4.1

Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Jeffrey Ichnowski, Ellen Novoseller, Minho Hwang, Michael Laskey, Joseph E. Gonzalez, and Ken Goldberg. Untangling dense non-planar knots by learning manipulation features and recovery policies. In *Robotics Science and Systems (RSS)*, 2021. 4.1, 5

Yoonchang Sung, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning when to quit: meta-reasoning for motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4692–4699. IEEE, 2021. 2.1

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 5

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. 2.3

Justin Svegliato, Kyle Hollins Wray, and Shlomo Zilberstein. Meta-level control of anytime algorithms with online performance prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018. 2.1

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021. 4.1, 5.1, 5.3.2

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 4.4, 5

Kirill Trapeznikov and Venkatesh Saligrama. Supervised sequential classification under budget constraints. In *Artificial intelligence and statistics*, pages 581–589. PMLR, 2013. 2.1

Shivam Vats, Oliver Kroemer, and Maxim Likhachev. Synergistic scheduling of learning and allocation of tasks in human-robot teams. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2789–2795. IEEE, 2022. 1.3, 1

Shivam Vats, Maxim Likhachev, and Oliver Kroemer. Efficient recovery learning using model predictive meta-reasoning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7258–7264. IEEE, 2023. 1.3, 1, 5.3.3

Shivam Vats, Devesh K. Jha, Oliver Koremer, Maxim Likhachev, and Romeres Romeres. Recoverychaining: Learning local recovery policies for robust manipulation. Under review, 2024. 1.3

Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013. 2.2, 3.4

Federico Vicentini. Collaborative robotics: a survey. *Journal of Mechanical Design*, 143(4):040802, 2021. 3

Austin S. Wang and Oliver Kroemer. Learning robust manipulation strategies with multimodal state transition models and recovery heuristics. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:1309–1315, 2019. ISSN 10504729. doi: 10.1109/ICRA.2019.8793623. 4.1, 5

Joseph Wang, Kirill Trapeznikov, and Venkatesh Saligrama. An lp for sequential learning under budgets. In *Artificial intelligence and statistics*, pages 987–995. PMLR, 2014. 2.1

Yanwei Wang, Nadia Figueroa, Shen Li, Ankit Shah, and Julie Shah. Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 94–105. PMLR, 2022. URL `https://proceedings.mlr.press/v205/wang23a.html`. 5.1

Albert Wilcox, Ashwin Balakrishna, Brijen Thananjeyan, Joseph E Gonzalez, and Ken Goldberg. Ls3: Latent space safe sets for long-horizon visuomotor control of sparse reward iterative tasks. In *Conference on Robot Learning*, pages 959–969. PMLR, 2022. 4.1, 5.1, 5.3.2, 5.3.3

David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011. 3.4

Annie Xie, Fahim Tajwar, Archit Sharma, and Chelsea Finn. When to ask for help: Proactive interventions in autonomous reinforcement learning. In *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*. 3.1

Haoyu Xiong, Russell Mendonca, Kenneth Shaw, and Deepak Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024. 5.5.1

Ofer Yehuda, Avihu Dekel, Guy Hacohen, and Daphna Weinshall. Active learning through a covering lens. *Advances in Neural Information Processing Systems*, 35: 22354–22367, 2022. 2.2

Peter A Zachares, Michelle A Lee, Wenzhao Lian, and Jeannette Bohg. Interpreting contact interactions to overcome failure in robot assembly tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3410–3417. IEEE, 2021. 4.1

Wenxuan Zhou and David Held. Learning to grasp the ungraspable with emergent extrinsic dexterity. In *Conference on Robot Learning*, pages 150–160. PMLR, 2023. 5

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020. 4.4, 5.4.1

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73, 1996. 2.1