Abstract

Acting under uncertainty is a fundamental challenge for any decision maker in the real world. As uncertainty is often the culprit of failure, many prior works attempt to reduce the problem to one with a known state. However, this fails to account for a key property of acting under uncertainty: we can often gain utility while uncertain. This thesis presents methods that utilize this property in two domains: active information gathering and shared autonomy.

For active information gathering, we present a general framework for reducing uncertainty just enough to make a decision. To do so, we formulate the Decision Region Determination (DRD) problem, modelling how uncertainty impedes decision making. We present two methods for solving this problem, differing in their computational efficiency and performance bounds. We show that both satisfy adaptive submodularity, a natural diminishing returns property that imbues efficient greedy policies with near-optimality guarantees. Empirically, we show that our methods outperform those which reduce uncertainty without considering how it affects decision making.

For shared autonomy, we first show how the general problem of assisting with an unknown user goal can be modelled as one of acting under uncertainty. We then present our framework, based on Hindsight Optimization or QMDP, enabling us assist for a distribution of user goals by minimizing the expected cost. We evaluate our framework on real users, demonstrating that our method achieves goals faster, requires less user input, decreases user idling time, and results in fewer user-robot collisions than those which rely on predicting a single user goal. Finally, we extend our framework to learn how user behavior changes with assistance, and incorporate this model into cost minimization.
Acknowledgements

First and foremost, I would like to thank my advisors Siddhartha Srinivasa and Drew Bagnell. It seems like just yesterday when I bumped into you both talking about co-advising me. At the time, I knew our interests aligned - but I could not have guessed you would be such fantastic mentors, teachers, and friends. Thank you for being understanding and encouraging when I wanted to explore new ideas and concepts, academic and otherwise.

I am grateful to the other members of my committee, Emma Brunskill and Wolfram Burgard. Thank you for bringing your unique perspectives to this work, and providing new insights and ideas throughout.

I have learned so much from my collaborators throughout the years. Thank you to Andreas Krause, Yuxin Chen, and Amin Karbasi for your contributions for information gathering under uncertainty. Thank you to Matthew Klingensmith and Nancy Pollard for your insights from the ARM-S project and help with Touch-Based Localization. Thank you to Katharina Muelling, Arun Venkatraman, Jean-Sebastien Valois, John Downey, Jeffrey Weiss, Martial Hebert, Andrew B. Schwartz, and Jennifer L. Collinger for your work on brain computer interfaces for shared control teleoperation. Thank you to Henny Admoni and Stefania Pellegrinelli for your help on shared autonomy experiments for human-robot teaming and shared control teleoperation. Thank you to Anca Dragan and Rachel Holladay for your work on modelling comparison based learning as information gathering under uncertainty.

Being in both the Personal Robotics Lab and LAIRLab meant I had two sets of amazing labmates to learn with. Thank you to everyone in the Personal Robotics Lab: Anca, Chris, Mehmet, Alvaro, Jenn, Laura, Clint, Pras, Aaron, Pyry, Clint, Mike, other Mike, other other Mike, Brian, other Brian, Shushman, Gilwoo, Liz, Rosario, Ariana, Oren, Daqing, Aditya, and Jimmy for making the lab such a great environment. An extra special thanks to Michael Koval for all your help with the Personal Robotics Lab systems infrastructure, and Stefanos Nikolaidis for all your help with understanding how to run user studies. Thank you to everyone in the LAIRLab: Paul, Kris, Katherina, Stephane, Daniel, Alex, Kevin, Dey, Arun, Allie, Jiaji, Wen, Echo, Nick, and Shaurya for your weekly reading group discussions, talk feedback, and influx of ideas that were always helpful and interesting.

These last few years haven’t been all work - but have been filled with intense fun beyond what I could have imagined that has changed how I approach life and challenging problems. I’d like to thank Juan Pablo Mendoza and Ben Eckart for being such close friends, and joining for many (mis)adventures. Thank you to my partner Caitlin Rice for challenging my thoughts and beliefs, and making me feel loved and supported through the toughest times. Thank you to the Hot Mass crew in Pittsburgh for providing such an accepting space. My regular visits to this weekly Mass have been instrumental to keeping me feeling creative, rejuvenated, and open.

Special thanks to my sisters Shabnam and Shiva. Shabnam, you often joke that you’re responsible for everything good in my life - and you’re not (entirely) wrong. Shiva, thank you for being there whenever I
was down and needed some cheering up.

Finally, I would like to thank my parents Vida and Ahmad, whose love, support, and encouragement help me everyday. I am forever grateful for your personal sacrifices, leaving your family, friends, and home years ago so that my sisters and I could have better opportunities.
# Contents

1  **Introduction**  9  
   1.1  **Goal-Directed Active Information Gathering**  10  
   1.2  **Goal-Directed Shared Autonomy**  11  
   1.3  **Contributions**  13  

2  **Active Information Gathering Background**  15  
   2.1  **Active Information Gathering in Robotics**  15  
   2.2  **Near-Optimal Active Information Gathering**  18  
   2.3  **Adaptive Submodularity Background**  20  

3  **Hypothesis Pruning for Touch-Based Localization**  27  
   3.1  **Problem Formulation**  28  
   3.2  **Metrics for Touch-Based Localization**  30  
   3.3  **Experiments**  34  
   3.4  **Discussion**  38  

4  **Decision Region Determination (DRD)**  41  
   4.1  **Decision Region Determination (DRD) Problem Statement**  41  
   4.2  **The HyperEdge Cutting (HEC) Method**  44  
   4.3  **HyperEdge Cutting (HEC) Experiments**  50  
   4.4  **The Decision Region Edge Cutting (DiRECt) Method**  53  
   4.5  **Decision Region Edge Cutting (DiRECt) Experiments**  57  
   4.6  **Discussion**  60
A Appendix 125
   A.1 Hypothesis Pruning Proofs 125
   A.2 HyperEdge Cutting (HEC) Proofs 132
   A.3 Multi-Target MDPs 141
1 Introduction

Uncertainty presents a fundamental challenge for any decision maker acting in the real world. It is particularly problematic in robotics, where it accumulating from inaccurate models, noisy sensors, and poor calibration. These challenges have been studied in manipulation [LP+84; EM88; Gol93; Hsi+08; Kov+16], mobile robotics [Cas+96; Bur+97; Fox+98; Roy+05], aerial robotics [Cho+17a], underwater robotic inspection [Hol+13], and human-robot collaboration [MB13; DS13a; LS15; Sad+16b]. Acting under uncertainty has also been studied in machine learning [Das04; Bal+06; Now09; KG09; Kar+12], statistics [Lin56; Ber85; CV95], and decision theory [How66].

Decision making in these domains is often formulated as a Partially Observable Markov Decision Process (POMDP) [Kae+98]. This enables us to optimize some objective function under uncertainty, naturally trading off between information gathering and task accomplishing for the overall objective. However, finding optimal solutions to POMDPs is PSPACE complete [PT87]. Although several promising approximate solvers have been developed [Roy+05; SS05; Kur+08; SV10; Sha+12; Som+13; Sei+15], they remain intractable in many real world settings.

For situations where POMDP solvers are not practical, many approximations have been proposed [Roy+05; Ros+08; DS13a; Heb+13; KS13]. As uncertainty is often the culprit of failure, a common strategy is to reach a known state with high probability, and then gain utility for that state [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Hsi+08; Heb+13; DS13a; KS13; LS15]. However, this strategy overlooks a key property of POMDP solutions: not all uncertainty impedes gaining utility. Even when uncertainty is high, there often exist actions which gain utility over the entire distribution. Thus, relying on a known state leads to suboptimal policies.

In this thesis, we formulate policies that do not rely on reducing uncertainty to a known state to gain utility. We formulate policies with this property in two domains: active information gathering and
human-robot collaboration. For active information gathering, we study problems where we are initially too uncertain to accomplish a task (fig. 1.1). However, we need not reduce uncertainty entirely to make a decision. While prior works rely on optimizing for uncertainty reduction [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b], we formulate an objective for gathering just enough information to make a decision. For human-robot collaboration, we study instances where the system must simultaneously predict a user’s goal while achieving a shared goal. We term this instance shared autonomy (fig. 1.2). While prior methods rely on predicting the user’s goal before acting [Yu+05; Kof+05; CD12; DS13a; KS13; Mue+15], we develop a method that enables progress for a distribution over user goals when it is possible. More succinctly:

*Not all uncertainty is problematic — this thesis formulates efficient policies for gaining utility under uncertainty in active information gathering and shared autonomy.*

We call methods that need not reduce uncertainty entirely goal-directed, as they deal with uncertainty only as required for achieving a goal.

### 1.1 Goal-Directed Active Information Gathering

In many situations, we may not be able to accomplish our task until some uncertainty is resolved. For example, in medical diagnosis, we may need to run tests about the state of a patient to determine proper treatment [Kon01]. In object search, we may require finding a particular object required for the task [Won+13; Li+16]. In search and rescue, we are tasked with quickly finding a target [Lim+15; Lim+16]. In biological conservation, we need to decide among expensive monitoring strategies to decide a strategy for conservation [Run+11]. In tasks of fine-manipulation, we must often accurately localize a target object to achieve our goal [Hsi+08; Heb+13].

In these situations, we are interested in gathering the required information as quickly as possible [Cas+96; Hsi+08; Run+11; Won+13; Heb+13]. To do so, these methods pick some measure of uncertainty, such as the Shannon entropy, and optimize for tests to reduce this measure. Once this measure reaches a target threshold, information gathering terminates, and the system tries to accomplish the task.

Ideally, the measure being optimized for considers uncertainty tolerance inherent in the task. Unfortunately, the most commonly used metric, the reduction of Shannon entropy [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b], does not have this property. Optimizing for this metric amounts...
to reducing uncertainty *indiscriminately*, without considering how uncertainty impedes gaining utility. Information gathering for the purpose of gaining utility is captured by the decision-theoretic Value of Information (VoI) [How66]. Unfortunately, optimally selecting tests for this measure is NP-complete [KG09].

Optimizing for many natural metrics related to uncertainty, such as the reduction of Fisher Information [Hoi+06] and reduction of Shannon Entropy [KG05], can be done efficiently while providing performance guarantees. This is done by showing these metrics are *submodular*, a natural diminishing returns property that renders greedy algorithms near-optimal [Nem+78; Wol82]. These guarantees hold in the open-loop setting, where a set of actions are chosen apriori in expectation over observations.

Newer notions of *adaptive submodularity* [GK11] extend these bounds to the adaptive setting, where action selection can depend on observations received. We review this property, and how it differs from submodularity, in section 2.3. A method similar to reducing the Shannon entropy, known as Generalized Binary Search (GBS) [Das04; Now08; Now09], uses this property to provide for indiscriminate uncertainty reduction while performing near-optimally [KB09; GB09; GK11]. Our first contribution is showing how a similar method can be applied to active information gathering in robotics to provide guarantees1 (chapter 3).

We extend this work to consider how uncertainty impedes decision making in chapter 4. We term this the *Decision Region Determination* (DRD) problem, with the goal of reducing uncertainty just enough to make a decision. See fig. 1.3 for an illustration. We present two methods for solving this problem, differing in their computational efficiency and performance bounds2–3. We show both are *adaptive sub modular*, enabling us to guarantee near-optimal performance with an efficient greedy algorithm. Furthermore, it is known that achieving a much better approximation for adaptive submodular maximization is NP-hard [GK11]. Thus, we believe our performance exceeds that of a general POMDP solver which does not utilize this property.

We apply this general framework to touch-based localization in robotic manipulation, wildlife conservation management, movie recommendation, and Behavioral economics in section 4.54.


4 In other works, we have also implemented this method for user preference learning [Hol+16] and motion planning [Cho+17b].

1.2 Goal-Directed Shared Autonomy

Human-robot collaboration studies interactions between humans and robots sharing a workspace. One instance of collaboration arises in *shared autonomy*, where both the user and robotic system act-
multaneously to achieve shared goals. For example, in *shared control teleoperation* [Goe63; Rosg3; AMg7; Deb+00; DS13a], both the user and system control a single entity, the robot, in order to achieve the user’s goal. In *human-robot teaming*, the user and system act independently to achieve a set of related goals [HB07; Ara+10; DS13b; KS13; MB13; Gom+14; Nik+17b].

While each instance of shared autonomy has many unique requirements, they share a key common challenge - for the autonomous system to be an effective collaborator, it needs to know the user’s goal. For example, feeding with shared control teleoperation, an important task for assistive robotics [Chu+13], requires knowing what the user wants to eat (fig. 1.4). Wrapping gifts with a human-robot team requires knowing which gift the user will wrap to avoid getting in their way and hogging shared resources (fig. 1.5).

In general, the system does not know the user’s goal a priori. We could alleviate this issue by requiring users to explicitly specify their goals (e.g. through voice commands). However, there are often a continuum of goals to choose from (e.g. location to place an object, size to cut a bite of food), making it impossible for users to precisely specify their goals. Furthermore, prior works suggest requiring explicit communication leads to ineffective collaboration [Van+03; GJ03; Gre+07]. Instead, implicit information should be used to make collaboration seamless. In shared autonomy, this suggests utilizing sensing of the environment and user actions to infer the user’s goal. This idea has been successfully applied for shared control teleoperation [LO03; Yu+05; Kra+05; Kof+05; AKo8; CD12; DS13a; Hau13; Mue+15] and human-robot teaming [HB07; Ngu+11; Mac+12; MB13; KS13; LS15].

Most shared autonomy methods do not assist when the goal is unknown. These works split shared autonomy into two parts: 1) predict the user’s goal with high probability, and 2) assist for that single goal, potentially using prediction confidence to regulate assistance. We refer to this approach as *predict-then-act*. While this has been effective in simple scenarios with few goals [Yu+05; Kof+05; CD12; DS13a; KS13; Mue+15], it is often impossible to predict the user’s goal until the end of execution (e.g. fig. 1.2), causing these methods to provide little assistance. Addressing this lack of assistance is of great practical importance - in our feeding experiment (section 7.2.1), a predict-then-act method provided assistance for only 31% of the time on average, taking 29.4 seconds on average before the confidence threshold was initially reached.

Instead, we would prefer a method that takes actions to assist the user even when uncertainty is present. While we may not be able to achieve a user’s goal without predicting it, we can often make progress towards multiple goals even when uncertain. To do so,
we frame shared autonomy as a general problem of minimizing the expected user cost.

While minimizing this quantity exactly is intractable, Hindsight Optimization [Cho+00; Yoo+08], or QMDP [Lit+95] approximates this solution and produces our desired behavior. These methods have been empirically successful in many domains of acting under uncertainty. We show how the general problem of shared autonomy can be modelled in this framework in chapter 6. Our user studies in chapter 7 demonstrate that our method outperforms predict-then-act approaches, enabling users to accomplish their goal faster and with less effort. Finally, we show how to construct and utilize better models that incorporate how user behavior changes as a system provides assistance in chapter 8.

Our method enables optimization in continuous action spaces, a notoriously difficult problem for POMDP solvers. While recent POMDP solvers provide approximate solutions in this domain [Sei+15], they remain computationally slow, taking multiple seconds to select an action. Unfortunately, this rate is too slow for shared autonomy, as this makes systems feel unresponsive. Our system operates at 50Hz, enabling shared-control teleoperation to feel fast and responsive while providing assistance for a distribution over goals.

1.3 Contributions

This thesis studies computationally efficient methods for dealing with uncertainty for active information gathering and shared autonomy. Compared to previous works, we incorporate the insight that goal-directed progress can be made computationally efficiently even when uncertainty is high. We make the following contributions:

• A connection between active information gathering in robotics and submodularity with application to touch-based localization (chapter 3, Jav+13).

• Provably near-optimal method for goal-directed information gathering under uncertainty. We provide both theoretical analysis and experimental evidence that these frameworks outperform approaches that reduce uncertainty indiscriminately (chapter 4, Jav+14; Che+15).

• A model for shared autonomy as acting under uncertainty, enabling us to make progress for a user’s goal even when uncertain what the goal is (chapter 6, Jav+15).

• User studies of our shared autonomy framework for both shared-control teleoperation and human-robot teaming, demonstrating


that acting over the entire distribution enables faster performance with less user effort (chapter 7, Jav+15; Pel+16).

• An extension of our shared autonomy framework learning how users respond to assistance, and applying this model to minimize their cost (chapter 8).
Active Information Gathering Background

Active information gathering methods attempt to quickly reduce the uncertainty by intelligently selecting information gathering actions. This general problem arises in many domains, such as automated medical diagnosis [Kon01], policy making [Run+11], active learning [Das04; Bal+06; LZ14], active perception [Baj88], decision theory [How66], automated data collection [Hol+12], robotic active localization [Fox+98; KR08; Hsi+08; Heb+13], interactive learning [H+08; VB10; GB11; Für+12; Kar+12; Sad+17] and more. We review background material most relevant to this thesis here, though there are many more works for this general problem.

2.1 Active Information Gathering in Robotics

Active information gathering has been studied in many robotics domains, such as manipulation [Hsi+08; Heb+13; Kov+16], mobile robotics [Cas+96; Bur+97; Fox+98; Roy+05], aerial robotics [Cho+17a], robotic inspection [Hol+13], policy learning [VB10; Akr+12; Akr+14; Dan+14], and human-robot collaboration [Sad+16a; Sad+17].

Many previous works on active information gathering utilize online planning within the POMDP framework [Ros+08], looking at locally reachable states during action selection. In general, these methods limit the search to a low horizon [Hsi09], often using the greedy strategy of selecting actions with the highest expected benefit in one step [Cas+96; Bur+97; Fox+98; Hsi+08; Heb+13]. After searching over this horizon, they apply some metric for the value of the resulting belief (e.g., information gained), and propagate that information through the POMDP to select the current action. This is out of necessity - computational time increases exponentially with the search depth. However, this simple greedy strategy often works surprisingly well, often even providing performance guarantees1.

Perhaps the most commonly used metric for information gathering is the expected decrease in Shannon entropy [Cas+96; Bur+97;
This is referred to as the information gain metric, and is submodular under certain assumptions [KGo5]. This property renders a greedy open-loop plan near-optimal\(^2\). Not surprisingly, many robotic systems which perform well with a low horizon use this metric [Cas+96; Bur+97; Fox+98; Bou+02; Hsi+08; Heb+13; Sad+16b], though most do not make the connection with submodularity\(^3\).

A common use case for active information gathering is viewpoint selection, where the system decides the location and direction of a sensor (e.g. camera, laser range, radar) to gather information [Baj88]. Burgard et al. [Bur+97] and Fox et al. [Fox+98] use the information gain metric for viewpoint selection of a mobile robot using laser range sensors. Bourgault et al. [Bou+02] incorporate this metric into the Simultaneous Localization and Mapping (SLAM) [LDW91] framework. Roy et al. [Roy+05] derive policies in this domain utilizing a full POMDP and belief compression. Kim and Likhachev [KL16] design an adaptive submodular framework similar to ours (chapter 3) for selecting viewpoints to gather information about partially occluded objects for grasping. For the related domain of robotic inspection, where a robot must decide where to examine a target to gather information about it, Hollinger et al. [Hol+13] explore active information gathering for constructing 3D meshes.

Similar ideas have been explored for information gathering in clutter, where the robot can both select a viewpoint and move objects around. Wong et al. [Won+13] present an algorithm for intelligently searching for a target object, utilizing priors about object constraints and object-object co-occurrences to gather information. Their algorithm considers searching in different containers, as well as moving occluding objects around, to gather information. Li et al. [Li+16] construct a policy for finding an object in clutter by extending DESPOT [Som+13] to incorporate task constraints. They show this outperforms greedy approaches in their setting.

An interesting avenue of recent work attempts to learn information gathering policies directly. Choudhury et al. [Cho+17a] do so through imitation learning [Pom89; Ros+11]. They generate information gathering policies offline, which can be non-greedy and have access to the full world state (referred to as the clairvoyant policy). They then train a policy with access only to partial observability, which will be the case during online execution, to imitate the clairvoyant policy.

Active methods have also been used to quickly learn policies that incorporate user preferences. These methods produce a query to present users (e.g. a pair of trajectories) and have them provide feedback about their preference (e.g. preferred trajectory). Wilson et al.\(^4\) However, most methods mentioned here apply it in the adaptive setting, where the policy changed based on observations. The guarantees of submodularity do not hold in this setting. We discuss the differences and related properties for the adaptive setting in section 2.3.

\(^1\) Hsiao [Hsi09] mentions that touch-based localization could be formulated as a submodular maximization.
[Wil+12] formulate a method for finding a parameterized policy by showing users short pairs of trajectories, and asking which is better. They present two active methods for selecting queries, and show that both require fewer rounds of feedback than randomly generated queries. Viappiani and Boutilier [VB10] formulate a criteria called the Expected Utility of Selection (EUS), similar to the information-theoretic Expected Value of Information (EVOI) while being computationally more efficient. They use this criteria to select informative choice queries, where a user chooses one item from a set. Akrour et al. [Akr+12] extend their method to learn a linear reward function in a continuous space by demonstrating a trajectory, and having a user rank it relative the highest-ranked trajectory so far. They show this outperforms random query selection. Akrour et al. [Akr+14] later extended this work to incorporate noisy user responses. Daniel et al. [Dan+14] formulate a method for active information within the framework of relative entropy policy search (REPS) [Pet+10]. While most of the aforementioned methods focus on having users rank trajectories, Daniel et al. [Dan+14] have users provide numerical values, which they argue provides more information than just a preference. Instead of optimizing over a set of predetermined queries, Sadigh et al. [Sad+17] actively synthesize trajectory pairs to show users to learn a reward function for autonomous driving.

2.1.1 Touch-Based Localization

A central problem considered in this thesis (chapters 3 and 4) is touch-based localization, where a robot uses its manipulator to localize itself or an object. Our work was motivated by promising results in the DARPA Autonomous Robotic Manipulation Software (ARM-S) challenge, where teams were required to localize and manipulate objects within a time limit. Prior to attempting the task, most teams relied on gathering information through a hand-coded sequence of touches. Early works in this domain focused on finding an open-loop sequence of actions to localize an object, potentially even without sensors. Lozano-Pérez et al. [LP+84] address the classic peg-in-hole problem by finding a sequence of complaint motions that handle uncertainty. Erdmann and Mason [EM88] explore motion strategies to localize a planar object by tilting a tray. Goldberg [Gol93] find a fixed sequence of parallel-jaw gripper actions to orient a polygonal part. When it is possible to compute a fixed sequence of actions offline, these methods are very promising, enabling object localization with minimal online computation.

4 DARPA Autonomous Robotic Manipulation (ARM) - Phase 1 video
5 HERB Prepares a Meal video
Theoretical analysis of the adaptivity gap (section 2.3.4), the difference in performance of the optimal adaptive policy compared to an open-loop plan, show that open-loop methods may require exponentially more actions than an adaptive policy to acquire the same information [GK11; Hol+11]. More recent works incorporate the sensing and action history into action selection to form adaptive policies [LH98; Hsi+08; Hsi09; Heb+13].

Many works in this domain utilize guarded moves [WG75], where a trajectory terminates when contact is made with any object. This gives us information about the location of a face of the object (where contact was made), as well as space free of objects (where no contact was made). Petrovskaya and Khatib [PK11] show that, with their well designed particle filter, randomly chosen guarded moves were able to localize a target object to within \( \sim 5 \text{mm} \) in a full 6DOF space. However, they required an average of 29 actions, which subsequent works reduce significantly with more intelligent action selection.

Also motivated by promising results in the DARPA ARM-S challenge, Hebert et al. [Heb+13] present a method for greedily selecting a touch sensing action. They select tests that maximize the one-step reduction of Shannon entropy like many works in other domains [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b].

Hsiao et al. [Hsi+08; Hsi09] select a sequence of information gathering tactile actions through forward search in a POMDP. Possible actions consist of pre-specified world-relative trajectories [Hsi+08], motions based on the current highest probability state. Actions are selected by maximizing one of two metrics: either the reduction of Shannon entropy, which indiscriminately reduces uncertainty about all hypotheses (our aim in chapter 3), or a decision-driven approach of maximizing the probability of grasp success\(^6\) (our aim in chapter 4). Not surprisingly, the decision-driven approach enables success with fewer information gathering actions [Hsi09], a result we also find in our experiments (sections 4.3 and 4.5).

Other’s have exploited the structure of contact sensing to utilize POMDPs for touch based localization. Erez and Smart [ES10] utilize a Gaussian belief space with local controllers, modelling contacts as constraints, to find policies that utilize contact to reduce uncertainty. Koval et al. [Kov+16] decompose policies into pre- and post- contact states to efficiently solve POMDPs for planar contacts.

2.2 Near-Optimal Active Information Gathering

Active information gathering, especially in discrete settings, has been studied very generally in machine learning [Das04; Bal+06; Now09;
The problem is formulated as sequentially selecting tests to reduce uncertainty about a set of hypotheses.

In many cases, the goal of active information gathering is to find the optimal sequence of tests, which have the minimum cost (in expectation) while achieving some objective (e.g. amount of uncertainty reduced). This can be modelled as a Partially Observable Markov Decision Process (POMDP) [SS73; Kae+98]. Unfortunately, as the state, action, and observation spaces are often large, the application of many black-box POMDP solvers (e.g., [Pin+06; Kur+08; Ros+08; SV10; Som+13]) are rendered infeasible.

While deriving the optimal policy even in simplified domains is NP-hard [Cha+07], computationally efficient methods with approximation results are known in some settings. Surprisingly, many methods with bounded near-optimal performance rely on greedy algorithms, which only look one step ahead when selecting each test.

One often studied case is when the objective is to find the true hypothesis. If tests are noise-free (i.e., deterministic functions of the hidden state), the problem is known as the Optimal Decision Tree (ODT) problem, and a simple greedy algorithm, called Generalized Binary Search (GBS) [Das04; Nowo8; Nowo9], performs near-optimally [KB99; GB09; GK11].

Extensions to the ODT problem, with similar bounds and methods, have been studied extensively. One line of work examines methods for noisy test outcomes. For independent tests with persistent noise, where the same test will produce the same noisy outcome, an algorithm known as Equivalence Class Edge Cutting (EC2) [Gol+10] performs near-optimally. For correlated tests with persistent noise, the Equivalence Class Edge Discounting (ECED) [Che+17] performs near-optimally.

A different line of work removes the assumption that the cost of tests is fixed, and attempts to find informative paths. Here, the cost of a test is related to the distance we would travel to some sensing location. For certain settings when the adaptivity gap (section 2.3.4) is known to be small, the Nonmyopic, Adaptive, Informative (NAIVE) [Sin+09] algorithm produces near-optimal results with a non-adaptive policy. For more general instances, the Recursive Adaptive Identification (RAId) Lim et al. [Lim+16] algorithm produces near-optimal paths. Lim et al. [Lim+15] later extend this work to incorporate noisy observations. While these works offer promising results and analyses, they are computationally infeasible for many real-world problems.

In some cases, the decisions (e.g. medical treatments) differ from the hypotheses (e.g. diseases). Many of the aforementioned algo-
rithms reduce uncertainty indiscriminately, without considering how it affects the task. Often, we only need to reduce uncertainty enough to make a decision. This is captured most generally by the decision-theoretic Value of Information (VoI) [How66]. Optimizing this criterion in general probabilistic models is NP complete [KG09].

For cases when each hypothesis corresponds to only one decision, the EC² algorithm above can be used to provide near-optimal test selection [Gol+10]. We extend this model to allow for the more general case where there are multiple valid decisions for each hypothesis (e.g. there are many ways to grasp an object) in chapter 4.

Many of these methods provide their guarantees by showing they correspond to an adaptive submodular maximization [GK11], which we discuss in section 2.3.

2.3 Adaptive Submodularity Background

In order to provide theoretical guarantees for an efficient lazy-greedy policy, this thesis casts problems of active information gathering into adaptive submodular maximizations. We briefly review this property here, and the results derived by Golovin and Krause [GK11].

We assume a known prior distribution over hypotheses $h \in \mathcal{H}$, given by $P(h)$. Each hypothesis represents a possible state of the world. We gather information by running tests $t \in \mathcal{T}$, each of which has a known cost $c(t)$. Upon running a test, we observe an outcome $o \in \mathcal{O}$, which is deterministic given a hypothesis $h$. Thus, each hypothesis $h \in \mathcal{H}$ can be considered a function $h : \mathcal{T} \rightarrow \mathcal{O}$ mapping tests to outcomes. We assume there exists a true hypothesis $h^* \in \mathcal{H}$, which will be consistent with all observed outcomes.

Suppose we have executed a set of tests $T = \{t_1, \ldots, t_m\} \subseteq \mathcal{T}$ (e.g., medical tests, items shown to the user, moves made by the robot), and have observed their outcomes $h^*(t_1), \ldots, h^*(t_m)$. Our evidence so far is captured by the set of test-outcome pairs, $S \subseteq \mathcal{T} \times \mathcal{O}$, where $S = \{(t_1, h^*(t_1)), \ldots, (t_m, h^*(t_m))\}$. We denote the tests in $S$ as $S_T$, and the outcomes as $S_O$.

Upon observing $S$, we can rule out inconsistent hypotheses, and update the distribution $P(h \mid S)$. We denote the resulting set of hypotheses as the version space given $S$:

$$\mathcal{V}(S) = \{h \in \mathcal{H} : \forall (t, o) \in S, h(t) = o\}$$

This quantity is similar to the notion of belief states in Partially Observable Mark Decision Processes (POMDPs) [Kae+98] in that it captures our distribution over the world given the evidence so far.
### Table 2.1: Variables used for adaptive submodular functions.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h \in \mathcal{H}$</td>
<td>Hypothesis, e.g. possible state of the world</td>
</tr>
<tr>
<td>$t \in \mathcal{T}$</td>
<td>Test, information gathering cation</td>
</tr>
<tr>
<td>$o \in \mathcal{O}$</td>
<td>Observation, outcome of a test</td>
</tr>
<tr>
<td>$f$</td>
<td>Objective function</td>
</tr>
<tr>
<td>$c$</td>
<td>Cost function, defined for each test</td>
</tr>
<tr>
<td>$S \subseteq \mathcal{T} \times \mathcal{O}$</td>
<td>Evidence so far, captured by test-outcome pairs</td>
</tr>
<tr>
<td>$\mathcal{V}(S) \subseteq \mathcal{H}$</td>
<td>Version space, hypotheses remaining given evidence $S$</td>
</tr>
<tr>
<td>$\pi$</td>
<td>Policy, which maps evidence $S$ to a test to run</td>
</tr>
<tr>
<td>$S(\pi, h)$</td>
<td>Evidence from running $\pi$ if $h$ generated outcomes</td>
</tr>
</tbody>
</table>

### 2.3.1 Problems

We define an objective function over the evidence so far $f(S)$, which we wish to maximize. To do so, we run tests, for which we pay an additive cost $C(S) = \sum_{t \in \mathcal{T}} c(t)$.

Our goal is to find a policy $\pi$ for running tests given the evidence so far. We generally would like for this policy to maximize our objective while minimizing the cost (e.g. gather enough information while minimizing cost). Let $S(\pi, h)$ be the evidence we would gather by running policy $\pi$ if hypothesis $h \in \mathcal{H}$ generated outcomes. Define $C(\pi)$ as the average-case cost of running $\pi$ over $\mathcal{H}$, $C(\pi) = \mathbb{E}_{\mathcal{H}}[C(S(\pi, h))]$.

We define two different problems we may want to optimize a policy for.

**Problem 1** (Adaptive Stochastic Minimum Cost Cover). Let $Q$ be some quota of the objective function we wish to obtain (e.g. gather enough information to make a decision). We seek a policy that obtains this quota for any hypothesis $h \in \mathcal{H}$ while minimizing the expected cost:

$$
\pi^* = \arg \min_{\pi} C(\pi) \text{ s.t. } f(S(\pi, h)) \geq Q \quad \forall h \in \mathcal{H} \tag{2.2}
$$

We can also consider the worst-case cost, $C_{wc}(\pi) = \max_h C(S(\pi, h))$.

For adaptive submodular problems, greedy policy provides guarantees for both.

For this thesis, we formulate our active information gathering problems as ones defined by problem 1. However, it turns out that the same greedy algorithm provides guarantees for other problems of interest as well$^8$.

**Problem 2** (Adaptive Stochastic Maximization). Let $B$ be some budget on the total cost of tests we can run. We seek a policy that maximizes our objective function $f$ (in expectation) subject this this budget constraint:

$$
\pi^* = \arg \max_{\pi} \mathbb{E}_{\mathcal{H}}[f(S(\pi, h))] \text{ s.t. } C(S(\pi, h)) \leq B \quad \forall h \in \mathcal{H} \tag{2.3}
$$

---

$^8$In addition to the problems defined here, we can also provide guarantees for the sum at each time step. This is known as the Adaptive Stochastic Min-Sum Cover problem. The definition and bounds are provided by Golovin and Krause [GK11]
It turns out that the same greedy policy provides guarantees for all of these if the function satisfies \textit{adaptive submodularity} and \textit{adaptive monotonicity}.

### 2.3.2 Submodularity

First, let us consider the case when we do not condition on outcomes, optimizing for an open-loop plan. For this section, the objective is defined only over tests, and not their outcomes. Let $T \subseteq \mathcal{T}$ be a set of tests. We define the marginal utility as:

$$\Delta_f(t \mid T) = f(T \cup \{t\}) - f(T)$$  \hspace{1cm} (2.4)

\textbf{Submodularity} [Nem+78] (diminishing returns): A function $f$ is submodular if whenever $T \subseteq T' \subseteq \mathcal{T}$, $t \in \mathcal{T} \setminus T'$:

$$\Delta_f(t \mid T) \geq \Delta_f(t \mid T')$$

That is, the benefit of $t$ to the smaller set $T$ is at least as much as adding it to the superset $T'$.

\textbf{Monotonicity} (more never hurts): A function $f$ is monotone if the marginal utility is always positive:

$$\Delta_f(t \mid T) \geq 0 \quad \forall T, t \notin T$$

The greedy algorithm maximizes $\frac{\Delta_f(t \mid T)}{c(t)}$, the marginal utility per unit cost. As outcomes are not incorporated, this corresponds to an open-loop plan.

If submodularity and monotonicity are satisfied, the greedy algorithm will be within a $(1 + \ln \max_t f(t))$ factor of the optimal solution to \textbf{problem 1} for integer valued $f$ [Wol82], and $(1 - \frac{1}{e})$ of the optimal solution to \textbf{problem 2} [Nem+78].

It turns out that many natural notions of information, such as the reduction of Fisher Information [Hoi+06] and reduction of Shannon Entropy [KG05], are submodular. Natural applications of submodular maximization arise in many problems, such as physical sensing optimization problems, where the goal is to find the best locations to place sensors [Mut+07], document summarization [LB11], optimization of control libraries [Dey+12a; Dey+12b], social network analysis [Les+07], and many more.

### 2.3.3 Adaptive Submodularity

The guarantees for submodular maximization only hold in the \textit{non-adaptive setting}, corresponding to an open-loop plan. Golovin and Krause [GK11] extended notions of submodularity and their corresponding bounds to the \textit{adaptive setting}, where test selection can
depend on past observations. In this setting, the expected marginal benefit of performing an action is:

\[ \Delta_f(t \mid S) = \sum_h P(h \mid S) [f(S \cup \{(t, h(t))\}) - f(S)] \]  

(2.5)

**Adaptive Submodularity** (diminishing returns in expectation): A function \( f \) is adaptive submodular if whenever \( S \subseteq S' \subseteq T \times O \), \( t \in T \setminus S' \):

\[ \Delta_f(t \mid S) \geq \Delta_f(t \mid S') \]

That is, the expected benefit of adding \( t \) to a smaller set of evidence \( S \) is at least as much as adding it to the superset \( S' \).

**Adaptive Monotonicity** (more never hurts in expectation): A function \( f \) is adaptive monotone if the expected marginal utility is always positive:

\[ \Delta_f(t \mid S) \geq 0 \quad \forall S, t \notin S_T \]

**Strong Adaptive Monotonicity** (more never hurts): A function \( f \) is strongly adaptive monotone if it increases for any outcome we might observe:

\[ f(S \cup \{(t, h(t))\}) - f(S) \geq 0 \quad \forall h, t \notin S_T \]

Similar to the submodular setting, the greedy algorithm maximizes \( \frac{\Delta_f(t \mid S)}{\varepsilon(t)} \). We refer to the policy that greedily maximizes this quantity as \( \pi^g \).

**Theorem 1** (Adaptive Stochastic Minimum Cost Cover [GK11]). Let \( f \) be an adaptive submodular, strongly adaptive monotone, and self-certifying function. Let \( \eta \) be any value such that \( f(S) > Q - \eta \) implies \( f(S) = Q \). Let \( p_{\min} \) be the minimum prior probability of any hypothesis, \( p_{\min} = \min_h P(h) \). Let \( \pi^* \) be any policy, (e.g. the optimal policy for problem 1). The expected cost of the greedy policy is bounded by:

\[ C(\pi^g) \leq C(\pi^*) \left( \ln \frac{Q}{\eta} + 1 \right) \]

And the worst case cost is bounded by:

\[ C_{wc}(\pi^g) \leq C_{wc}(\pi^*) \left( \ln \frac{Q}{\delta \eta} + 1 \right) \]

**Theorem 2** (Adaptive Stochastic Maximization [GK11]). Let \( f \) be an adaptive submodular and adaptive monotone function. Let \( \pi^* \) be any policy (e.g. the optimal policy for problem 2). The expected objective of the greedy policy is bounded by:

\[ f(S(\pi^g, h)) > \left( 1 - \frac{1}{e} \right) f(S(\pi^*, h)) \]
Theorem 1 and theorem 2 generalize the bounds for submodularity [Nem+78; Wol82] to the adaptive setting. Proofs are provided by Golovin and Krause [GK11]. Functions which naturally exhibit these properties arise in active learning settings, where the reduction of version space probability mass [Das04; Now08; GK11], and variants for noisy tests and outcomes [Gol+10; Che+17], are adaptive submodular. In addition to our work which utilizes this property to provide near-optimality guarantees in robotics (chapters 3 and 4) and user preference learning [Hol+11], it has been used by others for selecting viewpoints for partially occluded objects [KL16], and for autonomous driving [Sad+17]. Furthermore, adaptive submodularity enables the use of a lazy-greedy method [Min78; GK11], where we can skip the reevaluation of some tests13.

In this thesis, we utilize these proofs to provide bounds for our information gathering methods, either to discover the true state of the world (chapter 3) or for gathering enough information to make a decision (chapter 4).

### 2.3.4 Adaptivity Gap

We have mentioned that submodular functions provide guarantees for open-loop plans, which do not condition on outcomes, while adaptive submodular functions provide guarantees for adaptive policies. Numerous works have investigated the adaptivity gap for maximizing these functions, which is the difference in performance of the optimal adaptive policy as compared to the optimal open-loop plan.

For problem 1, Golovin and Krause [GK11] and Hollinger et al. [Hol+11] show that the adaptivity gap is exponential in the number of tests, even for adaptive submodular functions. As we formulate our information gathering problems (chapters 3 and 4) in this form, we implement adaptive policies.

However, for problem 2, the adaptivity gap can be much smaller. While a general adaptivity gap is not known at this time, it has been studied for special cases, such as for set cover [GV06], or the probing problem [AN16; Gup+17]. Depending on the particular problem, the adaptivity gap can range from \( \frac{1}{2} \) [AN16], to 3 [Gup+17], to a function of the target [GV06], to numerous other values. Nonetheless, if the application can be formulated as in problem 2, the adaptivity may be small. This has been used by Hollinger et al. [Hol+13] to provide bounds for an open-loop plan compared to the optimal policy for underwater robotic inspection.

13 Suppose we have the cost-normalized marginal benefit for some test \( t \), and it is greater than the previously computed cost-normalized marginal benefit for another test \( t' \). Due to adaptive submodularity, we know the benefit of \( t' \) could not have increased, and thus can skip it’s reevaluation.
2.3.5 Interactive Submodularity

Similar to adaptive submodularity, Guillory and Bilmes [GB10] define interactive submodularity for bounding the performance of greedy adaptive policies for problem 1. However, this framework only provides guarantees for the worst-case performance, whereas adaptive submodularity provides bounds for the average-case and worst-case performance.

While the average-case bound provided by adaptive submodularity makes it more appealing, interactive submodularity is generally easier to show. It only requires pointwise-submodularity, where the function is submodular for any fixed hypothesis $h$. That is, $f(T, h)$ is submodular for every fixed $h$.

\[^{14}\text{Guillory and Bilmes [GB11] extend these results to the case where } h^* \notin \mathcal{H}\]

\[^{15}\text{In particular, adaptive submodular functions require that the returns are diminishing for any observed outcome, and subsequent update to the distribution } P(h \mid S). \text{ In practice, showing that the expected marginal utility decreases for any possible update to this distribution can be tricky.}\]
Hypothesis Pruning for Touch-Based Localization

In this chapter, we draw a connection between touch-based localization and (adaptive) submodularity (section 2.3), a natural diminishing returns property that renders a greedy algorithm near-optimal. We are motivated by the wide application and success of works in robotics which use the reduction of Shannon entropy, known as the information gain, for active information gathering [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b]. This metric is submodular under certain assumptions [KG05].

The guarantees for submodular maximization only hold in the non-adaptive setting (section 2.3), though we still may hope for good performance. Adaptive submodularity [GK11] extends the guarantees of submodularity to the adaptive setting, requiring properties similar to those of submodular functions. Unfortunately, information gain does not have these properties. With information gain as our inspiration, we design similar metrics that do.

A natural analog of maximizing information gain, which aims to concentrate the target distribution to a single point, is to identify the true hypothesis. This is known as the Optimal Decision Tree (ODT) problem [KB99]. An adaptive submodular method known as Generalized Binary Search (GBS) [Das04; Now08; Now09] solves this problem near-optimally [GB09; GK11]. We extend this method for touch-based localization, modelling the necessary assumptions and allowing for noisy observations while maintaining adaptive submodularity.

In this chapter, we present three greedy methods for selecting uncertainty reducing actions. The first is our variant of information gain. Our method is similar to previous works [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b], though we also enforce the assumptions required for submodular maximization. While there are no formal guarantees for applying this metric in the adaptive setting, we might hope for good performance. The latter two methods both satisfy adaptive submodularity, and differ in their models of noisy observations. Like GBS, these metrics
seek to maximize the expected number of hypotheses disproved by information gathering actions. We refer to this as hypothesis pruning. We apply these methods to touch-based localization in simulation, and report accuracy and computation time in section 3.3. Finally, we show the applicability of these methods on a real robot.

### 3.1 Problem Formulation

Our formulation in this chapter builds on the general framework and variables described in section 2.3. We restate the relevant definitions here, while drawing a connection to touch-based localization.

We represent uncertainty as a set of hypotheses $h \in \mathcal{H}$ with prior distribution $P(h)$. For touch-based localization, each $h$ represents the full pose of the target object. We gather information by running tests $t \in \mathcal{T}$, which result in outcomes/observations $o \in O$. For touch-based localization, these correspond to guarded moves [WG75], where the hand moves along a path until it feels contact. The corresponding outcome $o$ tells us where along the trajectory the hand stopped, or that it reached the end without contact. Guarded moves been used for touch-based localization before [Hsi+08; Hsi09; PK11; Heb+13].

Given a set of test-outcome pairs, we can update the distribution over hypotheses, e.g. eliminate object locations which could not have resulted in contact at those locations. We call the set of test-outcome pairs our evidence $S \subseteq \mathcal{T} \times O$, where $S = \{(t_1,o_1), \ldots, (t_m,o_m)\}$. Given evidence $S$, we update our distribution as $P(h \mid S)$. This quantity is similar to the notion of belief states in Partially Observable Mark Decision Processes (POMDPs) [Kae+98] in that it captures our current distribution of the world given the evidence so far.

We would like to find a policy $\pi$ for running tests that allows us to reduce uncertainty below some threshold $Q$. Let $f(S)$ be a function measuring the reduction of uncertainty (e.g. decrease in Shannon entropy). A policy $\pi$ maps the set of evidence so far $S$ to the next test to choose (or to stop running tests). We denote $S(\pi, h)$ as the evidence we would gather (tests we would run and outcomes we would observe) by running policy $\pi$ if hypothesis $h$ were the true hypothesis. See fig. 3.1 for an example execution of a policy used to localize a door handle with an initially unknown pose.

Our ultimate goal is to find a policy $\pi$ which reduces uncertainty below $Q$ while minimizing the cost of running tests. Each test $t$ has a known cost $c(t)$. For touch-based localization, this corresponds to the length of the guarded-move trajectory, plus some cost for getting to the start. To gather evidence, we pay an additive cost for each test, $C(S) = \sum_{t \in S} c(t)$. We compute the expected and worst-case costs of
a policy \( \pi \) as:

\[
C(\pi) = \mathbb{E}_H[C(S(\pi, h))]
\]

\[
C_{\text{wc}}(\pi) = \max_H C(S(\pi, h))
\]

As formulated here, this corresponds exactly to the *Adaptive Stochastic Minimum Cost Cover Problem* (1), which we restate:

**Problem 1** (Adaptive Stochastic Minimum Cost Cover). Let \( Q \) be some quota of the objective function we wish to obtain (e.g. gather enough information to make a decision). We seek a policy that obtains this quota for any hypothesis \( h \in \mathcal{H} \) while minimizing the expected cost:

\[
\pi^* = \arg \min_{\pi} C(\pi) \text{ s.t. } f(S(\pi, h)) \geq Q \quad \forall h \in \mathcal{H}
\]  \hspace{1cm} (2.2)

This is no surprise - we constructed our model specifically to enable this connection. With this connection made, we can view touch-based localization as an instance of (adaptive) set cover, a classic example of submodular maximization. In particular, our method attempts to “cover” the uncertainty, as illustrated in fig. 3.2.

However, not all active touch-based localization methods fit in this framework. We require that certain assumptions are satisfied which are often violated [Hsi+08; Hsi09; Heb+13]. We now make clear these assumptions, and their implications for touch-based localization.

### 3.1.1 Submodularity Assumptions for Touch Localization

Fitting into the framework of submodular maximization necessitates certain assumptions, related to maintaining the diminishing-returns property of \( t \). Broadly speaking, this states that the expected benefit of \( t \) diminishes as we gain more evidence. See section 2.3 for a rigorous definition. In general, this corresponds to models where \( t \) does not change the state of the belief in such a way that a test \( t' \) becomes more informative than it would be now.

This requirement places certain restrictions on our framework. First, we cannot alter the underlying hypotheses \( h \in \mathcal{H} \), so tests are not allowed to change the state of the environment or objects.
If we could, a non-informative test $t$ could suddenly become very informative [Hsi09]. Therefore, we cannot intentionally reposition objects, or model object movement caused by contact. In a perfect world, guarded-moves would stop immediately on contact, satisfying this assumption.

Second, the cost of each test $C(t)$ must remain constant. In touch based localization, this means that a cost function based on the current position of the end-effector, which changes, is not applicable [Heb+13]. However, models where the end-effector is assumed to return to a fixed start position [Hsi09], do satisfy this requirement.

Finally, all tests must be available at every step. Intuitively, if tests are generated at each step, then a new test may simply be better than anything so far. Instead, we generate a large, fixed set of information gathering trajectories at the start. This further enables us to precompute quantities for our observations and updates, enabling faster computation.

When applied to touch-based localization, this framework lends itself towards heavy objects that remain stationary when touched. For such problems, we believe having an efficient algorithm with guaranteed near-optimality outweighs these limitations.\footnote{One possible way to alleviate these limitations would be through near-touch sensors [Hsi+09; JS12], which may enable information gathering actions similar to guarded moves without making contact.}

### 3.2 Metrics for Touch-Based Localization

We now define our various metrics for active information gathering for touch-based localization. Each corresponds to an instantiation of an objective function $f$, which we greedily optimize.

#### 3.2.1 Information Gain

Information gain has been applied to touch-based localization before [Hsi+08; Heb+13]. In contrast to these, we utilize a large fixed set of actions, enforce the assumptions from section 3.1.1, and use a particle-based model (as opposed to a histogram).

Following Krause and Guestrin [KG05], we define the information gain as the reduction in Shannon entropy. Let $H(\mathcal{H})$ be a measure of Shannon entropy for the distribution of hypotheses $\mathcal{H}$. Our objective is defined as:

$$f_{IG}(S) = H(\mathcal{H}) - H(\mathcal{H} | S)$$

At each iteration, we maximize for the cost-normalized marginal utility $\Delta_f$, defined here as\footnote{Instead of using the information gain, we could have also used the entropy of the resulting distribution directly, $\Delta_f = H(\mathcal{H} | S)$. This measure is also submodular [Fuj78], which follows directly from the “information never hurts” principle [CT91]. However, Krause and Guestrin [KG05] argue that this is a less direct measure for reducing uncertainty. Experimentally, they also show that information gain outperforms directly optimizing for entropy reduction.}:

$$\Delta_{f_{IG}}(t) = \mathbb{E}_o[f_{IG}(S \cup \{(t, o)\}) - f_{IG}(S) | S]$$

Krause and Guestrin [KG05] show that this function is monotone submodular if the evidence $S$ is conditionally independent given the
hypothesis pruning for touch-based localization

Thus, if we are evaluating this open-loop, we would be near-optimal compared to the optimal open-loop solution. However, this can actually perform exponentially worse than the online solution [GK11; Hol+11]. Therefore, we apply this method with an adaptive policy.

We also need to define the probability of an observation. Let \( t_h \) be the time of contact for using guarded move \( t \) if the object where at location \( h \) (fig. 3.3a). We consider a “blurred” measurement model where the probability of stopping at \( o \) conditioned on hypothesis \( h \) is weighted based on the time difference between \( o \) and \( t_h \), using a Gaussian with \( \sigma \) modelling the measurement noise:

\[
P(t_H = o | h) \propto \exp\left(-\frac{|o - t_h|}{2\sigma^2}\right)
\]

See fig. 3.3 for an illustration.

We could consider evaluating \( H(H | S) \) with a discrete entropy calculation, where each \( h \in H \) is treated as an individual item. However, our particle set \( H \) models an underlying continuous distribution, and we would like to capture that. Thus, we instead fit a Gaussian to \( P(H \mid S) \) and evaluate the entropy of that distribution. Let \( \Sigma_S \) be the covariance over the weighted set of hypotheses given evidence \( S \), and \( N \) the number of parameters (typically \( x, y, z, \theta \)). We approximate the entropy as:

\[
H(H \mid S) \approx \frac{1}{2} \ln((2\pi e)^N|\Sigma_S|)
\]

After performing the selected test, we update the belief by reweighting hypotheses using our observation model and Bayes rule. We continue gathering evidence until we reach some desired threshold of \( H(H \mid S) \).

3.2.2 Hypothesis Pruning

Intuitively, information gain is attempting to reduce uncertainty by removing probability mass. Here, we formulate a method with this underlying idea that also satisfies properties of adaptive submodularity and strong adaptive monotonicity. We refer to this as Hypothesis Pruning, since the idea is to prune away hypotheses that do not agree with observations. Golovin et al. describe the connection between this type of objective and adaptive submodular set cover [GK11]. Our formulation is similar - see fig. 3.2 for a visualization.

We note that adaptive submodular functions [GK11] cannot handle noise - they require any hypothesis \( h \) be consistent with only one observation per test. However, we would like to model sensor noise. A standard method for alleviating this is to construct a non-noisy
problem by generating a noisy hypothesis for every possible noisy observations of every \( h \in \mathcal{H} \). Let \( \Omega_T(h) = \{ \hat{h}_1, \hat{h}_2, \ldots \} \) be the function that generates the noisy hypotheses \( \hat{h}_i \) for every test \( T \). With this construction, we have transformed our problem into one where we have deterministic observations, one for each \( \hat{h}_i \). Underlying our formulation, we consider constructing this problem. Luckily, we can compute our objective function on the original \( \mathcal{H} \), and do not need to explicitly perform this construction. We present this more efficient computation below, and show how to construct the equivalent non-noisy problem in appendix A.1.1.

As before, we consider a “blurred” measurement model through two different observation models. In the first, we define a cutoff threshold \( d_T \). If a hypothesis is within the threshold, we keep it entirely. Otherwise, it is removed. We call this metric Hypothesis Pruning (HP). In the second, we downweight hypotheses with a (non-normalized) Gaussian, effectively removing a portion of the hypothesis. We call this metric Weighted Hypothesis Pruning (WHP).

The weighting functions are:

\[
\omega_{o}^{HP}(t_h) = \begin{cases} 
1 & \text{if } |o - t_h| \leq d_T \\
0 & \text{else} 
\end{cases}
\]

\[
\omega_{o}^{WHP}(t_h) = \exp \left( -\frac{|o - t_h|^2}{2\sigma^2} \right)
\]

Given evidence \( S \), we define \( w_S(h) \) as the downweighted \( h \) given \( S \), corresponding to the product of weights:

\[
w_S(h) = \left( \prod_{(t,o) \in S} \omega_{o}(t_h) \right) P(h)
\]

Note that this can never increase the probability - for any \( S \), \( w_S(h) \leq P(h) \).

Define \( M_S \) as the total weight of hypothesis given evidence \( S \), and \( m_{S,t,o} \) as the weight of hypotheses remaining after an additional test \( t \) and observation \( o \):

\[
M_S = \sum_{h \in \mathcal{H}} w_S(h)
\]

\[
m_{S,t,o} = \sum_{h \in \mathcal{H}} w_S(h) \omega_{o}(t_h)
\]

We can now define our objective function for any partial realization \( S \), corresponding to removing probability mass:

\[
f(S) = 1 - M_S \quad (3.1)
\]

In particular, we define two objective functions \( f_{HP} \) and \( f_{WHP} \), which correspond to computing the probability mass with the two weighting function \( \omega_{o}^{HP} \) and \( \omega_{o}^{WHP} \), respectively.
In practice, we need to discretize the infinite observation set $O$. Formally, we require that an equal number of observations per hypothesis $h$ are considered. That is, for any test $t$ and any hypotheses $h_i, h_j, |\Omega_t(h_i)| = |\Omega_t(h_j)|$. In practice, we sample observations uniformly along the trajectory to approximately achieve this effect.

To calculate the expected marginal utility, we also need to define the probability of receiving an observation over all hypotheses. We present it here, and show the derivation in appendix A.1.2. Intuitively, this will be proportional to how much probability mass agrees with the observation. Let $O_t$ be the set of all possible observations for test $t$:

$$P(t_H = o | S) = \frac{m_{S,t,o}}{\sum_{o' \in O_t} m_{S,t,o'}}$$

The expected marginal utility corresponds to the expected weight of hypotheses we remove:

$$\Delta_f(t | S) = \mathbb{E}_{o \in O_t} [f(S \cup \{(t,o)\}) - f(S) | S] = \sum_{o \in O_t} \frac{m_{S,t,o}}{\sum_{o' \in O_t} m_{S,t,o'}} [M - m_{S,t,o}]$$

The greedy policy $\pi^g$ maximizes the expected weight of hypotheses removed per unit cost, $\frac{\Delta_f(t | S)}{c(t)}$. After selecting an test and receiving an observation, hypotheses are removed or downweighted as described above, and test selection is iterated. We now present our main guarantee:

**Theorem 3 (Performance Bound of HP and WHP).** Let our objective function be $f$ as defined in eq. (3.1), utilizing either weighting function $\omega^{\text{HP}}$ or $\omega^{\text{WHP}}$. Define a threshold $Q$ for the total weight of hypotheses we wish to remove. Let $\eta$ be any value such that $f(S) > Q - \eta$ implies $f(S) \geq Q$ for all $S$. Let $\pi^g_{\text{avg}}$ and $\pi^*_{\text{wc}}$ be the optimal policies minimizing the expected and worst-case cost of tests selected, respectively. The greedy policy $\pi^g$ satisfies:

$$C(\pi^g) \leq C(\pi^*) \left( \ln \frac{Q}{\eta} + 1 \right)$$

$$C_{\text{wc}}(\pi^g) \leq C_{\text{wc}}(\pi^*) \left( \ln \frac{Q}{\delta \eta} + 1 \right)$$

With $\delta$ a constant based on the underlying non-noisy problem, described in appendix A.1.3.

Our proof, located in appendix A.1, shows that $f_{\text{HP}}$ and $f_{\text{WHP}}$ are adaptive submodular and strongly adaptive monotone. We then utilize theorems 5.8 and 5.9 of [GK11] to provide our bound.

In addition to being within a logarithmic factor of optimal, adaptive submodularity enables an efficient lazy-greedy algorithm, which

---

Note that we must be consistent between contact and no-contact observations. That is, if we believe test $t$ will contact $h_i$ but not $h_j$, it still must be that $|\Omega_t(h_i)| = |\Omega_t(h_j)|$. Thus, we also have multiple noisy no-contact observations. See appendix A.1.2 for details.
does not reevaluate all tests at every step, speeding up computation [Min78; GK11].

3.3 Experiments

We implement greedy test selection with each of the metrics described above (IG, HP, WHP). In addition, we compare against two other methods - random test selection, and a simple human-designed method which approaches the object orthogonally along the X, Y and Z axes. Each object pose \( h \) consist of a 4-tuple \( (x, y, z, \theta) \in \mathbb{R}^4 \), where \( (x, y, z) \) are the coordinates of the object’s center, and \( \theta \) is the rotation about the z axis.

We implement our algorithms using a 7-dof Barret arm with an attached 4-dof Barret hand. We localize two objects: a drill upright on a table, and a door. We define an initial sensed location \( X_s \in \mathbb{R}^4 \). To generate the initial \( \mathcal{H} \), we sample a Gaussian distribution \( N(\mu, \Sigma) \), where \( \mu = X_s \), and \( \Sigma \) is the prior covariance of the sensor’s noise. For simulation experiments, we also define the ground truth pose \( X_t \in \mathbb{R}^4 \).

For efficiency purposes, we also use a fixed number of hypotheses \( |\mathcal{H}| \) at all steps, and resample after each selection, adding small noise to the resampled set.

3.3.1 Action Generation

We generate our guarded moves [WG75] as linear motions of the end effector, consisting of a starting pose and a movement vector. Each test starts outside of all hypotheses, and moves as far as necessary to contact every hypothesis along the path. Note that using straight-line trajectories is not a requirement for our algorithm. We generate tests via three main techniques.

**Sphere Sampling**

Starting positions are generated by sampling a sphere around the sensed position \( X_s \). For each starting position, the end-effector is oriented to face the object, and the movement direction set to \( X_s \). A random rotation is applied about the movement direction, and a random translation along the plane orthogonal to the movement.

**Normal Sampling**

These tests are intended to have the hand’s fingers contact the object orthogonally. First, we uniformly sample random contacts from the surface of the object. Then, for each fingertip, we align its pre-defined
contact point and normal with our random sample, and randomly rotate the hand about the contact normal. We then set the movement direction as the contact normal.

**Table Contacting**

We generate random start points around the sensed position $X_s$, and orient the end effector in the $-z$ direction. These are intended to contact the table which the object is on, and reduce uncertainty in $z$.

### 3.3.2 Simulation Experiments Setup

We simulate an initial sensor error as $X_t - X_s = (0.015, -0.015, -0.01, 0.05)$ (in meters and radians). Our initial random realization $\mathcal{H}$ is sampled from $N(\mu, \Sigma)$ with $\mu = X_s$, and $\Sigma$ a diagonal matrix with $\Sigma_{xx} = 0.03$, $\Sigma_{yy} = 0.03$, $\Sigma_{zz} = 0.03$, $\Sigma_{\theta \phi} = 0.1$. We fix $|\mathcal{H}| = 1500$ hypotheses.

We then generate an identical test set $\mathcal{T}$ for each metric. The set consists of the 3 human designed trajectories, 30 sphere sampled trajectories, 160 normal trajectories, and 10 table contact trajectories (section 3.3.1), giving $|\mathcal{T}| = 203$.

We run 10 experiments using a different random seed for each, generating a different set $\mathcal{T}$ and $\mathcal{H}$, but ensuring each metric has the same $\mathcal{T}$ and initial $\mathcal{H}$ for a random seed. Each method chooses a sequence of five tests, except the human designed sequence which consists of only three tests.

### 3.3.3 Simulation Experiments Results

We analyze the uncertainty reduction of each metric as the sum of eigenvalues of the covariance matrix, shown in fig. 3.4. All metrics were able to reduce the uncertainty significantly – confirming that even random tests reduce uncertainty [PK11]. However, as the uncertainty is reduced, the importance of test selection increases, as evidenced by the relatively poor performance of random selection for the later tests. Additionally, we see the human designed trajectories are effective for the drill, but perform poorly on the door. Unlike the drill, the door is not radially symmetric, and its flat surface and protruding handle offer geometric landmarks that our test selection metrics can exploit.

For one drill experiment, we also display the first 5 tests selected in table 3.2. Note that the tests selected are very different, while performance appears similar.

**Observation 1**: Information Gain (IG), Hypothesis Pruning (HP), and Weighted Hypothesis Pruning (WHP) all perform similarly well. On the one hand, you might expect IG to perform poorly with adap-
tive greedy selection, as we don’t have any guarantees. On the other, Shannon entropy has many properties that make it a good measure of uncertainty. Figure 3.4 displays the covariance of all particles, which is the criterion IG directly optimizes. Note that, surprisingly, HP and WHP perform comparably despite not directly optimizing this measure.

**Observation 2:** The HP and WHP perform faster than IG (table 3.1). This is due to their inherent simplicity and the more efficient lazy-greedy algorithm [Min78; GK11]. Additionally, we lose little performance with large computational gains with the non-weighted observation model of HP.

### 3.3.4 Robot Experiments

We implemented each of our methods (IG, HP, WHP) on a robot with a Barret arm and hand, and attempted to open a door. $X_s$ is initialized with a vision system corrupted with an artificial error of 0.035 m in the y direction. Our initial random realization $H$ is sampled from $N(\mu, \Sigma)$ with $\mu = X_s$, and $\Sigma$ a diagonal matrix with $\Sigma_{xx} = 0.02, \Sigma_{yy} = 0.04, \Sigma_{zz} = 0.02, \Sigma_{\theta\theta} = 0.08$. We fix $|H| = 2000$ hypotheses. We initially generate 600 normal tests trajectories (section 3.3.1), though after checking for kinematic feasibility, only about 70 remain.

We utilize each of our uncertainty reducing methods prior to using an open-loop sequence to grasp the door handle. Once a method selects the next test, we motion plan to its start pose and perform the straight line guarded-move using a task space controller. We sense contact by thresholding the magnitude reported by a force torque sensor in the Barret hand.

Without touch localization, the robot missed the door handle entirely. With any of our localization methods, the robot successfully opened the door, needing only two uncertainty reducing tests to do so. Selected tests are shown in table 3.3.

**Observation 3:** Using our faster adaptive submodular metrics, selecting a test takes approximately as long as planning and executing it. This suggests that adaptive test selection will often outperform a non-adaptive plan generated offline that requires no planning time, but more tests.

<table>
<thead>
<tr>
<th></th>
<th>IG</th>
<th>HP</th>
<th>WHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>47.171 ± 0.25</td>
<td><strong>8.41 ± 0.58</strong></td>
<td>25.70 ± 0.29</td>
</tr>
</tbody>
</table>

Table 3.1: Time to select one test for each metric, average and 95% CI over drill experiments described in section 3.3.2
<table>
<thead>
<tr>
<th>Action</th>
<th>IG</th>
<th>HP</th>
<th>WHP</th>
<th>Random</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: First five tests selected for each metric in one of our drill experiments (except Human, which only has 3 tests). Hypotheses prior to the test are grey, and hypotheses updated after observation are yellow.
3.4 Discussion

In this work, we made a connection between submodularity and information gathering in robotics with specific application to touch-based localization. Our insight stems from noting that many natural notions of information, such as the reduction of Fisher Information [Hoi+06] and reduction of Shannon Entropy [KG05], are submodular. This property renders an efficient greedy policy near-optimal [Nem+78; Wol82], an attractive property for active information gathering as the computational cost grows exponentially with search depth.

We first provided the specific assumptions required to model active information gathering problems in robotics as submodular maximizations (section 3.1.1). With these assumptions, we presented our own submodular variant of the information gain (IG) (section 3.2.1), a commonly used metric in robotics [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b]. This renders the greedy algorithm near-optimal in the open-loop setting, where the system does not choose tests based on observations it has received. Next, we presented our own metrics, Hypothesis Pruning (HP) and Weighted Hypothesis Pruning (WHP) (section 3.2.2) which satisfy adaptive submodularity [GK11]. This enabled us to show that greedy selection is guaranteed to provide near-optimal performance in the adaptive setting. In addition, these metrics are much faster, both due
to their simplicity and a more efficient lazy-greedy algorithm [Min78; GK11].

One potential downside of this work is that it reduces uncertainty indiscriminately, without considering the task at hand. For example, actions meant to grasp a hand can often tolerate uncertainty inherently [DS10], enabling us to perform the task successfully without identifying the true hypothesis. We address this limitation in chapter 4.
In chapter 3, we choose tests that reduce uncertainty about the set of hypotheses directly. In many practical problems, we are primarily concerned about reducing uncertainty for the purpose of making a decision. That is, we would like to reduce uncertainty in a structured way to ensure a decision will be successful. Choosing tests that reduce uncertainty dramatically, but still leave it unclear what action to choose, will not be effective.

This is captured most generally by the decision-theoretic Value of Information (VoI) [How66]. Unfortunately, optimizing this criterion in general probabilistic models is NP-PP-complete [KG09].

Instead, we construct the Decision Region Determination (DRD) problem (section 4.1), which captures uncertainty reduction for decision making in a discrete setting. We design surrogate objectives for this problem, maximized if and only if uncertainty is reduced enough to make a decision. Crucially, we prove that our objectives satisfy adaptive submodularity and strong adaptive monotonicity (section 2.3), enabling us to provide near-optimality guarantees with a simple greedy algorithm [GK11]. Experimentally, we show that our methods outperform optimizing for VoI directly, requiring fewer tests before a decision can be made.

We present two efficient greedy methods with near-optimality guarantees. The first, HyperEdge Cutting (HEC, section 4.2), exhibits a tighter optimality bound, but is computationally inefficient in large domains. The second, Decision Region Edge Cutting (DiRECt, section 4.4), is computationally more efficient, with a looser bound, but has nearly the same empirical performance.

4.1 Decision Region Determination (DRD) Problem Statement

We present the Decision Region Determination (DRD) problem for reducing uncertainty for decision making. As before, we represent uncertainty as a set of hypotheses $h \in \mathcal{H}$ with prior distribution $P$. 
We gain information by running tests \( t \in T \), which produce outcomes \( o \in O \). Each test and outcome eliminate inconsistent hypotheses\(^1\).

In addition, suppose we have a set of decisions \( R \), with the eventual goal of selecting one after gathering information. For example, in medical diagnosis, we choose a treatment; in robotic manipulation, we press a button; in content search, we recommend a particular movie. Each decision region \( r \in R \) corresponds to the set of hypotheses for which it would succeed, i.e., \( r \subseteq \mathcal{H} \). Formally, we also assume that the set of hypotheses is covered by the collection of decision regions, i.e., \( \mathcal{H} = \cup_R r \).

Our goal is to select tests that quickly concentrate all consistent hypotheses in a single decision region. Upon doing so, we know a decision that would succeed. See fig. 4.1.

Important, we need not reduce all uncertainty (i.e. identify \( h^* \)) to make a decision. For example, to push a button, we can tolerate uncertainty related to the size of the button and the direction we will push (fig. 4.2). Compared to our prior work (chapter 3), and the commonly used metric of reduction of Shannon entropy [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Hsi+08; Heb+13; Sad+16b], we desire an algorithm that reduces uncertainty while considering the possible decisions. Similar methods have outperform indiscriminate uncertainty reduction, requiring fewer tests before a decision can be made [Hsi09].

Recall that our evidence so far is captured by the set of test-outcome pairs, \( S \subseteq T \times O \), where \( S = \{ (t_1, h^*(t_1)), \ldots, (t_m, h^*(t_m)) \} \). Given the evidence, we denote the resulting set of consistent hypotheses as the version space \( \mathcal{V}(S) \), defined in eq. (2.1).

We would like to find a policy \( \pi \) for running tests that allows us to determine a decision region \( r \) the true hypothesis \( h^* \) is guaranteed to lie in. Formally, a policy \( \pi \) is a function from a set of evidence so far \( S \), to the next test to choose (or to stop running tests). Upon termination, we require that \( \mathcal{V}(S) \subseteq r \) for some \( r \in R \).

Our ultimate goal is to find a policy that determines a suitable decision while minimizing the cost of tests. As before, let \( S(\pi, h) \) be the evidence we would gather (tests we would run and observations we would receive) by running policy \( \pi \) if hypothesis \( h \) were the true hypothesis. Each test \( t \) has a known cost \( c(t) \). To gather evidence, we pay an additive cost for each test, \( C(S) = \sum_{t \in S} c(t) \). We compute the expected cost of a policy \( \pi \) as\(^2\):

\[
C(\pi) = \mathbb{E}_{\mathcal{H}}[C(S(\pi, h))]
\]

We are now ready to define our problem, which is similar to problem 1:

---

\(^1\) See section 2.3 and table 2.1 for more rigorous variable definitions.

\(^2\) As before, we can also consider the worst-case cost, \( C_{\text{wc}}(\pi) = \max_{h} C(S(\pi, h)) \). We focus here on the average-case cost, but similar bounds also hold for the worst-case cost with the same methods. See section 2.3.3 for details.
Problem 3 (Decision Region Determination (DRD)). We seek a policy that guarantees the version space is encapsulated by any decision region while minimizing the cost to do so:

$$\pi^* = \arg \min_{\pi} C(\pi) \text{ s.t. } \forall h, \exists r : V(S(\pi, h)) \subseteq r$$ (4.1)

Special cases of problem 3 have been studied before. In particular, when each hypothesis is contained in a dedicated decision region, this is called the Optimal Decision Tree (ODT) problem [KB99]. More generally, the special case where the regions partition the hypothesis space (i.e., do not overlap), is called the Equivalence Class Determination (ECD) problem [Gol+10]. For both of these special cases, it is known that finding a policy $\pi$ for which $C(\pi) \leq C(\pi^*)O(\ln n)$ is NP-hard [Cha+07]. Here, $\pi^*$ indicates the optimum policy.

4.1.1 General Strategy

Optimizing directly for eq. (4.1) is intractable. Instead, our general strategy will be to transform problem 3 into an alternative representation which is more amenable to optimization. In particular, we construct surrogate objectives $f$ which are maximized if and only if problem 3 is solved. Importantly, these functions satisfy both strong adaptive monotonicity and adaptive submodularity. Golovin and Krause [GK11] show that if a function satisfies these properties, then an efficient greedy algorithm provides near optimal solutions. See section 2.3.3 for definitions and details.

We then choose tests to maximize $f$, which results in problem 3 being solved near-optimally.

4.1.2 Special case: Equivalence Class Determination

In general, multiple decisions are equally suitable for a hypothesis (e.g. many ways to grasp an object). Hence, decision regions overlap (fig. 4.1a), where hypotheses can be in multiple decision regions.

As a special case of the Decision Region Determination problem, the Equivalence Class Determination (ECD) problem [Gol+10] only allows disjoint decision regions, i.e., $r_i \cap r_j = \emptyset$ for $i \neq j$. This means that each hypothesis $h$ is associated with a unique decision. For this problem, Golovin et al. [Gol+10] present the EC$^2$ algorithm. Our algorithms extend EC$^2$ to the case of overlapping decision regions. We review this algorithm here, which serves as an inspiration for our approaches.

Here, hypotheses are considered as nodes in a graph $G = (V, E)$, and weighted edges are drawn between hypotheses in different deci-
sion regions. Formally:
\[ E = \bigcup_{i \neq j} \{ \{ h, h' \} : h \in r_i, h' \in r_j \} \]

Where the weight of an edge is \( w(\{ h, h' \}) = P(h) \cdot P(h') \); similarly, the weight of a set of edges is \( w(E') = \sum_{e \in E'} w(e) \). Note that, by construction, these edges are what must be disambiguated in order to make a decision - if an edge exists, then we are unsure which decision to make. Likewise, if there are no edges, everything is in a single region, and problem 3 is solved.

Analogous to the definition of \( V(S) \) in eq. (2.1), we define the set of edges consistent with \( S \) as those edges where both hypotheses incident to the edge are consistent:

\[ E(S) = \{ \{ h, h' \} \in E : \forall (t, o) \in S, h(t) = o, h'(t) = o \} \] (4.2)

A test \( t \) with outcome \( o \) is said to “cut” edges which are no longer consistent (when either of incident hypotheses are inconsistent). Performing tests will cut edges, and we aim to eliminate all edges while minimizing the expected cost incurred, i.e. the number of tests required.

The \( EC^2 \) objective is defined as the total weight of edges cut:

\[ f_{EC}(S) = w(E) - w(E(S)) \]

As in our general strategy, Golovin et al. [Gol+10] show that \( f_{EC} \) is maximized if and only if the ECD problem is solved, where all remaining hypotheses are encapsulated by one decision region. Furthermore, they show that \( f_{EC} \) satisfies both strong adaptive monotonicity and adaptive submodularity, rendering greedy solutions near-optimal. However, these results only hold when decision regions are disjoint\(^3\).

We now present two methods with the same general strategy for the more general DRD problem, and analyze them both theoretically and empirically.

4.2 The HyperEdge Cutting (HEC) Method

We now introduce and analyze our first method – HyperEdge Cutting (HEC). Here, we transform the problem into alternative representation – a hypergraph for splitting decision regions. Observing certain test outcomes corresponds to downweighting or cutting hyperedges in this hypergraph. The construction is chosen so that cutting all hyperedges is a necessary and sufficient condition for driving all uncertainty into a single decision region.

Briefly, a hypergraph \( G \) is a pair \( G = (X, E) \), where \( X \) is a set of elements called nodes, and \( E \) is a collection of multisets of \( X \) called

\(^3\) Additionally, this objective is fast to compute by noting that the sum of edge weights is a elementary symmetric polynomial of order 2. See [Gol+10] for details.
We note that we can fully specify the DRD problem by setting $X = \mathcal{H}$, $E = \mathcal{R}$. We refer to this hypergraph as the region hypergraph $G = (\mathcal{H}, \mathcal{R})$.

### 4.2.1 Splitting Hypergraph Construction

We construct a different hypergraph, the splitting hypergraph $G^s$, and define our objective on that. Here, our hyperedges are not sets, but multisets, a generalization of sets where members are allowed to appear more than once. As a result, a node can potentially appear in a hyperedge multiple times. The cardinality of a hyperedge refers to how many nodes it is connected to.

We observe that for solving the DRD problem, we can group together all hypotheses that share the same region assignments. We refer to this grouping as a subregion $g$, and the set of all subregions as $G$. More formally, for any pair $h_k \in g_j$ and $h_l \in g_i$, we have $h_k \in r_j$ if and only if $h_l \in r_j$. In a slight abuse of notation, we say that a subregion is contained in a region, $g \in r$, if $\forall h \in g, h \in r$ (fig. 4.3b).

Similarly, we say that $h \in e$ if $\exists g \in e$ s.t. $h \in g$. It is easy to see that all remaining hypotheses $V(S)$ are contained in $r$ if and only if all remaining subregions are contained in $r$.

We construct the splitting hypergraph $G^s$ over these subregions. Each subregion $g \in G$ corresponds to a node. The hyperedges $e \in E$ consist of all multisets of precisely $k$ subregions, $e = \{g_1, \ldots, g_k\}$, such that a single decision region does not contain them all (we will describe how $k$ is selected momentarily). Note that hyperedges can contain the same subregion multiple times. Formally,

$$E = \{e: |e| = k \land \exists r \text{ s.t. } \forall h \in e, h \in r\}. \quad (4.3)$$

Our splitting hypergraph is defined as $G^s = (G, E)$. Figure 4.3b illustrates the splitting hypergraph obtained from the DRD instance of fig. 4.3a.

**Hyperedge Cardinality $k$.** Key to attaining our results is the proper selection of hyperedge cardinality $k$. If $k$ is too small, our results won’t hold, and our method won’t solve the DRD problem. If $k$ is too large, we waste computational effort, and our theoretical bounds loosen. Here, we define the cardinality we use practically. Our theorems hold for a smaller, more difficult to compute $k$ as well. See appendix A.2 for details.

$$k = \min \left( \max_{h \in \mathcal{H}} |\{r: h \in r\}|, \max_{r \in \mathcal{R}} |\{g: g \in r\}| \right) + 1 \quad (4.4)$$

Note that each term is a property of the original region hypergraph $G'$: $\max_h |\{r: h \in r\}|$ is the maximum degree of any node, and

---

4 We could consider defining a hypergraph where hyperedges are sets instead of multisets, without containing the same subregion multiple times, and use this to solve the DRD problem. However, when hyperedges have varying cardinality, the objective no longer satisfies adaptive submodularity.
max_r |\{g: g \in r\}| bounds the maximum cardinality of hyperedges.\(^5\)

4.2.2 Relating DRD and HEC

By construction, these hyperedges correspond to hypotheses we must disambiguate if we are to make a decision. We utilize this to make progress for problem 3. Observing a set of test-outcomes \(S \subseteq \mathcal{T} \times \mathcal{O}\) eliminates inconsistent hypotheses, and consequently downweights or eliminates (“cuts”) incident hyperedges (fig. 4.3c). Analogous to the definition of \(\mathcal{V}(S)\) in eq. (2.1) and consistent edges in EC\(^2\) in eq. (4.2), we define the set of hyperedges consistent with \(S\) by

\[
\mathcal{E}(S) = \{e \in \mathcal{E} : \forall(t,o) \in S \forall h \in e, h(t) = o\} \quad (4.5)
\]

The following result guarantees that cutting all hyperedges is a necessary and sufficient condition for success, i.e., driving all uncertainty into a single decision region.

**Theorem 4 (Relation of DRD and HEC).** Suppose we construct a splitting hypergraph by drawing hyperedges of cardinality \(k\) according to eq. (4.3). Let \(S \subseteq \mathcal{T} \times \mathcal{O}\) be a set of evidence. All consistent hypotheses lie in some decision region if and only if all hyperedges are cut, i.e.,

\[
\mathcal{E}(S) = \emptyset \iff \exists r : \mathcal{V}(S) \subseteq r
\]

The proof is provided in appendix A.2.2. Thus, the problem 3 is equivalent to finding a policy of minimum cost that cuts all hyperedges. This insight suggests a natural algorithm: select tests that cut as many edges as possible (in expectation). In the following, we formalize this approach.

4.2.3 Solving DRD through HyperEdge Cutting

Given the above construction, we define a suitable objective function whose maximization will ensure that we pick tests to remove hyperedges quickly, thus providing us with a method that identifies a correct decision region. First, we define the weight of a subregion as the sum of hypothesis weights, \(p(g) = \sum_{h \in g} p(h)\). We define the weight of a hyperedge \(e = \{g_1, \ldots, g_k\}\) as \(w(e) = \prod_{i=1}^{k} P(g_i)\). More generally, we define the weight of a collection of hyperedges as \(w(\{e_1, \ldots, e_n\}) = \sum_{i=1}^{n} w(e_i)\). Now, given a pair of test/observation \((t,o)\), we can identify the set of inconsistent hypotheses, which in turn implies the set of hyperedges that should be downweighted or removed. Formally, given a set of test/observation pairs \(S \subseteq \mathcal{T} \times \mathcal{O}\), we define its utility \(f_{\text{HEC}}(S)\) as

\[
f_{\text{HEC}}(S) = w(\mathcal{E}) - w(\mathcal{E}(S)). \quad (4.6)
\]

\(^1\) It is precisely the maximum cardinality of any hyperedge if we grouped hypotheses into subregions in \(G'\).

![Figure 4.3](image-url)

(a) Regions and hypotheses

(b) Subregions and Hypergraph

(c) Edges cut if all \(h \in g_3\) inconsistent
Thus $f_{\text{HEC}}(S)$ is the total mass of all edges cuts when we have evidence $S$.

A natural approach to the problem 3 is thus to seek policies that maximize eq. (4.6) as quickly as possible. Arguably the simplest approach is a greedy approach that iteratively chooses the test that increases eq. (4.6) as much as possible, in expectation over test outcomes.

Analogous to eq. (2.5), we define the expected marginal gain of a test $t$ given evidence $S \subseteq T \times O$ as follows:

$$
\Delta f_{\text{HEC}}(t \mid S) = E_h[f_{\text{HEC}}(S \cup \{(t, h(t))\}) - f_{\text{HEC}}(S) \mid S]
$$

Thus, $\Delta f_{\text{HEC}}(t \mid S)$ quantifies, for test $t$, the expected reduction in hyperedge mass upon observing the outcome of the test. It is apparent that all hyperedges are cut if and only if $\Delta f_{\text{HEC}}(t \mid S) = 0$ for all tests $t \in T$. Given this, our HEC method simply starts with $S = \emptyset$. It then proceeds in an iterative manner, greedily selecting the test $t^*$ that maximizes the cost-normalized expected marginal benefit, $t^* = \arg \max_t \frac{\Delta f_{\text{HEC}}(t \mid S)}{c(t)}$, observes the outcome $h(t^*)$ and adds the pair $(t^*, h(t^*))$ to $S$. It stops as soon as all edges are cut (i.e., the marginal gain of all tests is 0).

4.2.4 Theoretical Analysis

The key insight behind our analysis is that $f_{\text{HEC}}$ satisfies two properties: strong adaptive monotonicity and adaptive submodularity. Those properties are formally established for our $f_{\text{HEC}}$ objective and the associated marginal gain $\Delta f_{\text{HEC}}$ in the following Theorem:

**Theorem 5** (Adaptive Submodularity of HEC). The objective function $f_{\text{HEC}}$ defined in eq. (4.6) is adaptive submodular and strongly adaptive monotone.

As stated before, Golovin and Krause [GK11] prove that for sequential decision problems satisfying adaptive monotonicity and adaptive submodularity, greedy policies are competitive with the optimal policy.

In particular, as a consequence of theorem 5 and Theorem 5.8 of Golovin and Krause [GK11], we obtain the following result for our HEC method:

**Theorem 6** (HEC Performance Bound). Assume that the prior probability distribution $P$ on the set of hypotheses is rational. Then, the performance of $\pi_{\text{HEC}}$ is bounded as follows:

$$
C(\pi_{\text{HEC}}) \leq (k \ln(1/p_{\min}) + 1)C(\pi^*),
$$

where $p_{\min} = \min_{h \in \mathcal{H}} P(h)$ and $\pi^*$ is the optimal policy.
Algorithm 1: Hyperedge Weight

\begin{algorithm}
\begin{procedure}{Hyperedge Weight}($\mathcal{H}, k$)
\begin{algorithmic}
\Statex Compute subregions $\mathcal{G}$ from $\mathcal{H}$
\Statex $W \leftarrow CHP_k(\mathcal{G})$
\Statex Initialize queue $Q_1$ with every subregion $g \in \mathcal{G}$
\ForAll{$\tilde{k} \leq k$}
\ForAll{$\tilde{z}_k \in Q_k$}
\If{$\exists r \text{ s.t. } \forall h \in \tilde{z}_k, h \in r$}
\Statex $W \leftarrow W - \prod_{g \in \tilde{z}_k} p(g)CHP_{k-\tilde{k}}(\tilde{z}_k)$
\Statex Add all supersets of $\tilde{z}_k$ to $Q_{\tilde{k}+1}$
\EndIf
\EndFor
\EndFor
\Return $W$
\end{algorithmic}
\end{procedure}
\end{algorithm}

Proofs for these theorems are provided in appendix A.2. For the special case of disjoint regions (i.e., the ECD Problem, corresponding to $k = 2$), our objective $f_{HEC}$ is equivalent to the objective function proposed by Golovin et al. [Gol+10], and hence our Theorem 6 strictly generalizes their result. Furthermore, in the special case where each test can have at most two outcomes, and we set $k = 1$, the HEC method is equivalent to the Generalized Binary Search algorithm for the ODT problem, and recovers its approximation guarantee.

4.2.5 Efficient Implementation

Our HEC method computes $\Delta_{f_{HEC}}(t \mid S)$ for every test in $T$, and greedily selects one at each time step. Naively computing this quantity involves constructing the splitting hypergraph $G^s$ for every possible observation, and summing the edge weights. This is computationally expensive, as constructing the graph requires enumerating every multiset of order $k$ and checking if any region contains them all, resulting in a runtime of $O(|G|^k)$. We can, however, quickly prune checks and iteratively consider multisets of growing cardinality during our computation by utilizing the following fact:

**Proposition 1.** A set of subregions $G$ shares a region only if all subsets $G' \subset G$ also share that region.

4.2.5.1 Utilizing Complete Homogeneous Symmetric Polynomials

Our general strategy will be to compute the sum of weights over all multisets of cardinality $k$, and subtract those that correspond to a shared region. To do so efficiently, we identify algebraic structure in computing a sum of multisets, where a multiset corresponds to a
product. Namely, it is equivalent to computing a complete homogeneous symmetric polynomial.

For any $G \subseteq \mathcal{G}$ and cardinality $\hat{k}$, we define $\mathcal{G}_k(G)$ as all multisets over groups $G$ of cardinality $\hat{k}$. Unlike hyperedges, these multisets can share a region. Formally,

$$\mathcal{G}_k(G) = \\{ \{ g_1, \ldots, g_{\hat{k}} \} \subseteq G \}$$

Recall that $w(\mathcal{G}_k(G)) = \sum_{g \in G} \prod g P(g)$. Computing $w(\mathcal{G}_k(G))$ can be performed efficiently as this quantity is exactly equivalent to the complete homogeneous symmetric polynomial (CHP) of degree $\hat{k}$ over $G$.

We now turn our attention to efficiently computing the weight of all multisets that correspond to subregions encapsulated by a region. Let $\xi$ be a set (not multiset) of subregions that shares a region. Formally:

$$\xi = \{ g_1 \ldots g_{\hat{k}} \} \quad \hat{k} \leq k, \#r \text{ s.t. } \xi \subseteq r$$

We compute the term corresponding to $\xi$ we subtract from $CHP_k(G)$ (weight of all multisets), as $\xi$ shares a region. To avoid double counting, we force the term to include $\prod_{g \in \xi} p(g)$ as a factor, i.e. if we think of a hyperedge as a product, we force one link to each element of $\xi$.

$$w(\xi) = \prod_{g \in \xi} p(g) \sum_{l_1 + \ldots + l_{\hat{k} - \hat{k}}} \cdot \ldots \cdot p(g_{\hat{k}})$$

Using this, we compute $w(\mathcal{E}) = CHP_k(G) - \sum_{\xi \subseteq G} w(\xi)$ by finding every set $\xi \subseteq G$ that shares a region. Furthermore, we can utilize

$$\text{Figure 4.4: A depiction of our method as hyperedges. (a) The equivalent hyperedges of } CHP_3(G) \text{. (b) First iteration of algorithm 1 which removes all } |\xi| = 1 \text{ (light edges) by subtracting } g_1CHP_2(\{g_1\}) \text{. (c) Second iteration of algorithm 1 which removes all } |\xi| = 2 \text{ (light edges) by subtracting } g_1g_2CHP_1(\{g_1, g_2\}) + g_2CHP_2(\{g_2\}) \text{. }$$
proposition 1 to prune sets, and only consider $\zeta_{k+1}$ which are supersets of any $\zeta_k$. The method is detailed in algorithm 1, and depicted in fig. 4.4.

Additionally, we note that region assignments do not change as observations are received. In practice, we find all sets of subregions that share a region once. At each time step, we need only sum over the terms corresponding to remaining hypotheses.

Note that in the worst case, this method still has complexity $O(|G|^k)$. This occurs when many, at least $k$, subregions share a single region. The complexity is then controlled by how many distinct subregions a single region can be shattered into, and the largest number of regions a single hypothesis can belong to. However, for many practical problems, we might expect many regions to be separated, e.g., when $|R| \gg k$. In this case, algorithm 1 will be significantly more efficient.

Finally, we note that we can utilize a lazy-greedy algorithm\(^6\), applicable to all adaptive submodular functions, which directly uses the diminishing returns property to skip reevaluation of actions [Min78; GK11].

4.3 HyperEdge Cutting (HEC) Experiments

In this section, we empirically evaluate HEC on the two applications - approximate comparison-based learning and touch based localization with a robotic end effector.

We compare HEC with five baselines. The first two are variants of methods for the specialized versions of the DRD problem described earlier - generalized binary search [Now09] and equivalence class edge cutting [Gol+10]. For generalized binary search (GBS), we assign each hypothesis to its own decision region, and run HEC on this hypothesis-region assignment until only one hypothesis remains. To apply equivalence class edge cutting (EC\(^2\)), decision regions must be disjoint. Thus, we randomly assign each hypothesis to one of the decision regions that it belongs to, and run EC\(^2\) until only one of these new regions remains. For each of these, we also run a slightly modified version, termed GBS-DRD and EC\(^2\)-DRD respectively, which selects tests based on these methods, but terminates once all hypotheses are contained in one decision region in the original DRD problem (i.e. when the HEC termination condition is met).

The last baseline is a classic heuristic from decision theory: myopic value of information (VoI) [How66]. We define a utility function $U(h, r)$ which is 1 if $h \in r$ and 0 otherwise. The utility of $V(S)$ corresponds to the maximum expected utility of any decision region, i.e., the expected utility if we made a decision now. VoI greedily

\(^6\) We describe this algorithm in more detail in section 2.3.3.
chooses the test that maximizes (in expectation over observations) the gain in this utility. Note that if we could solve the intractable problem of nonmyopically optimizing VoI (i.e., look ahead arbitrarily to consider outcomes of sequences of tests), we could solve the DRD problem optimally. In some sense, HEC can be viewed as a surrogate function for nonmyopic value of information.

4.3.1 Comparison-Based Preference Learning

We evaluate HEC on the MovieLens 100k dataset [Her+99], which consists of 1 to 5 ratings of 1682 movies from 943 users. We partition movies into decision regions using these ratings, with the goal of recommending any movie in a decision region. In order to get a similarity measurement between movies, we map them into a 10-dimensional feature space by computing a low-rank approximation of the user/rating matrix through SVD. We then use k-means to partition the set of movies into $|R|$ (non-overlapping) clusters, corresponding to decision regions. Each movie is then assigned to the closest cluster centroids. See Fig. 4.6 for an illustration. A test corresponds to comparing two movies, an observation to selecting one of the two, and consist hypotheses are those which are closer to the selected movie (Euclidean distance in 10-dimensional feature space).

Each experiment corresponds to sampling one movie as the “true” movie. The size of a decision region determines how close our solution is to this (exact) target hypothesis. As the number of regions increases, the size of each decision region shrinks. As a result, the problem requires the selected movie be closer to the true target, at the expense of increased query complexity. Figure 4.5a shows the query complexity of different methods as a function of the number of regions, with the cardinality of the HEC hypergraph fixed to $k = 3$ (i.e., each hypothesis belongs to two decision regions). An extreme
case is when there are only two regions and all hypotheses belong to both regions, giving a query complexity of 0. Other than that, we see that HEC performs consistently better than other methods (e.g., to identify the true region out of 8 regions, it takes on average 6.7 queries for HEC, as opposed to 8 queries for EC²-DRD, 8.5 queries for GBS-DRD, and 10.3 queries for VoI).

To see how the cardinality and region overlap influence performance, we compare the query complexity of different methods by varying the number of regions each hypothesis is assigned to. If we assign more regions to a hypothesis, then the search result is allowed to be further away from the true target, and thus the number of queries required for approximated search should be smaller. fig. 4.5b demonstrates such an effect. We fix the number of clusters to 12, and vary the number of assigned regions (and thus the hyperedge cardinality) from 1 to 4 (k from 2 to 5, respectively). We see that higher cardinality enables HEC to save more queries. For k = 5, it takes HEC 5.3 queries to identify a movie, whereas VoI, GBS-DRD, and EC²-DRD took 8.8, 7.4, and 6.4 queries, respectively. Additionally, table 4.1 shows the running time of HEC for these instances. We see that the accelerated implementation described in section 2.3.3 enables HEC to run efficiently with reasonable hyperedge cardinality on this data set.

4.3.2 Touch-Based Localization

We evaluate HEC on a simple robotic manipulation example. Our task is to push a button with the finger of a robotic end effector. Given a distribution over object location, we generate a set of decisions, corresponding to the end effector going to a particular pose and moving forward in a straight line. Each of these decisions will succeed on a subset of hypotheses, corresponding to a decision region. Decision regions may overlap, as a button can be pushed with many decision actions. See fig. 4.7.

All hypotheses are not contained in a single decision region, so we perform tests to reduce uncertainty. These tests correspond to guarded moves [WG75], where the end effector moves along a path until contact is sensed. After sensing contact, hypotheses are updated by eliminating object locations which could not have produced contact, e.g., if they are far away. Our goal is to find the shortest sequence of tests such that after performing them, there is a single button-push

<table>
<thead>
<tr>
<th>k</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>t(HEC)</td>
<td>0.026s</td>
<td>0.071s</td>
<td>2.58s</td>
<td>&lt; 2min</td>
</tr>
</tbody>
</table>

Table 4.1: Running time of HEC on MovieLens 100k with different cardinality k (|R| = 12)

Figure 4.7: Touch based localization for pushing the button of a microwave. Given hypotheses over object location (a), decision actions are generated. The corresponding decision regions are computed by forward simulating to find hypotheses for which it would succeed (b). Decision regions will overlap. In (c), we see two regions (blue and grey) and their overlap (yellow).
decision that would succeed for all remaining hypotheses.

Given some object location $X_s$, we generate an initial set of 2000 hypotheses $\mathcal{H}$ by sampling from $N(\mu, \Sigma)$ with $\mu = X_s$, and $\Sigma$ a diagonal matrix with $\Sigma_{xx} = \Sigma_{yy} = \Sigma_{zz} = 0.04$. The robot generates 50 decision regions by picking different locations and simulating the end effector forward, and noting which object poses it would succeed on. Hypotheses range from being in zero decision regions to 6, giving us a cardinality $k = 7$. For tests, the robot generates 150 guarded moves by sampling a random start location and orientation.

We conduct experiments on 10 random environments, and randomly sample 100 hypotheses to be the “true” object location (for producing observations during execution), for a total of 1000 experiments. Figure 4.8 shows the query complexity of different methods averaged over these instances. We see that HEC performs well, outperforming GBS, GBS-DRD, EC$^2$, and EC$^2$-DRD handily. Note that myopic VoI performs essentially the same as HEC on these experiments. This is likely due to the short horizon, where 2-3 actions were usually sufficient for reducing uncertainty to a single decision region. We would expect that for longer horizons, myopic VoI would not perform as well.

4.4 The Decision Region Edge Cutting (DiRECT) Method

Even with our implementation based on efficient calculation of complete homogeneous symmetric polynomials, in the worst case, we remain exponential in our hyperedge cardinality $k$. For cases where regions overlap greatly, this is intractable to compute. We now present an method which is linear in the maximum degree of any node in $G'$, which is related to the hyperedge cardinality $k$, defined in eq. (4.4).

Our basic strategy here is to construct an adaptive submodular subproblem for each decision region, which is solved if and only if all uncertainty is encapsulated within that decision region. We combine these problems through a noisy-or formulation, and show that this maintains the adaptive submodularity. Finally, we show how some subproblems can be combined, enabling us to tighten our bound.

4.4.1 The Noisy-OR Construction

Suppose there are $m$ possible decisions: $|R| = m$. We first reduce the DRD problem to $O(m)$ instances of the ECD problem, such that solving any one of them is sufficient for solving the DRD problem. Crucially, the problem we end up solving depends on the unknown hypothesis $h^*$. We design our surrogate DiRECT so that it adaptively determines which instance to solve in order to minimize the expected
acting under uncertainty for information gathering and shared autonomy

Concretely, we construct $m$ different graphs, one for each decision. The role of graph $i$ is to determine whether the unknown hypothesis $h^*$ is contained in decision region $r_i$ or not. Thus we aim to distinguish all the hypotheses in this decision region from the rest. To achieve this, we model graph $i$ as an ECD problem, with one of the decision regions being $r_i$. See fig. 4.9 for illustration.

4.4.2 Relating DRD and D\textsc{irect}

Notice that in this ECD problem, once all the edges are cut, either $i$ is the optimal decision, or one of the subregions encodes the optimal decision. Therefore, optimizing the ECD problem associated with one of the $m$ graphs is a sufficient condition for identifying the optimal decision.

Further notice that, among the $m$ ECD problems associated with the $m$ graphs, at least one of them has to be solved (i.e., all edges cut) before we uncover the optimal decision. Therefore, we get a necessary condition of the DRD constraints: we have to cut all the edges in at least one of the $m$ graphs. This motives us to apply a logical OR operation on the $m$ optimization problems. Denote the EC$^2$ objective function for graph $i$ as $f_{EC}^i$ and normalize them so that $f_{EC}^i(\emptyset) = 0$ corresponds to observing nothing and $f_{EC}^i(S) = 1$ corresponds to all edges being cut. We combine the objective functions $f_{EC}^1, \ldots, f_{EC}^m$ using a Noisy-OR formulation:

$$f_{DRE}(S) = 1 - \prod_i \left(1 - f_{EC}^i(S) \right) \quad (4.7)$$

Note that by design $f_{DRE}(S) = 1$ iff $f_{EC}^i(S) = 1$ for at least one $i$.

Similar to before, we define our expected marginal benefit for $f_{DRE}$ as:

$$\Delta_{DRE}(t | S) = \mathbb{E}_R [f_{DRE}(S \cup \{(t, h(t))\}) - f_{DRE}(S) | S]$$

As before, this suggests a natural algorithm: greedily select the test $t^*$ that maximizes the cost-normalized expected marginal benefit,
The DRD problem

\[
t^* = \arg \max_t \frac{\Delta_{f_{DRE}}(t|S)}{c(t)},
\]
observes the outcome \( h(t^*) \) and adds the pair \((t^*, h(t^*))\) to \( S \). Continue until all hypotheses are encapsulated in any decision region (i.e., the marginal gain of all tests is 0).

4.4.3 Theoretical Analysis

As before, we show this formulation satisfies the properties of strong adaptive monotonicity and adaptive submodularity, rendering this objective amenable to greedy optimization. More formally:

**Theorem 7.** The objective function \( f_{DRE} \) defined in eq. (4.7) is adaptive submodular and strongly adaptive monotone.

The proof of this result can be found in Chen et al. [Che+15]. The key here is that applying the noisy-or operator on multiple EC\(^2\) instances preserves the adaptive submodularity of EC\(^2\). Note that this is not generally true for any adaptive submodular function, but we show that this property is preserved for EC\(^2\). These properties make \( f_{DRE} \) amenable for efficient greedy optimization. More formally:

**Theorem 8.** Assume that the prior probability distribution \( P \) on the set of hypotheses is rational. Then, the performance of \( \pi_{DRE} \) is bounded as follows:

\[
C(\pi_{DRE}) \leq (2m \ln(1/p_{\min}) + 1)C(\pi^*),
\]

where \( p_{\min} = \min_{h \in \mathcal{H}} P(h) \) and \( \pi^* \) is the optimal policy.

This result follows from theorem 7 and the general performance analysis of the greedy policy for adaptive submodular problems by Golovin and Krause [GK11]. More details are provided in Chen et al. [Che+15]. The bound of the greedy algorithm is linear in the number of decision regions. Here the factor \( m \) is a result of taking the product of \( m \) EC\(^2\) instances. However, this bound can often be improved.

4.4.4 Improving the Bound via Graph Coloring

For certain applications, the number of decisions \( m \) can be large. In the extreme case where we have a unique decision for each possible
observation, the bound of Theorem 8 becomes trivial. As noted, this is a result of taking the product of $m$ EC$^2$ instances. Thus, we can improve this bound by constructing fewer instances, each with several non-overlapping decision regions. As long as every decision region is accounted for by at least one ECD instance, this problem remains equivalent to the DRD problem. We select the sets of decision regions for each ECD instance through graph coloring. See Figure 4.10 for illustration.

Formally, we construct an undirected graph $G := \{D, E\}$ over all decision regions, where we establish an edge between any pair of overlapping decision regions. That is, two decision regions $r_i$ and $r_j$ are adjacent in $G$ iff there exists a hypothesis $h$ which is contained in both decision regions, i.e., $h \in r_i \cap r_j$. Finding a minimal set of non-overlapping decision region sets that covers all decisions is equivalent to solving a graph coloring problem, where the goal is to color the vertices of the graph $G$, such that no two adjacent vertices share the same color, using as few colors as possible. Thus, we can construct one ECD problem for all the decision regions of the same color, resulting in a different instances, and then use the Noisy-OR formulation to assemble these objective functions. That gives us the following theorem:

**Theorem 9.** Assume that the prior probability distribution $P$ on the set of hypotheses is rational. Let $\pi_{\text{DRE}}$ be the adaptive greedy policy computed over ECD problem instances obtained via graph coloring, where $\alpha$ is the number of colors used. Then, the performance of $\pi_{\text{DRE}}$ is bounded as:

$$C(\pi_{\text{DRE}}) \leq (2\alpha \ln \frac{1}{p_{\text{min}}}) + 1)C(\pi^*),$$

where $p_{\text{min}} = \min_{h \in H} P(h)$ and $\pi^*$ is the optimal policy.

While obtaining minimum graph colorings is NP-hard in general, one can show that every graph can be efficiently colored with at most one color more than the maximum vertex degree, denoted by deg, using a greedy coloring algorithm [WP67]: consider the vertices in descending order according to the degree; we assign to a vertex the smallest available color not used by its neighbours, adding a fresh color if needed. In the DRD setting, deg is the maximal number of decision regions that any decision region can be overlapped with. In practice, greedy coloring often requires far fewer colors than this upper bound. Additionally, note that when regions are disjoint, $\text{deg} = 0$ and DrRECt reverts to the EC$^2$ algorithm.
4.5 Decision Region Edge Cutting (DiRECT) Experiments

We now consider four instances of the general non-myopic value of information problem. Table 4.2 summarizes how these instances fit into our framework.

<table>
<thead>
<tr>
<th>Application</th>
<th>Test</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Loc.</td>
<td>guarded move</td>
<td>manipulation</td>
</tr>
<tr>
<td>Pref. learning</td>
<td>pair of movies</td>
<td>recommendation</td>
</tr>
<tr>
<td>Conservation</td>
<td>monitoring / probing</td>
<td>conservation</td>
</tr>
<tr>
<td>Risky choice</td>
<td>pair of lottery choices</td>
<td>valuation theory</td>
</tr>
</tbody>
</table>

Table 4.2: Tests and decisions for different applications

For each of the problems, we compare DiRECT against several existing approaches as baselines. The first baseline is myopic optimization of the decision-theoretic value of information (VoI) [How66]. At each step we greedily choose the test that maximizes the expected value given the current observations $S$, i.e., $t \in \arg \max_t \mathbb{E}_h[U(S \cup \{(t, h(t))\})]$. We also compare with algorithms designed for special cases of the DRD problem: generalized binary search (GBS) and equivalence class edge cutting (EC2)⁹. We compare with two versions of these algorithms: one with the algorithms’ original stopping criteria, which we call GBS and EC2, and one with the stopping criteria of the DRD problem, which is referred to as GBS-DRD and EC2-DRD in the results. Finally, we also compare to HEC.

4.5.1 Active Touch-Based Localization

Our first application is a robotic manipulation task of pushing a button, with uncertainty over the target’s pose. Tests consist of guarded moves [WG75], where the end effector moves along a path until contact is sensed. Those hypotheses which would not have produced contact at that location (e.g., they are far away) can be eliminated. Decisions correspond to putting the end effector at a particular location and moving forward. The coinciding decision region consists of all object poses where the button would successfully be pushed. Our goal is to concentrate all consistent hypotheses within a single decision region using the fewest tests.

We model pose uncertainty with 4 parameters: $(x, y, z)$ for positional uncertainty, and $\theta$ for rotation about the $z$ axis. An initial set of 20000 hypotheses are sampled from a normal distribution $N(\mu, \Sigma)$, where $\mu$ is some initial location (e.g., from a camera), and $\Sigma$ is diagonal with $\sigma_x = \sigma_y = \sigma_z = 2.5 \text{cm}$, and $\sigma_\theta = 7.5^\circ$. To compute the myopic...
value of information (VoI) [How66], we define a utility function $u(h, r)$ which is 1 if $h \in r$ and 0 otherwise.

We run DiRECT and HEC on simulated data. In the first simulated experiment, we preselect a grid of 25 button pushing decisions $D$ while ensuring the overlap $r$ is minimal. We randomly generate guarded moves $T$ to select from, varying $|T|$ . In the second, we fix $|T| = 250$ while randomly generate decisions to vary $|D|$. Results are plotted in fig. 4.11. Note that HEC cannot be computed in the latter experiment, as the overlap $r$ becomes very large and HEC quickly becomes intractable.

We see that HEC and DiRECT generally outperform all other baselines, and perform very similarly in fig. 4.12a, despite HEC having a tighter bound. Interestingly, EC$^2$ actually performs worse when the number of decisions increases, as randomly assigning each hypothesis to a single decision region decreases the number of hypotheses in each. We also note that myopic VoI performs comparably – likely because the problem is solved within a short horizon.

We also demonstrate DiRECT on a real robot platform as illustrated in fig. 4.11.

### 4.5.2 Comparison-Based Preference Learning

The second application considers a comparison-based movie recommendation system, which learns a user’s movie preference (e.g., the favorable genre) by sequentially showing her pairs of candidate movies, and letting her choose which one she prefers. We use the MovieLens 100k dataset [Her+99], which consists a matrix of 1 to 5 ratings of 1682 movies from 943 users. For each movie we extract a 10-d feature representation from the rating matrix through SVD. To generate decision regions, we cluster movies using k-means, and assign each movie to the $r$ closest cluster centers.
We demonstrate the performance of HEC and DiRECt on MovieLens in fig. 4.13a and fig. 4.13b. We fix the number of clusters (i.e., decision regions) to 12, and vary \( r \), the number of assigned regions for each hypothesis, from 1 to 6. Note that \( r \) controls the hyperedge cardinality in HEC, which crucially affects the computational complexity. As we can observe, the query complexity (i.e., the number of queries needed to identify the target region) of HEC and DiRECt are lower than all other baselines. However, DiRECt is significantly faster to compute than HEC, and comparable to other baselines. See fig. 4.13b (for \( r = 5 \), HEC failed to pick any tests within an hour).

4.5.3 Adaptive Management for Wild-Life Conservation

Our third application is a real-world value of information problem in natural resource management, where one needs to determine which management action should be undertaken for wild-life conservation. Specifically, the task is to preserve the Eastern Migration Population of whooping cranes (EMP Cranes). An expert panel came up with 8 hypotheses for possible causes of reproductive failure, along with 7 management strategies (as decisions). The decision-hypothesis utility matrix is specified in Table 5 of Runge et al. [Run+11]. Tests aim to resolve specific sources of uncertainty. Our goal is to find the best conservation strategy using the minimal number of tests.

To create decision regions, we assign all hypothesis to decision regions which are \( \# \)-optimal, or all decisions which are within \( \# \) utility of the highest utility decision.\(^{10}\)

Results are plotted in fig. 4.13c. We see that HEC and DiRECt perform comparably well, while significantly outperforming myopic VoI and all other baselines.

\(^{10}\) These results actually correspond to a model which can incorporate test noise in the form of flipped outcomes - see [Che+15] for details. Here, a maximum of 1 flip is allowed for each outcome vector.
4.5.4 Preference Elicitation in Behavioral Economics

We further conduct experiments in an experimental design task. Several theories have been proposed in behavioral economics to explain how people make decisions under risk and uncertainty. We test HEC and DiRECt on six theories of subjective valuation of risky choices [Wak10; TK92; Sha64], namely the (1) expected utility with constant relative risk aversion, (2) expected value, (3) prospect theory, (4) cumulative prospect theory, (5) weighted moments, and (6) weighted standardized moments. Choices are between risky lotteries, i.e., known distribution over payoffs (e.g., the monetary value gained or lost). Tests are pairs of lotteries, and hypotheses correspond to parametrized theories that predict, for a given test, which lottery is preferable. The goal, is to adaptively select a sequence of tests to present to a human subject in order to distinguish which of the six theories best explains the subject’s responses.

We employ the same set of parameters used in Ray et al. [Ray+12] to generate tests and hypotheses. The original setup in Ray et al. [Ray+12] was designed for testing EC², and therefore test realizations of different theories cannot collide. In our experiments, we allow a tolerance $\epsilon$ - that is, if one hypothesis differs from another by at most $\epsilon$, they are considered to be similar, and thus have the same set of optimal decisions. Results for simulated test outcomes with varying $\epsilon$ are shown in Figure 4.13d. We see that HEC and DiRECt generally perform best in this setting, and comparably well to eachother.

4.6 Discussion

This chapter provides a framework for reducing uncertainty for the purpose of making a decision. To do so, we formulated the Decision Region Determination (DRD) problem (section 4.1). We represent each possible decision by the subset of hypotheses for which it would succeed, called the decision region. Our objective was to find a policy that can guarantee the true hypothesis encapsulated by a single decision region. As optimizing for this directly is intractable, we instead design two surrogate objectives (sections 4.2 and 4.4) that are more amenable to optimization. These surrogates were designed to have specific properties: they are maximized if and only if the problem is solved, and they satisfy adaptive submodularity and strong adaptive monotonicity, enabling us to provide near-optimality guarantees.

We experimentally validated our methods (sections 4.3 and 4.5) against a range of baselines. A key takeaway from these results is that decision driven uncertainty reduction outperforms indiscriminate uncertainty reduction. That is, across our experiments, methods which
considered the decision when selecting tests were able to solve problem 3 faster than those that did not. Furthermore, selecting tests without considering decisions and simply terminating once we could make a decision (e.g. GBS-DRD and EC^2-DRD) was not enough, as this was also outperformed by our methods. This is especially notable as the most commonly used criteria for active information gathering, the reduction of Shannon entropy [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b], does not consider decisions.

Additionally, we knowing that considering only decision regions also has poor performance. Golovin et al. [Gol+10] consider greedily reducing the entropy over decision regions, compared to EC^2. Even in the non-overlapping setting\(^1\), they show how this method can perform exponentially worse than EC^2, which gains some utility for removing hypotheses. The HEC and DRECt methods we present share this property, providing near-optimal information gathering that outperforms considering only hypotheses, or only decision regions.

Our two methods, HEC and DRECt, have a few notable differences. The bound of HEC is tighter, and one can construct cases where DRECt requires many graph colors, but HEC can use low-cardinality hyperedges. Empirically, the performance of both methods is similar across our experiments (section 4.5). The biggest difference lies in the computational complexity, where DRECt is significantly more efficient, enabling its application for many more real-world problems.

\(^1\) In the overlapping setting, this metric does not capture how multiple decisions can be used for a single hypothesis, rendering this method further suboptimal.
5

Shared Autonomy Background

In this chapter, we review background and relevant work for our methods on shared autonomy, where both the user and robotic system act simultaneously to achieve shared goals. We focus on two application areas for our framework: shared control teleoperation [Goe63; Ros93; AM97; Deb00; DS13a] and human-robot teaming [HB07; Ara10; DS13b; KS13; MB13; Gom14; Nik17b].

We first review different teleoperation interfaces, highlighting the limitations for providing teleoperation inputs to a robot in different application areas (section 5.1). This will help inform our design decisions for shared-control teleoperation system. Next, we discuss relevant works for intent prediction in section 5.2, which we use to infer user intent in our framework. Finally, we review background material for both shared control teleoperation (section 5.3) and human-robot teaming (section 5.4).

5.1 Teleoperation Interfaces

There are many different interfaces for robotic teleoperation - see fig. 5.1 for examples. In surgical robotics, master-slave systems are common, giving the surgeon control of all degrees of freedom of the robotic end effector [Sim12]. Interfaces for remote vehicles, such as UAVs or cars, come in many forms, such as multi-axis joysticks, buttons on a web interface, or multimodal controls where an operator can switch control modes based on situational requirements [FT01; Fon01].

Assistive robotics have many different interfaces for input, depending on the impairments of the user and the control channels available to them. For assistive wheelchairs, joysticks, chin joysticks, switches, sip-and-puff devices, head tracking, and teeth clicking are all used [Van03; Sim08]. Assistive arms, which have even more degrees of freedom, present a different set of challenges. Interfaces include 3d joysticks with different modes for control [Mah11], task
level point and click interfaces to specify objects to grasp [Tsu+08; Lee+12; Kim+12], GUIs specifying end-effector waypoints [YH11; Lee+12] or individual joint control [Lee+12], and many more.

Brain Computer Interfaces (BCIs) offer an attractive alternative for assistive robotics and prostheses, especially for users with severe motor impairment. Interfaces vary from those placed on the scalp (e.g. EEG), on the surface of the brain (e.g. ECoG), and within the cerebral cortex (e.g. LFP) [Sch+06]. EEG based devices, which sit on the scalp, have been successfully applied to wheelchairs giving simple motion commands [Gal+08] and for arms to initiate the execution of sub-tasks [Sch+15], or to interact with a GUI controlling an arm [Lut+07]. More invasive intracortical devices give superior bandwidth and have been used for continuous high degree of freedom control of upper limb prosthetics [Col+13]. These devices often degrade over time, through there remains some usable signal [Sim+11].

5.2 Intent Prediction

Many prior works suggest that effective human-robot collaboration should not rely on explicit mechanisms for communication (e.g. buttons) [Van+03; Gj03; Gre+07]. Instead, implicit information should be used to make collaboration more seamless. In shared autonomy, this suggests utilizing user inputs and sensing of the environment to infer user intent. This idea has been successfully applied to shared autonomy settings [LO03; Yu+05; Kra+05; Kof+05; Ak08; CD12; DS13a; Hau13; Mue+15].

A variety of models and methods have been used for intent prediction. Hidden markov model (HMM) based methods [LO03; Kra+05; Aar+05; Ak08] predict subtasks or intent during execution, treating the intent as latent state. Schrempf et al. [Sch+07] use a Bayesian network constructed with expert knowledge. Koppula and Saxena [KS13] use conditional random fields (CRFs) with object affordance to predict potential human motions. Wang et al. [Wan+13] learn a generative predictor by extending Gaussian Process Dynamical Models (GPDMs) with a latent variable for intention. Hauser [Hau13] utilizes a Gaussian mixture model over task types (e.g. reach, grasp), and predicts both the task type and continuous parameters for that type (e.g. movements) using Gaussian mixture autoregression.

Many successful works in shared autonomy utilize of maximum entropy inverse optimal control (MaxEnt IOC) [Zie+08] for user goal prediction. Briefly, the user is modelled as a stochastic policy approximately optimizing some cost function. By minimizing the worst-case predictive loss, Ziebart et al. [Zie+08] derive a model where trajectory probability decreases exponentially with cost. They then derive
a method for inferring a distribution over goals from user inputs, where probabilities correspond to how efficiently the inputs achieve each goal [Zie+09]. A key advantage of this framework for shared autonomy is that we can directly optimize for the cost function used to model the user.

Exact global inference over these distributions is computationally infeasible in general continuous state and action spaces. For the special case of LQR systems, Ziebart et al. [Zie+12] show that exact global inference is possible, and provide a computationally efficient method for doing so. Levine and Koltun [LK12] provide a method that considers the expert demonstrations as only locally optimal, and utilize Laplace’s method about the expert demonstration to estimate the log likelihood during learning. Similarly, Dragan and Srinivasa [DS13a] use Laplace’s method about the optimal trajectory between any two points to approximate the distribution over goals during shared control teleoperation. Finn et al. [Fin+16] simultaneously learn a cost function and policy consistent with user demonstrations using deep neural networks, utilizing importance sampling to approximate inference with few samples. Inspired by Generative Adversarial Nets [Goo+14], Ho and Ermon [HE16] directly learn a policy to mimic the user through Generative Adversarial Imitation Learning.

We use the approximation of Dragan and Srinivasa [DS13a] in our framework due to empirical evidence of effectiveness in shared autonomy systems [DS13a; Mue+15].

5.3 Shared Control Teleoperation

Shared control teleoperation has been used to assist disabled users using robotic arms [Kim+06; Kim+12; McM+14; Kat+14; Sch+15; Mue+15] or wheelchairs [Arg14; CD12], operate robots remotely [She+04; YH11; Lee+12], decrease operator fatigue in surgical settings [Par+01; Mar+03; Kra+05; Aar+05; Li+07], and many other applications. As such, there are a great many methods catering to the specific needs of each domain.

One common paradigm launches a fully autonomous takeover when some trigger is activated, such as a user command [She+04; Bie+04; Simo5; Kim+12], or when a goal predictor exceeds some confidence threshold [Fag+04; Kof+05; McM+14; Kat+14]. Others have utilized similar triggers to initiate a subtask in a sequence [Sch+15; Jai+15]. While these systems are effective at accomplishing the task, studies have shown that users often prefer having more control [Kim+12].

Another line of work utilizes high level user commands, and relies

![Modal control used in our feeding experiment on the Kinova MICO, with three control modes and a 2 degree-of-freedom input device. Fewer input DOFs means more modes are required to control the robot.](image)
on autonomy to generate robot motions. Systems have been developed to enable users to specify an end-effector path in 2D, which the robot follows with full configuration space plans [YH11; Hau13]. Point-and-click interfaces have been used for object grasping with varying levels of autonomy [Lee+12]. Eye gaze has been utilized to select a target object and grasp position [Bie+04].

Another paradigm augments user inputs minimally to maintain some desired property, e.g. collision avoidance, without necessarily knowing exactly what goal the user wants to achieve. Sensing and complaint controllers have been used increase safety during teleoperation [Kim+06; Vog+14]. Potential field methods have been employed to push users away from obstacles [CG02] and towards goals [AM97]. For assistive robotics using modal control, where users control subsets of the degrees-of-freedom of the robot in discrete modes (fig. 5.2), Herlant et al. [Her+16] demonstrate a method for automatic time-optimal mode switching.

Similarly, methods have been developed to augment user inputs to follow some constraint. Virtual fixtures, commonly used in surgical robotics settings, are employed to project user commands onto path constraints (e.g. straight lines only) [Par+01; LO03; Mar+03; Kra+05; Aar+05; Li+07]. Mehr et al. [Meh+16] learn constraints online during execution, and apply constraints softly by combining constraint satisfaction with user commands. While these methods benefit from not needing to predict the user’s goal, they generally rely on a high degree-of-freedom input, making their use limited for assistive robotics, where disabled users can operate few DOF at a time and thus rely on modal control [Her+16].

Blending methods [DS13a] attempt to bridge the gap between highly assistive methods with little user control, and minimal assistance with higher user burden. User actions and full autonomy are treated as two independent sources, which are combined by some arbitration function that determines the relative contribution of each (fig. 5.3). Dragan and Srinivasa [DS13a] show that many methods of shared control teleoperation (e.g. autonomous takeover, potential field methods, virtual fixtures) can be generalized as blending with a particular arbitration function.

Blending is one of the most used shared control teleoperation paradigms due to computational efficiency, simplicity, and empirical effectiveness [Li+11; CD12; DS13a; Mue+15; Gop+16]. However, blending has two key drawbacks. First, as two independent decisions are being combined without evaluating the action that will be executed, catastrophic failure can result even when each independent decision would succeed [Tra15]. Second, these systems rely on a predict-then-act framework, predicting the single goal the user is
trying to achieve before providing any assistance. Often, assistance will not be provided for large portions of execution while the system has low confidence in its prediction, as we found in our feeding experiment (section 7.1.2).

Recently, Hauser [Hau13] presented a system which provides assistance for a distribution over goals. Like our method, this policy-based method minimizes an expected cost-to-go while receiving user inputs (fig. 5.4). The system iteratively plans trajectories given the current user goal distribution, executes the plan for some time, and updates the distribution given user inputs. In order to efficiently compute the trajectory, it is assumed that the cost function corresponds to squared distance, resulting in the calculation decomposing over goals. Our model generalizes these notions, enabling the use of any cost function for which a value function can be computed.

In this work, we assume the user does not change their goal or actions based on autonomous assistance, putting the burden of goal inference entirely on the system. Nikolaidis et al. [Nik+17c] present a game-theoretic approach to shared control teleoperation, where the user adapts to the autonomous system. Each user has an adaptability, modelling how likely the user is to change goals based on autonomous assistance. They use a POMDP to learn this adaptability during execution. While more general, this model is computationally intractable for continuous state and actions.

5.4 Human-Robot Teaming

In human-robot teaming, robot action selection that models and optimizes for the human teammate leads to better collaboration. Hoffman and Breazeal [HB07] show that using predictions of a human collaborator during action selection led to more efficient task completion and more favorable perception of robot contribution to team success. Lasota and Shah [LS15] show that planning to avoid portions of the workspace the user will occupy led to faster task completion, less user and robot idling time, greater user satisfaction, and greater perceived safety and comfort. Arai et al. [Ara+10] show that users feel high mental strain when a robot collaborator moves too close or too quickly.

Motion planners have been augmented to include user models and collaboration constraints. For static users, researchers have incorporated collaboration constraints such as safety and social acceptability [Sis+07], and task constraints such as user visibility and reachability [Sis+10; PA10; Mai+11]. For moving users, Mainprice and Berenson [MB13] use a Gaussian mixture model to predict user motion, and select a robot goal that avoids the predicted user locations.
Similar ideas have been used to avoid moving pedestrians. Ziebart et al. [Zie+09] learn a predictor of pedestrian motion, and use this to predict the probability a location will be occupied at each time step. They build a time-varying cost map, penalizing locations likely to be occupied, and optimize trajectories for this cost. Chung and Huang [CH11] use A* search to predict pedestrian motions, including a model of uncertainty, and plan paths using these predictions. Bandyopadhyay et al. [Ban+12] use fixed models for pedestrian motions, and focus on utilizing a POMDP framework with SARSOP [Kur+08] for selecting good actions. Like our approach, this enables them to reason over the entire distribution of potential goals. They show this outperforms utilizing only the maximum likelihood estimate of goal prediction for avoidance.

Others develop methods for how the human-robot team should be structured. Gombolay et al. [Gom+14] study the effects of having the user and robot assign goals to each other. They find that users were willing to cede decision making to the robot if it resulted in greater team fluency [Gom+14]. However, Gombolay et al. [Gom+17] later show that having the autonomous entity assign goals led to less situational awareness. Inspired by training schemes for human-human teams, Nikolaidis and Shah [NS13] present a human-robot cross training method, where the user and robot iteratively switch roles to learn a shared plan. Their model leads to greater team fluency, more concurrent motions, greater perceived robot performance, and greater user trust. Koppula and Saxena [KS13] use conditional random fields to predict the user goal (e.g. grasp cup), and have a robot achieve a complementary goal (e.g. pour water into cup).

Others have studied how robot motions can influence the belief of users. Sisbot et al. [Sis+10] fix the gaze of the robot on its goal to communicate intent. Dragan and Srinivasa [DS13]b incorporate legibility into the motion planner for a robotic arm, causing the robot to exaggerate its motion to communicate intent. They show this leads to more quickly and accurately predicting the robot intent [Dra+13]. Rezvani et al. [Rez+16] study the effects of conveying a robot’s state (e.g. confidence in action selection, anomaly in detection) directly on a user interface for autonomous driving.

Recent works have gone one step further, selecting robot actions that not only change the perceptions of users, but also their actions. Nikolaidis et al. [Nik+17b] model how likely users are to adopt the robot’s policy based on robot actions. They utilize a POMDP to simultaneously learn this user adaptability while steering users to more optimal goals to achieve greater reward. Nikolaidis et al. [Nik+17a] present a more general game theoretic approach where users change their actions based on robot actions, while not com-
pletely adopting the robot’s policy. Similarly, Sadigh et al. [Sad+16b] generate motions for an autonomous car using predictions of how other drivers will respond, enabling them to change the behavior of other users, and infer the internal user state [Sad+16a].

Teaming with an autonomous agent has also been studied outside of robotics. Fern and Tadepalli [FT10] have studied MDPs and POMDPs for interactive assistants that suggest actions to users, who then accept or reject each action. They show that optimal action selection even in this simplified model is PSPACE-complete. However, a simple greedy policy has bounded regret. Nguyen et al. [Ngu+11] and Macindoe et al. [Mac+12] apply POMDPs to cooperative games, where autonomous agents simultaneously infer human intentions and take assistance actions. Like our approach, they model users as stochastically optimizing an MDP, and solve for assistance actions with a POMDP. In contrast to these works, our state and action spaces are continuous.
In this chapter, we switch gears to address acting under uncertainty in *shared autonomy*, an instance of human-robot collaborations where the user and robotic system act simultaneously to achieve shared goals. For example, in *shared control teleoperation* [Goe63; Ros93; AM07; Deb+00; DS13a], both the user and system control a single entity, the robot, in order to achieve the user’s goal. In *human-robot teaming*, the user and system act independently to achieve a set of related goals [HB07; Ara+10; DS13b; KS13; MB13; Gom+14; Nik+17b].

A key challenge for these problems is *acting while uncertain* of the user’s goal. To address this, most prior works utilise a *predict-then-act* framework, which splits the task into two parts: 1) predict the user’s goal with high probability, and 2) assist for that single goal, potentially using prediction confidence to regulate assistance [Yu+05; Kof+05; CD12; DS13a; KS13; Mue+15]. Unfortunately, it is often impossible to predict the user’s goal until the end of execution (e.g. cluttered scenes), causing these methods to provide little assistance.

Instead, we aim to gain utility without reducing the problem to one with a known goal. In chapter 4, our key insight was that we could make a decision even while uncertain. We apply a similar insight for shared autonomy: there are often useful assistance actions for *distributions over goals*, even when confidence for a particular goal is low (e.g. move towards multiple goals, figs. 1.4 and 1.5).

In this chapter, we present a general framework for goal-directed shared autonomy that does not rely on predicting a single user goal (fig. 6.1). We assume the user’s goal is fixed (e.g. they want a particular bite of food), and the autonomous system should adapt to the user goal⁴. We formalize our model as a Partially Observable Markov Decision Process (POMDP) [Kae+98], treating the user’s goal as hidden state. When the system is uncertain of the user’s goal, our framework naturally optimizes for an assistance action that is helpful for many goals. When the system confidently predicts a single user goal, our framework focuses assistance given that goal (fig. 6.2).

---

⁴ While we assume the goal is fixed, we do not assume how the user will achieve that goal (e.g. grasp location) is fixed.
Computational efficiency is an important requirement for any shared autonomy system, as we need the system to feel responsive. As our state and action spaces are both continuous, generic POMDP solvers do not fulfill this requirement [Roy+05; SS05; Kur+08; SV10; Sha+12; Som+13; Sei+15]. Instead, we approximate using QMDP [Lit+95], also referred to as Hindsight Optimization [Cho+00; Yoo+08]. This approximation has many properties suitable for shared autonomy: it is computationally efficient, works well when information is gathered easily [Kov+14], and will not oppose the user to gather information. The result is a system that minimizes the expected cost-to-go to assist for any distribution over goals.

A key element of any shared autonomy framework is the user model. Ideally, we model shared autonomy as a cooperative two-player game [Aum61; Li+15; Nik+17a], where the user and robot learn and adapt their strategies to each other. Unfortunately, this is computationally difficult for use in real-time systems. Instead, we approximate by assuming the user acts without knowledge of assistance, allowing us to leverage existing works in user prediction in our shared autonomy framework. We discuss the ideal model and implications of our assumption in section 6.3.

In chapter 7, we show how to apply our framework for shared-control teleoperation (section 7.1) and human-robot teaming (section 7.2). We evaluate our framework on real users, demonstrating that our method achieves goals faster, requires less user input, decreases user idling time, and results in fewer user-robot collisions than those which rely on predicting a single user goal (chapter 7).
### Table 6.1: Shared Autonomy variable definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \in X$</td>
<td>Environment state, e.g. robot and human pose</td>
</tr>
<tr>
<td>$g \in G$</td>
<td>User goal</td>
</tr>
<tr>
<td>$s \in S$</td>
<td>$s = (x, g)$. State and user goal</td>
</tr>
<tr>
<td>$u \in U$</td>
<td>User action</td>
</tr>
<tr>
<td>$a \in A$</td>
<td>Robot action</td>
</tr>
<tr>
<td>$C^u(s, u) = C^u_g(x, u)$</td>
<td>Cost function for each user goal</td>
</tr>
<tr>
<td>$C^r(s, u, a) = C^r_g(x, u, a)$</td>
<td>Robot cost function for each goal</td>
</tr>
<tr>
<td>$T(x'</td>
<td>x, u, a)$</td>
</tr>
<tr>
<td>$T((x', g)</td>
<td>(x, g), u, a) = T(x'</td>
</tr>
<tr>
<td>$T^u(x'</td>
<td>x, u) = T(x'</td>
</tr>
<tr>
<td>$V_g(x) = V^* (s)$</td>
<td>The value function for a user goal and environment state</td>
</tr>
<tr>
<td>$Q_g(x, u, a) = Q^* (s, u, a)$</td>
<td>The action-value function for a user goal and environment state</td>
</tr>
<tr>
<td>$b$</td>
<td>Belief, or distribution over states in our POMDP.</td>
</tr>
<tr>
<td>$\tau(b', b, u, a)$</td>
<td>Transition function of belief state</td>
</tr>
<tr>
<td>$V^{\pi^r}(b)$</td>
<td>Value function for following policy $\pi^r$ given belief $b$</td>
</tr>
<tr>
<td>$Q^{\pi^r}(b, u, a)$</td>
<td>Action-Value for taking actions $u$ and $a$ and following $\pi^r$ thereafter</td>
</tr>
<tr>
<td>$V^{HS}(b)$</td>
<td>Value given by Hindsight Optimization approximation</td>
</tr>
<tr>
<td>$Q^{HS}(b, u, a)$</td>
<td>Action-Value given by Hindsight Optimization approximation</td>
</tr>
</tbody>
</table>

### 6.1 Problem Statement

We present our problem statement for minimizing a cost function for shared autonomy with an unknown user goal. We assume the user’s goal is fixed, and they take actions to achieve that goal without considering autonomous assistance. These actions are used to predict the user’s goal based on how optimal the action is for each goal (section 6.3.1). Our system uses this distribution to minimize the expected cost-to-go (section 6.1.2). As solving for the optimal action is infeasible, we use hindsight optimization to approximate a solution (section 6.2). For reference, see table 6.1 for variable definitions.

#### 6.1.1 Cost Minimization with a Known Goal

We first formulate the problem for a known user goal, which we will use in our solution with an unknown goal. We model this problem as a Markov Decision Process (MDP).

Formally, let $x \in X$ be the environment state (e.g. human and robot pose). Let $u \in U$ be the user actions, and $a \in A$ the robot actions. Both agents can affect the environment state - if the user takes action $u$ and the robot takes action $a$ while in state $x$, the environment stochastically transitions to a new state $x'$ through $T(x' | x, u, a)$.

We assume the user has an intended goal $g \in G$ which does not
change during execution. We augment the environment state with this goal, defined by \( s = (x, g) \in X \times G \). We overload our transition function to model the transition in environment state without changing the goal, \( T((x', g) \mid (x, g), u, a) = T(x' \mid x, u, a) \).

We assume access to a user policy for each goal \( p_u(s) = p_u(g) \). We model this policy using the maximum entropy inverse optimal control (MaxEnt IOC) framework of Ziebart et al. [Zie+08], where the policy corresponds to stochastically optimizing a cost function \( C_u(x, u) \). We assume the user selects actions based only on \( s \), the current environment state and their intended goal, and does not model any actions that the robot might take. Details are in section 6.3.1.

The robot selects actions to minimize a cost function dependent on the user goal and action \( C_r(s, u, a) = C_g(x, u, a) \). At each time step, we assume the user first selects an action, which the robot observes before selecting \( a \). The robot selects actions based on the state and user inputs through a policy \( \pi_r(a \mid s, u) = p(a \mid s, u) \). We define the value function for a robot policy \( V_{\pi_r} \) as the expected cost-to-go from a particular state, assuming some user policy \( \pi_u \):

\[
V_{\pi_r}(s) = \mathbb{E} \left[ \sum_{t} C_r(s_t, u_t, a_t) \mid s_0 = s \right]
\]

(6.1)

\( u_t \sim \pi_u(\cdot \mid s_t) \)
\( a_t \sim \pi_r(\cdot \mid s_t, u_t) \)
\( s_{t+1} \sim T(\cdot \mid s_t, u_t, a_t) \)

The optimal value function \( V^* \) is the cost-to-go for the best robot policy:

\[
V^*(s) = \min_{\pi_r} V_{\pi_r}(s)
\]

The action-value function \( Q^* \) computes the immediate cost of taking action \( a \) after observing \( u \), and following the optimal policy thereafter:

\[
Q^*(s, u, a) = C_r(s, u, a) + \mathbb{E}[V^*(s')] \]

Where \( s' \sim T(\cdot \mid s, u, a) \). The optimal robot action is given by \( \arg\min_a Q^*(s, u, a) \).

In order to make explicit the dependence on the user goal, we often write these quantities as:

\[
V_g(x) = V^*(s)
Q_g(x, u, a) = Q^*(s, u, a)
\]

Computing the optimal policy and corresponding action-value function is a common objective in reinforcement learning. We assume
access to this function in our framework, and describe our particular implementation in the experiments.

6.1.2 Cost Minimization with an Unknown Goal

We formulate the problem of minimizing a cost function with an unknown user goal as a Partially Observable Markov Decision Process (POMDP). A POMDP maps a distribution over states, known as the belief $b$, to actions. We assume that all uncertainty is over the user’s goal, and the environment state is known. This subclass of POMDPs, where uncertainty is constant, has been studied as a Hidden Goal MDP [FT10], robust MDPs [Bago04], POMDP-lite [Che+16], and generally as the theory of dual control [Fel60].

In this framework, we infer a distribution of the user’s goal by observing the user actions $u$. Similar to the known-goal setting (section 6.1.1), we define the value function of a belief as:

$$V^\pi(b) = \mathbb{E} \left[ \sum_t C_t(s_t, u_t, a_t) \mid b_0 = b \right]$$

$$s_t \sim b_t$$
$$u_t \sim \pi^u(\cdot \mid s_t)$$
$$a_t \sim \pi^a(\cdot \mid s_t, u_t)$$
$$b_{t+1} \sim \tau(\cdot \mid b_t, u_t, a_t)$$

Where the belief transition $\tau$ corresponds to transitioning the known environment state $x$ according to $T$, and updating our belief over the user’s goal as described in section 6.3.1. We can define quantities similar to above over beliefs:

$$V^*(b) = \min_\pi V^\pi(b)$$

$$Q^*(b, u, a) = \mathbb{E} \left[ C_t(b, u, a) + \mathbb{E}_{b'}[V^*(b')] \right]$$

6.2 Hindsight Optimization

Computing the optimal solution for a POMDP with continuous states and actions is generally intractable. Instead, we approximate this quantity through Hindsight Optimization [Cho+00; Yoo+08], or QMDP [Lit+95]. This approximation estimates the value function by
switching the order of the min and expectation in eq. (6.2):

$$V_{HS}(b) = \mathbb{E}_b \left[ \min \pi^r V^r(s) \right] = \mathbb{E}_g \left[ V^g(x) \right]$$

$$Q_{HS}(b,u,a) = \mathbb{E}_b \left[ C^r(s,u,a) + \mathbb{E}_{s'} \left[ V_{HS}(s') \right] \right] = \mathbb{E}_g \left[ Q^g(x,u,a) \right]$$

Where we explicitly take the expectation over $$g \in G$$, as we assume that is the only uncertain part of the state.

Conceptually, this approximation corresponds to assuming that all uncertainty will be resolved at the next timestep. At the next timestep, the optimal cost-to-go would then be given by the value function without uncertainty, $$V^g$$. The expectation comes from uncertainty resolved with probability proportional to the current distribution. Note that this is the best case scenario given our current distribution, as we would no longer require hedging against uncertainty. Thus, this is a lower bound of the cost-to-go, $$V_{HS}(b) \leq V^*(b)$$.

As it assumes all uncertainty will be resolved, this method never explicitly gathers information [Lit+95], and thus performs poorly when this is necessary. Nonetheless, hindsight optimization has demonstrated effectiveness in other domains [Yoo+07; Yoo+08].

We believe this method is suitable for shared autonomy for many reasons. Conceptually, we assume the user provides inputs at all times, and therefore we gain information without explicit information gathering. Works in other domains with similar properties have shown that this approximation performs comparably to methods that consider explicit information gathering [Kov+14]. Computationally, computing $$Q_{HS}$$ can be done with continuous state and action spaces, enabling fast reaction to user inputs.

Computing $$Q^g$$ for shared autonomy requires utilizing the stochastic user policy $$\pi^u_g$$ and the robot policy:

**Stochastic user with robot**

Estimate $$u$$ using $$\pi^u_g$$ at each time step, e.g. by sampling and performing rollouts, and utilize the full cost function $$C^r_g(x, u, a)$$ and transition function $$T(x' \mid x, u, a)$$ to compute $$Q^g$$. This would be the standard QMDP approach for our POMDP.

Unfortunately, this can be computationally expensive: for each sample of the user policy, we would need to compute the corresponding robot action $$a$$ we would take, and repeat this process for the duration of the trial. To estimate the value function, we would need many many such rollouts.
Deterministic user with robot

Estimate the user action as the most likely $u$ from $\pi^u_\delta(x)$ at each time step, and utilize the full cost function $C^c_\delta(x, u, a)$ and transition function $T(x' | x, u, a)$ to compute $Q_\delta$. This uses our policy predictor, as above, but does so deterministically, and is thus more computationally efficient. However, this approximation relies heavily on the accuracy of the deterministic policy, and still requires performing a rollout for each user goal.

Robot takes over

Assume the user will stop supplying inputs, and the robot will complete the task. This enables us to use the cost function $C^c_\delta(x, 0, a)$ and transition function $T(x' | x, 0, a)$ to compute $Q_\delta$. For many cost functions, we can analytically compute this value, e.g. cost of always moving towards the goal at some velocity. An additional benefit of this method is that it makes no assumptions about the user policy $\pi^u_\delta$, making it more robust to modelling errors. We use this method in our experiments.

User takes over

Assume the user will complete the task without any assistance. This corresponds to using the distribution $\pi^u_\delta$ to generate user actions $u$ at each timestep, and transition with $T(x' | x, u, 0)$ to compute $Q_\delta$. Unlike the robot takes over approximation, this requires using the stochastic user policy to compute the value function. For discrete state and action spaces, this can be done efficiently prior to execution [Zie+09].

Finally, as we often cannot calculate $\arg\max_a Q^{HS}(b, u, a)$ directly, we use a first-order approximation, which leads us to following the gradient of $Q^{HS}(b, u, a)$.

6.3 User Modelling

Ideally, we model shared autonomy as a cooperative two-player game [Aum61], where the user is aware of the shared autonomy assistance strategy. To achieve this, we might model the user as a learner, changing their actions as they interact with the shared autonomy system and learn its behavior. The user could even be aware of how the shared autonomy system adapts to the user’s behavior, and actively teaches the shared autonomy system the assistance behaviour they desire [HM+16]. These models have been suggested for human-robot collaboration in some settings [Li+15; Nik+17a].
Unfortunately, models for how a user learns are limited, and such models are computationally intractable in continuous domains. A simplifying assumption is that the user already knows the shared autonomy strategy, and acts in accordance with their model of future assistance. We explore an instantiation of this model in chapter 8. These models are also computationally challenging, and our framework in chapter 8 is limited to discrete worlds.

For application to continuous state and action spaces, we assume the user acts without knowledge of assistance. This affects both user goal prediction and the assistance strategy. For goal prediction, we rely on models where a user input is based only on the current state $x$. For the assistance strategy, we optimize for a cost function $C^r_g$ that minimizes the user’s cost under the same model. This can be suboptimal - for example, knowing the user will rely on assistance may lead us to take paths that are difficult for a user but easy for autonomy.

In section 6.3.1, we present our model for goal prediction under the assumption that the user acts without knowledge of assistance. Empirically, we find our assumption this works well for user goal prediction, quickly honing in on the user goal during interaction. However, the assistance strategy may behave differently with knowledge of assistance. In chapter 8, we show how optimizing for a user cost that considers how they respond to assistance improves performance, enabling users to achieve goals with less cost in discrete settings. We leave extensions of this model to the continuous setting as future work.

### 6.3.1 User Goal Prediction

In order to infer the user’s goal, we construct a user model $\pi^u_g$ to provide the distribution of user actions at state $x$ for user goal $g$. In principle, we could use any generative predictor for this model, e.g. [KS13; Wan+13]. We choose to use maximum entropy inverse optimal control (MaxEnt IOC) [Zie+08], as it explicitly models a user cost function $C^u_g$. Our assistance policy optimize for this user cost directly by defining $C^r_g$ as a function of $C^u_g$.

We define the user MDP with states $x \in X$ and user actions $u \in U$ as before, transition $T^u(x' | x, u) = T(x' | x, u, 0)$, and cost $C^u_g(x, u)$. With this MDP, we use MaxEnt IOC to compute a distribution over user actions for each goal. The distribution of actions at a single state are computed based on how optimal that action is for minimizing cost over a horizon $T$. To compute this quantity, we first compute the distribution over trajectories from any state, and compute the distribution over a single action as the sum over the trajectories which
have that as the first action.

Define a sequence of environment states and user inputs as $\tilde{\xi} = \{x_0, u_0, \ldots, x_T, u_T\}$. Note that sequences are not required to be trajectories, in that $x_{t+1}$ is not necessarily the result of applying $u_t$ in state $x_t$. Define the cost of a sequence as the sum of costs of all state-input pairs, $C^u_\tilde{\xi}(\tilde{\xi}) = \sum_t C^u_S(x_t, u_t)$. Let $x^0 \rightarrow t$ be a sequence from time 0 to $t$, and $x^t \rightarrow T$ a sequence of from time $t$ to $T$, starting at $x$.

Ziebart [Zie10] shows that minimizing the worst-case predictive loss results in a model where the probability of a sequence decreases exponentially with cost, $p(\tilde{\xi} \mid g) \propto \exp(-C^u_\tilde{\xi}(\tilde{\xi}))$. Importantly, one can efficiently learn a cost function consistent with this model from demonstrations [Zie+08].

Computationally, the difficulty in computing $p(\tilde{\xi} \mid g)$ lies in the normalizing constant $\int_x \exp(-C^u_\tilde{\xi}(\tilde{\xi}))$, known as the partition function. Evaluating this explicitly would require enumerating all sequences and calculating their cost. However, as the cost of a sequence is the sum of costs of all state-action pairs, dynamic programming can be utilized to compute this through soft-minimum value iteration when the state is discrete [Zie+09; Zie+12]:

$$ Q^\text{soft}_{S,t}(x, u) = C^u_S(x, u) + \mathbb{E}_{\tilde{x}^t \sim T^u} \left[ V^\text{soft}_{S,t+1}(\tilde{x}^t) \right] $$

$$ V^\text{soft}_{S,t}(x) = \text{softmin}_u Q^\text{soft}_{S,t}(x, u) $$

Where $\text{softmin}_y f(y) = -\log \int_y \exp(-f(y))dy$ and $x' \sim T^u(\cdot \mid x, u)$.

The log partition function is given by the soft value function,

$$ V^\text{soft}_{S,t}(x) = -\log \int_{\tilde{\xi}^t \rightarrow T} \exp \left( -C^u_\tilde{\xi}(\tilde{\xi}^t \rightarrow T) \right), $$

where the integral is over all sequences starting at $x$ and time $t$. Furthermore, the probability of a single input at a given environment state is given by $\pi^u_{S,t}(u \mid x, g) = \exp(V^\text{soft}_{S,t}(x) - Q^\text{soft}_{S,t}(x, u))$ [Zie+09].

Many works derive a simplification that enables them to only look at the start and current states, ignoring the inputs in between [Zie+12; DS13a]. Key to this assumption is that $\tilde{\xi}$ corresponds to a trajectory, where applying action $u_t$ at $x_t$ results in $x_{t+1}$. However, if the system is providing assistance, this may not be the case. In particular, if the assistance strategy believes the user’s goal is $g$, the assistance strategy will select actions to minimize $C^u_S$. Applying these simplifications will result positive feedback, where the robot makes itself more confident about goals it already believes are likely. In order to avoid this, we ensure that the prediction comes from user inputs only, and not robot actions:

$$ p(\tilde{\xi} \mid g) = \prod_t \pi^u_t(u_t \mid x_t, g) $$

Finally, we apply Bayes’ rule to compute the probability of a goal
given the partial sequence up to \( t \),

\[
p(g \mid s^{0\to t}) = \frac{p(s^{0\to t} \mid g)p(g)}{\sum_{g'} p(s^{0\to t} \mid g')p(g')}
\]

### 6.3.2 Continuous state and action approximation

Soft-minimum value iteration is able to find the exact partition function when states and actions are discrete. However, it is computationally intractable to apply in general continuous state and action spaces\(^2\). Instead, we follow Dragan and Srinivasa [DS13a] and use a second order approximation about the optimal trajectory. They show that, assuming a constant Hessian, we can replace the difficult to compute soft-min functions \( V_{\text{soft}}^g \) and \( Q_{\text{soft}}^g \) with the min value and action-value functions \( V^u_g \) and \( Q^u_g \):

\[
\pi^u_t(u \mid x, g) = \exp(V^u_g(x) - Q^u_g(x, u))
\]

Recent works have explored extensions of the MaxEnt IOC model for continuous spaces [Bou+11; LK12; Fin+16]. We leave experiments using these methods for learning and prediction as future work.

### 6.4 Multi-Target MDP

There are often multiple ways to achieve a goal. We refer to each of these ways as a target. For a single goal (e.g. object to grasp), let the set of targets (e.g. grasp poses) be \( k \in K \). We assume each target has a cost function \( C_k \), from which we compute the corresponding value and action-value functions \( V_k \) and \( Q_k \), and soft-value functions \( V_{\text{soft}}^k \) and \( Q_{\text{soft}}^k \). We derive the quantities for goals, \( V_g, Q_g, V_{\text{soft}}^g, Q_{\text{soft}}^g \), as functions of these target functions.

We state the theorems below, and provide proofs in the appendix (appendix A.3).

#### 6.4.1 Multi-Target Assistance

We assign the cost of a state-action pair to be the cost for the target with the minimum cost-to-go after this state:

\[
C_g(x, u, a) = C_{k^*}(x, u, a) \quad k^* = \arg \min_k V_k(x')
\]

Where \( x' \) is the environment state after actions \( u \) and \( a \) are applied at state \( x \). For the following theorem, we require that our user policy be deterministic, which we already assume in our approximations when computing robot actions in section 6.2.

---

\(^2\) For the special case of LQR systems, Ziebart et al. [Zie+12] provide a computationally efficient method for exact inference.
Theorem 10. Let $V_k$ be the value function for target $k$. Define the cost for the goal as in eq. (6.3). For an MDP with deterministic transitions, and a deterministic user policy $\pi^u$, the value and action-value functions $V_g$ and $Q_g$ can be computed as:

\[
Q_g(x, u, a) = Q_k(x, u, a) \quad \kappa^* = \arg\min_k V_k(x')
\]

\[
V_g(x) = \min_k V_k(x)
\]

6.4.2 Multi-Target Prediction

Here, we don’t assign the goal cost to be the cost of a single target $C_k$, but instead use a distribution over targets.

Theorem 11. Define the probability of a trajectory and target as $p(\xi, k) \propto \exp(-C_k(\xi))$. Let $V_k^{soft}$ and $Q_k^{soft}$ be the soft-value functions for target $k$. For an MDP with deterministic transitions, the soft value functions for goal $g$, $V_g^{soft}$ and $Q_g^{soft}$, can be computed as:

\[
V_g^{soft}(x) = \text{softmin}_k V_k^{soft}(x)
\]

\[
Q_g^{soft}(x, u) = \text{softmin}_k Q_k^{soft}(x, u)
\]

Figure 6.3: Value function for a goal (grasp the ball) decomposed into value functions of targets (grasp poses). (a), (b) Two targets and their corresponding value function $V_k$. In this example, there are 16 targets for the goal. (c) The value function of a goal $V_g$ used for assistance, corresponding to the minimum of all 16 target value functions. (d) The soft-min value function $V_g^{soft}$ used for prediction, corresponding to the soft-min of all 16 target value functions.
7

Shared Autonomy User Studies

In this chapter, we study the efficacy of our shared autonomy framework (section 6.1) on real users. We implement this framework with two applications in mind: shared control teleoperation (section 7.1) and human-robot teaming (section 7.2).

For shared control teleoperation, users performed two tasks: a simpler object grasping task (section 7.1.1), and a more difficult feeding task (section 7.1.2). In both cases, we find that our POMDP based method enabled users to achieve goals faster and with less joystick input than a state-of-the-art predict-then-act method [DS13a]. Subjective user preference differed for each task, with no statistical difference for the simpler object grasping task, and users preferring our POMDP method for the more difficult feeding task.

For human-robot teaming (section 7.2.1), the user and robot performed a collaborative gift-wrapping task, where both agents had to manipulate the same set of objects while avoiding collisions. We found that users spent less time idling and less time in collision while collaborating with a robot using our method. However, results for total task completion time are mixed, as predict-then-act methods are able to take advantage of more optimized motion planners, enabling faster execution once the user goal is confidently predicted.

7.1 Shared Control Teleoperation

We apply our shared autonomy framework to two shared control teleoperation tasks: a simpler task of object grasping (section 7.1.1) and a more complicated task of feeding (section 7.1.2). Formally, the state $x$ corresponds to the end-effector pose of the robot, each goal $g$ an object in the world, and each target $k$ a pose for achieving that goal (e.g. pre-grasp pose). The transition function $T(x' | x, u, a)$ deterministically transitions the state by applying both $u$ and $a$ as end-effector velocities. We map user joystick inputs to $u$ as if the user were controlling the robot through direct teleoperation.
For both tasks, we hand-specify a simple user cost function, $C^u_k$, from which everything is derived. Let $d$ be the distance between the robot state $x' = T^u(x, u)$ and target $k$:

$$C^u_k(x, u) = \begin{cases} 0 & d > \delta \\ \frac{\alpha}{\delta} d & d \leq \delta \end{cases}$$

That is, a linear cost near a target ($d \leq \delta$), and a constant cost otherwise. This is based on our observation that users make fast, constant progress towards their goal when far away, and slow down for alignment when near their goal. This is by no means the best cost function, but it does provide a baseline for performance. We might expect, for example, that incorporating collision avoidance into our cost function may enable better performance [YH11]. We use this cost function, as it enables closed-form value function computation, enabling inference and execution at 50Hz.

For prediction, when the distance is far away from any target ($d > \delta$), our algorithm shifts probability towards goals relative to how much progress the user action makes towards the target. If the user stays close to a particular target ($d \leq \delta$), probability mass automatically shifts to that goal, as the cost for that goal is less than all others.

We set $C^r_k(x, a, u) = C^u_k(x, a)$, causing the robot to optimize for the user cost function directly\(^1\), and behave similar to how we observe users behaved. When far away from goals ($d > \delta$), it makes progress towards all goals in proportion to their probability of being the user’s goal. When near a target ($d \leq \delta$) that has high probability, our system reduces assistance as it approaches the final target pose, letting users adjust the final pose if they wish.

We believe hindsight optimization is a suitable POMDP approximation for shared control teleoperation. A key requirement for shared control teleoperation is efficient computation, in order to

---

\(^1\) In our prior work [Jav+15], we used $C^r_k(x, a, u) = C^u_k(x, a) + (a - u)^2$ in a different framework where only the robot action transitions the state. Both formulations are identical after linearization. Let $a'$ be the optimal optimal robot action in this framework. The additional term $(a - u)^2$ leads to executing the action $u + a'$, equivalent to first executing the user action $u$, then $a'$, as in this framework.
make the system feel responsive. With hindsight optimization, we can provide assistance at 50Hz, even with continuous state and action spaces.

The primary drawback of hindsight optimization is the lack of explicit information gathering [Lit+95]: it assumes all information is revealed at the next timestep, negating any benefit to information gathering. As we assume the user provides inputs at all times, we gain information automatically when it matters. When the optimal action is the same for multiple goals, we take that action. When the optimal action differs, our model gains information proportional to how suboptimal the user action is for each goal, shifting probability mass towards the user goal, and providing more assistance to that goal.

For shared control teleoperation, explicit information gathering would move the user to a location where their actions between goals were maximally different. Prior works suggest that treating users as an oracle is frustrating [GB11; Ame+14], and this method naturally avoids it.

We evaluated this system in two experiments, comparing our POMDP based method, referred to as policy, to a conventional predict-then-act approach based on Dragan and Srinivasa [DS13a], referred to as blend (fig. 5.3). In our feeding experiment, we additionally compare to direct teleoperation, referred to as direct, and full autonomy, referred to as autonomy.

The blend baseline of Dragan and Srinivasa [DS13a] requires estimating the predictor’s confidence of the most probable goals, which controls how user action and autonomous assistance are arbitrated (fig. 5.3). We use the distance-based measure used in the experiments of Dragan and Srinivasa [DS13a], \( \text{conf} = \max \left(0, 1 - \frac{d}{D}\right) \), where \( d \) is the distance to the nearest target, and \( D \) is some threshold past which confidence is zero.

### 7.1.1 Grasping Experiment

Our first shared-control teleoperation user study evaluates two methods, our POMDP framework and a predict-then-act blending method [DS13a], on the task of object grasping. This task appears broadly in teleoperation systems, appearing in nearly all applications of teleoperated robotic arms. Additionally, we chose this task for its simplicity, evaluating these methods on tasks where direct teleoperation is relatively easy.
7.1.1.1 Metrics

Our experiment aims to evaluate the efficiency and user satisfaction of each method.

Objective measures. We measure the objective efficiency of the system in two ways. Total execution time measures how long it took the participant to grasp an object, measuring the effectiveness in achieving the user’s goal. Total joystick input measures the magnitude of joystick movement during each trial, measuring the user’s effort to achieve their goal.

Subjective measures. We also evaluated user satisfaction with the system through through a seven-point Likert scale survey. After using each control method, we asked users to rate if they would like to use the method. After using both methods, we asked users which they preferred.

7.1.1.2 Hypotheses

Prior work suggests that more autonomy leads to greater efficiency for teleoperated robots [YH11; Lee+12; DS13a; Hau13; Jav+15]. Additionally, prior work indicates that users subjectively prefer more assistance when it leads to more efficient task completion [YH11; DS13a]. Based on this, we formulate the following hypotheses:

- **H1a** Participants using the policy method will grasp objects significantly faster than the blend method.
- **H1b** Participants using the policy method will grasp objects with significantly less control input than the blend method.
- **H1c** Participants will agree more strongly on their preferences for the policy method compared to the blend method.

7.1.1.3 Experiment Design

We set up our experiments with three objects on a table: a canteen, a block, and a cup (fig. 7.2). Users teleoperated a robot arm using two joysticks on a Razer Hydra system. The right joystick mapped to the horizontal plane, and the left joystick mapped to the height. A button on the right joystick closed the hand. Each trial consisted of moving from the fixed start pose, shown in fig. 7.2, to the target object, and ended once the hand was closed.

7.1.1.4 Procedure

We conducted a within-subjects study with one independent variable (control method) that had two conditions (policy, blend). We counteract the effects of novelty and practice by counterbalancing the order.
of conditions. Each participant grasped each object one time for each condition for a total of 6 trials.

We recruited 10 participants (9 male, 1 female), all with experience in robotics, but none with prior exposure to our system. To counterbalance individual differences of users, we chose a within-subjects design, where each user used both systems.

Users were told they would be using two different teleoperation systems, referred to as “method1” and “method2”. Users were not provided any information about the methods. Prior to the recorded trials, users went through a training procedure: First, they teleoperated the robot directly, without any assistance or objects in the scene. Second, they grasped each object one time with each system, repeating if they failed the grasp. Users were then given the option of additional training trials for either system if they wished.

Users then proceeded to the recorded trials. For each system, users picked up each object one time in a random order. Users were told they would complete all trials for one system before the system switched, but were not told the order. However, it was obvious immediately after the first trial started, as the policy method assists from the start pose and blend does not. Upon completing all trials for one system, they were told the system would be switching, and then proceeded to complete all trials for the other system. If users failed at grasping (e.g. they knocked the object over), the data was discarded and they repeated that trial. Execution time and total user input were measured for each trial.

Upon completing all trials, users were given a short survey. For each system, they were asked for their agreement on a 1-7 Likert scale for the following statements:

1. “I felt in control”
2. “The robot did what I wanted”
3. “I was able to accomplish the tasks quickly”
4. “If I was going to teleoperate a robotic arm, I would like to use the system”

They were also asked “which system do you prefer”, where 1 corresponded to blend, 7 to policy, and 4 to neutral. Finally, they were asked to explain their choices and provide any general comments.

7.1.1.5 Results

Users were able to successfully use both systems. There were a total of two failures while using each system - once each because the user attempted to grasp too early, and once each because the user knocked the object over. These experiments were reset and repeated.
We assess our hypotheses using a significance level of $\alpha = 0.05$. For data that violated the assumption of sphericity, we used a Greenhouse-Geisser correction. If a significant main effect was found, a post-hoc analysis was used to identify which conditions were statistically different from each other, with Holm-Bonferroni corrections for multiple comparisons.

**Trial times** and total control input were assessed using a two-factor repeated measures ANOVA, using the assistance method and object grasped as factors. Both trial times and total control input had a significant main effect. We found that our policy method resulted in users accomplishing tasks more quickly, supporting $H_{1a} (F(1, 9) = 12.98, p = 0.006)$. Similarly, our policy method resulted in users grasping objects with less input, supporting $H_{1b} (F(1, 9) = 7.76, p = 0.021)$. See fig. 7.5 for more detailed results.

To assess user preference, we performed a Wilcoxon paired signed-rank test on our survey question asking if they would like to use each system, and a Wilcoxon rank-sum test on the survey question of which system they prefer against the null hypothesis of no preference (value of 4). There was no evidence to support $H_{1c}$.

In fact, our data suggests a trend towards the opposite: that users prefer blend over policy. When asked if they would like to use the system, there was a small difference between methods (blend: $M = 4.90, SD = 1.58$, policy: $M = 4.10, SD = 1.64$). However, when asked which system they preferred, users expressed a stronger preference for blend ($M = 2.90, SD = 1.76$). While these results are not statistically significant according to our Wilcoxon tests and $\alpha = 0.05$, it does suggest a trend towards preferring blend. See fig. 7.3 for results for all survey questions.

We found this surprising, as prior work indicates a strong correlation between task completion time and user satisfaction, even at the cost of control authority, in both shared autonomy [DS13; Hau13] and human-robot teaming [Gom+14] settings. Not only were users faster, but they recognized they could accomplish tasks more quickly (see quickly in fig. 7.3). One user specifically commented that “[Policy] took more practice to learn… but once I learned I was able to do things a little faster. However, I still don’t like feeling it has a mind of its own”.

Users agreed more strongly that they felt in control during blend ($Z = -2.687, p = 0.007$). Interestingly, when asked if the robot did what they wanted, the difference between methods was less drastic. This suggests that for some users, the robot’s autonomous actions were in-line with their desired motions, even though the user did not feel that they were in control.

Users also commented that they had to compensate for policy in...
Figure 7.5: Task completion times and total input for all trials. On the left, box plots for each system. On the right, the time and input of blend minus policy, as a function of the time and total input of blend. Each point corresponds to one trial, and colors correspond to different users. We see that policy was faster ($p < 0.01$) and resulted in less input ($p < 0.05$). Additionally, the difference between systems increases with the time/input of blend.

their inputs. For example, one user stated that “[policy] did things that I was not expecting and resulted in unplanned motion”. This can perhaps be alleviated with user-specific policies, matching the behavior of particular users.

Some users suggested their preferences may change with better understanding. For example, one user stated they “disliked (policy) at first, but began to prefer it slightly after learning its behavior. Perhaps I would prefer it more strongly with more experience”. It is possible that with more training, or an explanation of how policy works, users would have preferred the policy method. We leave this for future work.

7.1.1.6 Examining trajectories

Users with different preferences had very different strategies for using each system. Some users who preferred the assistance policy changed their strategy to take advantage of the constant assistance towards all goals, applying minimal input to guide the robot to the correct goal (fig. 7.6a). In contrast, users who preferred blending were often opposing the actions of the autonomous policy (fig. 7.6b). This suggests the robot was following a strategy different from their own.
7.1.2 Feeding Experiment

Building from the results of the grasping study (section 7.1.1), we designed a broader evaluation of our system. In this evaluation, we test our system in an eating task using a Kinova Mico robot manipulator. We chose the Mico robot because it is a commercially available assistive device, and thus provides a realistic testbed for assistive applications. We selected the task of eating for two reasons. First, eating independently is a real need; it has been identified as one of the most important tasks for assistive robotic arms [Chu+13]. Second, eating independently is hard; interviews with current users of assistive arms have found that people generally do not attempt to use their robot arm for eating, as it requires too much effort [Her+16]. By evaluating our systems on the desirable but difficult task of eating, we show how shared autonomy can improve over traditional methods for controlling an assistive robot in a real-world domain that has implications for people’s quality of life.

We also extended our evaluation by considering two additional control methods: direct teleoperation and full robot autonomy. Direct teleoperation is how assistive robot manipulators like the Mico are currently operated by users. Full autonomy represents a condition in which the robot is behaving “optimally” for its own goal, but does

Figure 7.6: User input and autonomous actions for two users, using both blend and policy. We plot the user input, autonomous assistance with the estimated distribution, and what the autonomous assistance would have been had the predictor known the true goal. We subtract the user input from the assistance when plotting, to show the autonomous action as compared to direct teleoperation. The top 3 figures show each dimension separately. The bottom shows the dot product between the user input and assistance action. (a) This user, who preferred policy, changed their strategy during policy assistance, letting the assistance do the bulk of the work. Note that this user never applied input in the ‘X’ dimension in this or any of their three policy trials, as the assistance always went towards all objects in that dimension. (b) This user, who preferred blend, opposed the autonomous assistance during policy (such as in the ‘X’ dimension) for both the estimated distribution and known goal, suggesting the cost function didn’t accomplish the task in the way the user wanted. Even still, the user was able to accomplish the task faster with the autonomous assistance then blending.
not take the user’s goal into account.

Thus, in this evaluation, we conducted a user study to evaluate four methods of robot control—our POMDP framework, a predict-then-act blending method [DS13a], direct teleoperation, and full autonomy—in an assistive eating task.

7.1.2.1 Metrics

Our experiments aim to evaluate the effectiveness and user satisfaction of each method.

Objective measures. We measure the objective efficiency of the system in four ways. Success rate identifies the proportion of successfully completed trials, where success is determined by whether the user was able to pick up their intended piece of food. Total execution time measures how long it took the participant to retrieve the food in each trial. Number of mode switches identifies how many times participants had to switch control modes during the trial (fig. 5.2). Total joystick input measures the magnitude of joystick movement during each trial. The first two measures evaluate how effectively the participant could reach their goal, while the last two measures evaluate how much effort it took them to do so.

Subjective measures. We also evaluated user satisfaction with the system through subjective measures. After five trials with each control method, we asked users to respond to questions about each system using a seven point Likert scale. These questions, specified in section 7.1.2.4, assessed user preferences, their perceived ability to achieve their goal, and feeling they were in control. Additionally, after they saw all of the methods, we asked users to rank order the methods according to their preference.

7.1.2.2 Hypotheses

As in the previous evaluation, we are motivated by prior work that suggests that more autonomy leads to greater efficiency and accuracy for teleoperated robots [YH11; Lee+12; DS13a; Hau13; Jav+15]. We formulate the following hypotheses regarding the efficiency of our control methods, measured through objective metrics.

H2a Using methods with more autonomous assistance will lead to more successful task completions

H2b Using methods with more autonomous assistance will result in faster task completion

H2c Using methods with more autonomous assistance will lead to fewer mode switches

H2d Using methods with more autonomous assistance will lead to less joystick input
Feeding with an assistive arm is difficult [Her+16], and prior work indicates that users subjectively prefer more assistance when the task is difficult even though they have less control [YH11; DS13a]. Based on this, we formulate the following hypotheses regarding user preferences, measured through our subjective metrics:

\textbf{H2e} \quad \textit{Participants will more strongly agree on feeling in control for methods with less autonomous assistance} \\
\textbf{H2f} \quad \textit{Participants will more strongly agree preference and usability subjective measures for methods with more autonomous assistance} \\
\textbf{H2g} \quad \textit{Participants will rank methods with more autonomous assistance above methods with less autonomous assistance}

Our hypotheses depend on an ordering of “more” or “less” autonomous assistance. The four control methods in this study naturally fall into the following ordering (from least to most assistance): direct teleoperation, blending, policy, and full autonomy. Between the two shared autonomy methods, policy provides more assistance because it creates assistive robot behavior over the entire duration of the trajectory, whereas blend must wait until the intent prediction confidence exceeds some threshold before it produces an assistive robot motion.

7.1.2.3 \textbf{Experimental Design}

To evaluate each robot control algorithm on a realistic assistive task, participants tried to spear bites of food from a plate onto a fork held in the robot’s end effector (fig. 7.7). For each trial, participants controlled the robot through a joystick and attempted to retrieve one of three bites of food on a plate.

Each trial followed a fixed bite retrieval sequence. First, the robot would move to a pose where its wrist-mounted camera could detect bites of food on the plate. This step ensured that the system was robust to bite locations and could operate no matter where on the plate the bites were located. While the camera captured and processed the
scene to identify bite locations, we asked users to verbally specify which bite they wanted to retrieve\(^3\), which allowed us to identify whether people were able to successfully retrieve their target bite.

Next, participants used the joystick to position the robot’s end effector so that the fork was directly above their target bite. Six DOF control was available in three modes of 2 DOF each (fig. 5.2), and participants could switch between modes by pressing a button on the joystick.

Once they had the fork positioned above their target bite, the participant prompted the robot to retrieve the bite by pressing and holding the mode switch button. The robot would then automatically move straight down to the height of the table, spearing the bite on the fork. Finally, the robot automatically served the bite.

### 7.1.2.4 Procedure

We conducted a within-subjects study with one independent variable (control method) that had four conditions (full teleoperation, blend, policy, and full autonomy). Because each participant saw all control methods, we counteract the effects of novelty and practice by fully counterbalancing the order of conditions. Each participant completed five trials for each condition for a total of 20 trials. The bite retrieval sequence described in section 7.1.2.3 was the same in each trial across the four control conditions. The only difference between trials was the control method used for the alignment step, where the fork is positioned above the bite. We measure the metrics discussed in section 7.1.2.2 only during this step.

We recruited 23 able-bodied participants from the local community (11 male, 12 female, ages 19 to 59). After obtaining written consent, participants were given a brief overview of the feeding task, and told the robot may provide help or take over completely. Users then received instruction for teleoperating the system with modal control, and were given five minutes to practice using the robot under direct teleoperation. An eye tracking system was then placed on users for future data analysis, but participant gaze had no effect on the assistance provided by the robot.

As described in section 7.1.2.3, participants used a joystick to spear a piece of food from a plate on a fork held in the robot’s end effector. The different control methods were never explained or identified to users, and were simply referred to by their order of presentation (e.g., “method 1,” “method 2,” etc.). After using each method, users were given a short questionnaire pertaining to that specific method. The questions were:

1. “I felt in control”
2. “The robot did what I wanted”
3. “I was able to accomplish the tasks quickly”
4. “My goals were perceived accurately”
5. “If I were going to teleoperate a robotic arm, I would like to use the system”

These questions are identical to those asked in the previous evaluation (section 7.1.1), with the addition of question 4, which focuses specifically on the user’s goals. Participants were also provided space to write additional comments. After completing all 20 trials, participants were asked to rank all four methods in order of preference and provide final comments.

7.1.2.5 Results

One participant was unable to complete the tasks due to lack of comprehension of instructions, and was excluded from the analysis. One participant did not use the blend method because the robot’s finger broke during a previous trial. This user’s blend condition and final ranking data were excluded from the analysis, but all other data (which were completed before the finger breakage) were used. Two other participants missed one trial each due to technical issues.

Our metrics are detailed in section 7.1.2.1. For each participant, we computed the task success rate for each method. For metrics measured per trial (execution time, number of mode switches, and total
joystick input), we averaged the data across all five trials in each condition, enabling us to treat each user as one independent datapoint in our analyses. Differences in our metrics across conditions were analyzed using a repeated measures ANOVA with a significance threshold of $\alpha = 0.05$. For data that violated the assumption of sphericity, we used a Greenhouse-Geisser correction. If a significant main effect was found, a post-hoc analysis was used to identify which conditions were statistically different from each other, with Holm-Bonferroni corrections for multiple comparisons.

**Success Rate** differed significantly between control methods ($F(2.33,49.00) = 4.57, \ p = 0.011$). Post-hoc analysis revealed that more autonomy resulted in significant differences of task completion between policy and direct ($p = 0.021$), and a significant difference between policy and blend ($p = 0.0498$). All other comparisons were not significant. Surprisingly, we found that policy actually had a higher average task completion ratio than autonomy, though not significantly so. Thus, we found support for $H_2a$ (fig. 7.8a).

**Total execution time** differed significantly between methods ($F(1.89,39.73) = 43.55, \ p < 0.001$). Post-hoc analysis revealed that more autonomy resulted in faster task completion: autonomy condition completion times were faster than policy ($p = 0.001$), blend ($p < 0.001$), and direct ($p < 0.001$). There were also significant differences between policy and blend ($p < 0.001$), and policy and direct ($p < 0.001$). The only pair of methods which did not have a significant difference was blend and direct. Thus, we found support for $H_2b$ (fig. 7.8b).

**Number of mode switches** differed significantly between methods ($F(2.30,48.39) = 65.16, \ p < 0.001$). Post-hoc analysis revealed that more autonomy resulted fewer mode switches between autonomy and blend ($p < 0.001$), autonomy and direct ($p < 0.001$), policy and blend ($p < 0.001$), and policy and direct ($p < 0.001$). Interestingly, there was not a significant difference in the number of mode switches between full autonomy and policy, even though users cannot mode switch when using full autonomy at all. Thus, we found support for $H_2c$ (fig. 7.8c).

**Total joystick input** differed significantly between methods ($F(1.67,35.14) = 65.35, \ p < 0.001$). Post-hoc analysis revealed that more autonomy resulted in less total joystick input between all pairs of methods: autonomy and policy ($p < 0.001$), autonomy and blend ($p < 0.001$), autonomy and direct ($p < 0.001$), policy and blend ($p < 0.001$), policy and direct ($p < 0.001$), and blend and direct ($p = 0.026$). Thus, we found support for $H_2d$ (fig. 7.8d).

User reported subjective measures for the survey questions are assessed using a Friedman’s test and a significance threshold of $\alpha =$
0.05. If significance was found, a post-hoc analysis was performed, comparing all pairs with Holm-Bonferroni corrections.

User agreement on control differed significantly between methods, $\chi^2(3) = 15.44, p < 0.001$, with more autonomy leading to less feeling of control. Post-hoc analysis revealed that all pairs were significant, where autonomy resulting in less feeling of control compared to policy ($p < 0.001$), blend ($p = 0.001$), and direct ($p < 0.001$). Policy resulted in less feeling of control compared to blend ($p < 0.001$) and direct ($p = 0.008$). Blend resulted in less feeling of control compared to direct ($p = 0.002$). Thus, we found support for $H_{2e}$.

User agreement on preference and usability subjective measures sometimes differed significantly between methods. User agreement on liking differed significantly between methods, $\chi^2(3) = 8.74, p = 0.033$. Post-hoc analysis revealed that between the two shared autonomy methods (policy and blend), users liked the more autonomous method more ($p = 0.012$).

User agreement on their perceived ability for achieving goals quickly also differed significantly between methods, $\chi^2(3) = 11.90, p = 0.008$. Post-hoc analysis revealed that users felt they could achieve their goals more quickly with policy than with blend ($p = 0.010$) and direct ($p = 0.043$). We found no significant differences for our other measures. Thus, we find partial support for $H_{2f}$ (fig. 7:10).

Ranking differed significantly between methods, $\chi^2(3) = 10.31, p = 0.016$. Again, post-hoc analysis revealed that between the two shared autonomy methods (policy and blend), users ranked the more autonomous one higher ($p = 0.006$). Thus, we find support for $H_{2g}$. As for the like rating, we also found that on average, users ranked direct teleoperation higher than both blend and full autonomy, though not significantly so (fig. 7:8f).
7.1.2.6 Discussion

The robot in this study was controlled through a 2 DOF joystick and a single button, which is comparable to the assistive robot arms in use today.

As expected, we saw a general trend in which more autonomy resulted in better performance across all objective measures (task completion ratio, execution time, number of mode switches, and total joystick input), supporting H2a–H2d. We also saw evidence that autonomy decreased feelings of control, supporting H2e. However, it improved people’s subjective evaluations of usability and preference, particularly between the shared autonomy methods (policy and blend), supporting H2f and H2g. Most objective measures (particularly total execution time, number of mode switches, and total joystick input) showed significant differences between all or nearly all pairs of methods, while the subjective results were less certain, with significant differences between fewer pairs of methods.

We can draw several insights from these findings. First, autonomy improves people’s performance on a realistic assistive task by requiring less physical effort to control the robot. People use fewer mode switches (which require button presses) and move the joystick less in the more autonomous conditions, but still perform the task more quickly and effectively. For example, in the policy method, 8 of our 22 users did not use any mode switches for any trial, but this method yielded the highest completion ratio and low execution times. Clearly, some robot autonomy can benefit people’s experience by reducing the amount of work they have to do.

Interestingly, full autonomy is not always as effective as allowing the user to retain some control. For example, the policy method had a slightly (though not significantly) higher average completion ratio than the full autonomy method. This appears to be the result of
users fine-tuning the robot’s end effector position to compensate for small visual or motor inaccuracies in the automatic bite localization process. Because the task of spearing relatively small bites of food requires precise end effector localization, users’ ability to fine-tune the final fork alignment seems to benefit the overall success rate. Though some users were able to achieve it, our policy method isn’t designed to allow this kind of fine-tuning, and will continually move the robot’s end effector back to the erroneous location against the user’s control. Detecting when this may be occurring and decreasing assistance would likely enhance people’s ability to fine-tune alignment, and improve their task completion rate even further.

Given the success of blending in previous studies [Li+11; CD12; DS13a; Mue+15; Gop+16], we were surprised by the poor performance of blend in our study. We found no significant difference for blending over direct teleoperation for success rate, task completion time, or number of mode switches. We also saw that it performed the worst among all methods for both user liking and ranking. One possible explanation is that blend spent relatively little time assisting users (fig. 7.8e). For this task, the goal predictor was unable to confidently predict the user’s goal for 69% of execution time, limiting the amount of assistance (fig. 7.9c). Furthermore, the difficult portion of the task—rotating the fork tip to face downward—occurred at the beginning of execution. Thus, as one user put it “While the robot would eventually line up the arm over the plate, most of the hard work was done by me.” In contrast, user comments for shared autonomy indicated that “having help earlier with fork orientation was best.” This suggests that the magnitude of assistance was less important then assisting at a time that would have been helpful. And in fact, assisting only during the portion where the user could do well themselves resulted in additional frustration.

Although worse by all objective metrics, participants tended to prefer direct teleoperation over autonomy. This is not entirely surprising, given prior work where users expressed preference for more control [Kim+12]. However, for difficult tasks like this one, users in prior works tend to favor more assistance [YH11; DS13a]. Many users commented that they disliked autonomy due to the lack of item selection, for example, “While [autonomy] was fastest and easiest, it did not account for the marshmallow I wanted.” Another user mentioned that autonomy “made me feel inadequate.”

We also found that users responded to failures by blaming the system, even when using direct teleoperation. Of the eight users who failed to successfully spear a bite during an autonomous trial, five users commented on the failure of the algorithm. In contrast, of the 19 users who had one or more failure during teleoperation, only two
commented on their own performance. Instead, users made comments about the system itself, such as how the system “seemed off for some reason” or “did not do what I intended.” One user blamed their viewpoint for causing difficulty for the alignment, and another the joystick. This suggests that people are more likely to penalize autonomy for its shortcomings than their own control. Interestingly, this was not the case for the shared autonomy methods. We find that when users had some control over the robot’s movement, they did not blame the algorithm’s failures (for example, mistaken alignments) on the system.

7.2 Human-Robot Teaming

In human-robot teaming, the user and robot want to achieve a set of related goals. Formally, we assume a set of user goals $g^u \in G^u$ and robot goals $g^r \in G^r$, where both want to achieve all goals. However, there may be constraints on how these goals can be achieved (e.g. user and robot cannot simultaneously use the same object [HB07]). We apply a conservative model for these constraints through a goal restriction set $\mathcal{R} = \{(g^u, g^r) : \text{Cannot achieve } g^u \text{ and } g^r \text{ simultaneously}\}$. In order to efficiently collaborate with the user, our objective is to simultaneously predict the human’s intended goal, and achieve a robot goal not in the restricted set. We remove the achieved goals from their corresponding goal sets, and repeat this process until all robot goals are achieved.

The state $x$ corresponds to the state of both the user and robot, where $u$ affects the user portion of state, and $a$ affects the robot portion. The transition function $T(x' \mid x, u, a)$ deterministically transitions the state by applying $u$ and $a$ sequentially.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Autonomy Policy</th>
<th>Autonomy Blend</th>
<th>Autonomy Direct</th>
<th>Policy Blend</th>
<th>Policy Direct</th>
<th>Blend Direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>0.050</td>
<td>0.021</td>
<td>NS</td>
</tr>
<tr>
<td>Completion Time</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>NS</td>
</tr>
<tr>
<td>Mode Switches</td>
<td>NS</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>NS</td>
</tr>
<tr>
<td>Control Input</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>$0.004$</td>
</tr>
<tr>
<td>Ranking</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>0.006</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>Like Rating</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>0.012</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>Control Rating</td>
<td>$&lt;0.001$</td>
<td>.001</td>
<td>$&lt;0.001$</td>
<td>$&lt;0.001$</td>
<td>0.008</td>
<td>.002</td>
</tr>
<tr>
<td>Quickly Rating</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>0.010</td>
<td>0.043</td>
<td>NS</td>
</tr>
</tbody>
</table>

Table 7.1: Post-Hoc p-value for every pair of algorithms for each hypothesis. For Success rate, completion time, mode switches, and total joystick input, results are from a repeated measures ANOVA. For like rating and ranking, results are from a Wilcoxon signed-rank test. All values reported with Holm-Bonferroni corrections.
For prediction, we used the same cost function for $C_k^u$ as in our shared teleoperation experiments (section 7.1). Let $d$ be the distance between the robot state $x' = T^u(x, u)^4$ and target $\kappa$:

$$C_k^u(x, u) = \begin{cases} 
\alpha & d > \delta \\
\frac{\alpha}{2}d & d \leq \delta 
\end{cases}$$

Which behaves identically to our shared control teleoperation setting: when the distance is far away from any target ($d > \delta$), probability shifts towards goals relative to how much progress the user makes towards them. When the user stays close to a particular target ($d \leq \delta$), probability mass shifts to that goal, as the cost for that goal is less than all others.

Unlike our shared control teleoperation setting, our robot cost function does not aim to achieve the same goal as the user, but rather any goal not in the restricted set. As in our shared autonomy framework, let $g$ be the user’s goal. The cost function for a particular user goal is:

$$C_r^u(x, u, a) = \min_{g^* \text{ s.t. } (g, g^*) \notin R} C_k^u(x, a)$$

Where $C_k^u$ uses the cost for each target $C_k^u$ to compute the cost function as described in section 6.4. Additionally, note that the min over cost functions looks identical to the min over targets to compute the cost for a goal. Thus, for deterministic transition functions, we can use the same proof for computing the value function of a goal given the value function for all targets (section 6.4.1) to compute the value function for a robot goal given the value function for all user goals:

$$V_r^u(x) = \min_{g^* \text{ s.t. } (g, g^*) \notin R} V_k^u(x)$$

This simple cost function provides us a baseline for performance. We might expect better collaboration performance by incorporating costs for collision avoidance with the user [MB13; LS15], social acceptability of actions [Sis+07], and user visibility and reachability [Sis+10; PA10; Mai+11]. We use this cost function to test the viability of our framework as it enables closed-form computation of the value function.

This cost and value function causes the robot to go to any goal currently in it’s goal set $g^* \in G^*$ which is not in the restriction set of the user goal $g$. Under this model, the robot makes progress towards goals that are unlikely to be in the restricted set and have low cost-to-go. As the form of the cost function is identical to that which we used in shared control teleoperation, the robot behaves similarly: making constant progress when far away ($d > \delta$), and slowing down for alignment when near ($d \leq \delta$). The robot terminates and completes the task once some condition is met (e.g. $d \leq \epsilon$).

---

4 We sometimes instead observe $x'$ directly (e.g. sensing the pose of the user hand).
Hindsight Optimization for Human-Robot Teaming

Similar to shared control teleoperation, we believe hindsight optimization is a suitable POMDP approximation for human-robot teaming. The efficient computation enables us to respond quickly to changing user goals, even with continuous state and action spaces. For our formulation of human-robot teaming, explicit information gathering is not possible: As we assume the user and robot affect different parts of state space, robot actions are unable to explicitly gather information about the user’s goal. Instead, we gain information freely from user actions.

7.2.1 Human-Robot Teaming Experiment

We apply our shared autonomy framework to a human-robot teaming task of gift-wrapping, where the user and robot must both perform a task on each box to be gift wrapped. Our goal restriction set enforces that they cannot perform a task on the same box at the same time.

In a user study, we compare three methods: our shared autonomy framework, referred to as policy, a standard predict-then-act system, referred to as plan, and a non-adaptive system where the robot executes a fixed sequence of motions, referred to as fixed.

7.2.1.1 Metrics

Task fluency involves seamless coordination of action. One measure for task fluency is the minimum distance between the human and robot end effectors during a trial. This was measured automatically by a Kinect mounted on the robot’s head, operating at 30Hz. Our second fluency measure is the proportion of trial time spent in collision. Collisions occur when the distance between the robot’s end effector and the human’s hand goes below a certain threshold. We determined that 8cm was a reasonable collision threshold based on observations before beginning the study.

Task efficiency relates to the speed with which the task is completed. Objective measures for task efficiency were total task duration for robot and for human, the amount of human idle time during the trial, and the proportion of trial time spent idling. Idling is defined as time a participant spends with their hands still (i.e., not completing the task). For example, idling occurs when the human has to wait for the robot to stamp a box before they can tie the ribbon on it. We only considered idling time while the robot was executing its tasks, so idle behaviors that occurred after the robot was finished stamping the boxes—which could not have been caused by the robot’s behavior—
were not taken into account.

We also measured subjective human satisfaction with each method through a seven-point Likert scale survey evaluating perceived safety (four questions) and sense of collaboration (four questions). The questions were:

1. “HERB was a good partner”
2. “I think HERB and I worked well as a team”
3. “I’m dissatisfied with how HERB and I worked together”
4. “I trust HERB”
5. “I felt that HERB kept a safe distance from me”
6. “HERB got in my way”
7. “HERB moved too fast”
8. “I felt uncomfortable working so close to HERB”

### 7.2.1.2 Hypotheses

We hypothesize that:

**H3a** Task fluency will be improved with our policy method compared with the plan and fixed methods

**H3b** Task efficiency will be improved with our policy method compared with the plan and fixed methods

**H3c** People will subjectively prefer the policy method to the plan or fixed methods

### 7.2.1.3 Experimental Design

We developed a gift-wrapping task (fig. 7.11). A row of four boxes was arranged on a table between the human and the robot; each box had a ribbon underneath it. The robot’s task was to stamp the top of each box with a marker it held in its hand. The human’s task was to tie a bow from the ribbon around each box. By nature of the task, the goals had to be selected serially, though ordering was unspecified. Though participants were not explicitly instructed to avoid the robot, tying the bow while the robot was stamping the box was challenging because the robot’s hand interfered, which provided a natural disincentive toward selecting the same goal simultaneously.

### 7.2.1.4 Implementation

We implemented the three control methods on HERB Srinivasa et al. [Sri+12], a bi-manual mobile manipulator with two Barrett WAM arms. A Kinect was used for skeleton tracking and object detection. Motion planning was performed using CHOMP, except for our policy method in which motion planning works according to section 6.1.
The stamping marker was pre-loaded in HERB’s hand. A stamping action began at a home position, the robot extended its arm toward a box, stamped the box with the marker, and retracted its arm back to the home position.

To implement the fixed method, the system simply calculated a random ordering of the four boxes, then performed a stamping action for each box. To implement the predict-then-act method, the system ran the human goal prediction algorithm from section 6.3.1 until a certain confidence was reached (50%), then selected a goal that was not within the restricted set $R$ and performed a stamping action on that goal. There was no additional human goal monitoring once the goal action was selected. In contrast, our policy implementation performed as described in section 7.2, accounting continually for adapting human goals and seamlessly re-planning when the human’s goal changed.

7.2.1.5 Procedure

We conducted a within-subjects study with one independent variable (control method) that had 3 conditions (policy, plan, and fixed). Each performed the gift-wrapping task three times, once with each robot control method. To counteract the effects of novelty and practice, we counterbalanced on the order of conditions.

We recruited 28 participants (14 female, 14 male; mean age 24, SD 6) from the local community. Each participant was compensated $5 for their time. After providing consent, participants were introduced to the task by a researcher. They then performed the three gift-wrapping trials sequentially. Immediately after each trial, before continuing to the next one, participants completed an eight question Likert-scale survey to evaluate their collaboration with HERB on that trial. At the end of the study, participants provided verbal feedback about the three methods. All trials and feedback were video recorded.

7.2.1.6 Results

Two participants were excluded from all analyses for noncompliance during the study (not following directions). Additionally, for the fluency objective measures, five other participants were excluded due to Kinect tracking errors that affected the automatic calculation of minimum distance and time under collision threshold. Other analyses were based on video data and were not affected by Kinect tracking errors.

We assess our hypotheses using a significance level of $\alpha = 0.05$. For data that violated the assumption of sphericity, we used a
Greenhouse-Geisser correction. If a significant main effect was found, a post-hoc analysis was used to identify which conditions were statistically different from each other, with Holm-Bonferroni corrections for multiple comparisons.

To evaluate \( H_3a \) (fluency), we conducted a repeated measures ANOVA testing the effects of method type (policy, plan, and fixed) on our two measures of human-robot distance: the minimum distance between participant and robot end effectors during each trial, and the proportion of trial time spent with end effector distance below the 8cm collision threshold (fig. 7.12). The minimum distance metric was not significant \( (F(2, 40) = 1.405, p = 0.257) \). However, proportion of trial time spent in collision was significantly affected by method type \( (F(2, 40) = 3.639, p = 0.035) \). Interestingly, the policy method never entered under the collision threshold. Post-hoc pairwise comparisons with a Holm-Bonferroni correction reveal that the policy method yielded significantly \( (p = 0.027) \) less time in collision than the plan method (policy \( M = 0.0\% \), \( SD = 0 \); plan \( M = 0.44\% \), \( SD = 0.7 \)).

Therefore, \( H_3a \) is partially supported: the policy method actually yielded no collisions during the trials, whereas the plan method yielded collisions during 0.4% of the trial time on average. This confirms the intuition behind the differences in the two methods: the policy continually monitors human goals, and thus never collides with the human, whereas the plan method commits to an action once a confidence level has been reached, and is not adaptable to changing human goals.

To evaluate \( H_3b \) (efficiency), we conducted a similar repeated measures ANOVA for the effect of method type on task durations for robot and human (fig. 7.14), as well as human time spent idling (fig. 7.13). Human task duration was highly variable and no significant effect for method was found \( (F(2, 50) = 2.259, p = 0.115) \). On the other hand, robot task duration was significantly affected by method condition \( (F(2, 50) = 79.653, p < 0.001) \). Post-hoc pairwise comparisons with a Bonferroni correction reveal that differences between all conditions are significant at the \( p < 0.001 \) level. Unsurprisingly, robot task completion time was shortest in the fixed condition, in which the robot simply executed its actions without monitoring human goals \( (M = 46.4s, SD = 3.5s) \). It was significantly longer with the plan method, which had to wait until prediction reached a confidence threshold to begin its action \( (M = 56.7s, SD = 6.0) \). Robot task time was still longer for the policy method, which continually monitored human goals and smoothly replanned motions when required, slowing down the overall trajectory execution \( (M = 64.6s, SD = 5.3) \).

Total task duration (the maximum of human and robot time) also
showed a statistically significant difference ($F(2, 50) = 4.887, p = 0.012$). Post-hoc tests with a Bonferroni-Holm correction show that both fixed ($M = 58.6s, SD = 14.1$) and plan ($M = 60.6s, SD = 7.1$) performed significantly ($p = 0.026$ and $p = 0.032$, respectively) faster than policy ($M = 65.9s, SD = 6.3$). This is due to the slower execution time of the policy method, which dominates the total execution time.

Total idle time was also significantly affected by method type ($F(2, 50) = 3.809, p = 0.029$). Post-hoc pairwise comparisons with Bonferroni correction reveal that the policy method yielded significantly ($p = 0.048$) less idle time than the fixed condition (policy $M = 0.46s, SD = 0.93$, fixed $M = 1.62s, SD = 2.1$). Idle time percentage (total idle time divided by human trial completion time) was also significant ($F(2, 50) = 3.258, p = 0.047$). Post-hoc pairwise tests with Bonferroni-Holm correction finds no significance between pairs. In other words, the policy method performed significantly better than the fixed method for reducing human idling time, while the plan method did not.

Therefore, $H_{3b}$ is partially supported: although total human task time was not significantly influenced by method condition, the total robot task time and human idle time were all significantly affected by which method was running on the robot. The robot task time was slower using the policy method, but human idling was significantly reduced by the policy method.

To evaluate $H_{3c}$ (subjective responses), we first conducted a Chronbach’s alpha test to assure that the eight survey questions were internally consistent. The four questions asked in the negative (e.g., “I’m dissatisfied with how HERB and I worked together”) were reverse coded so their scales matched the positive questions. The result of the test showed high consistency ($\alpha = 0.849$), so we proceeded with our analysis by averaging together the participant ratings across all eight questions.

During the experiment, participants sometimes saw collisions with the robot. We predict that collisions will be an important covariate on the subjective ratings of the three methods. In order to account for whether a collision occurred on each trial in our within-subjects design, we cannot conduct a simple repeated measures ANOVA. Instead, we conduct a linear mixed model analysis, with average rating as our dependent variable; method (policy, plan, and fixed), collision (present or absent), and their interaction as fixed factors; and method condition as a repeated measure and participant ID as a covariate to account for the fact that participant ratings were not independent across the three conditions. Table 7.2 shows details of the scores for each method broken down by whether a collision occurred.
We found that collision had a significant effect on ratings $(F(1, 47.933) = 6.055, p = 0.018)$, but method did not $(F(1, 47.933) = 0.312, p = 0.733)$. No interaction was found. In other words, ratings were significantly affected by whether or not a participant saw a collision, but not by which method they saw independent of that collision. Therefore, $\textbf{H}_3c$ is not directly supported. However, our analysis shows that collisions lead to poor ratings, and our results above show that the policy method yields fewer collisions. We believe a more efficient implementation of our policy method to enable faster robot task completion, while maintaining fewer collisions, may result in users preferring the policy method.

### 7.3 Discussion

In chapter 6, we presented a method for shared autonomy that does not rely on predicting a single user goal, but assists for a distribution over goals. Our motivation was a lack of assistance when using predict-then-act methods - in our own experiment (section 7.1.2), resulting in no assistance for 69% of execution time. To assist for any distribution over goals, we formulate shared autonomy as a POMDP with uncertainty over user goals. To provide assistance in real-time over continuous state and action spaces, we used hindsight optimization [Lit+95; Cho+00; Yoo+08] to approximate solutions. We tested our method on two shared-control teleoperation scenarios, and one human-robot teaming scenario. Compared to predict-then-act methods, our method achieves goals faster, requires less user input, decreases user idling time, and results in fewer user-robot collisions.

In our shared control teleoperation experiments, we found user preference differed for each task, even though our method outperformed a predict-then-act method across all objective measures for both tasks. This is not entirely surprising, as prior works have also been mixed on whether users prefer more control authority or better task completion [YH11; Kim+12; DS13a]. In our studies, users tended to prefer a predict-then-act approach for the simpler grasping scenario, though not significantly so. For the more complex eating task, users significantly preferred our shared autonomy method to a predict-then-act method. In fact, our method and blending were

<table>
<thead>
<tr>
<th>Method</th>
<th>No Collision</th>
<th>Collision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean (SD)</td>
<td>N</td>
</tr>
<tr>
<td>Fixed</td>
<td>5.625 (1.28)</td>
<td>14</td>
</tr>
<tr>
<td>Plan</td>
<td>5.389 (1.05)</td>
<td>18</td>
</tr>
<tr>
<td>Policy</td>
<td>5.308 (0.94)</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 7.2: Subjective ratings for each method condition, separated by whether a collision occurred during that trial.
the only pair of algorithms that had a significant difference across all objective measures and the subjective measuring of like and rank (table 7.1).

However, we believe this difference of rating cannot simply be explained by task difficulty and timing, as the experiments had other important differences. The grasping task required minimal rotation, and relied entirely on assistance to achieve it. Using blending, the user could focus on teleoperating the arm near the object, at which point the predictor would confidently predict the user goal, and assistance would orient the hand. For the feeding task, however, orienting the fork was necessary before moving the arm, at which point the predictor could confidently predict the user goal. For this task, predict-then-act methods usually did not reach their confidence threshold until users completed the most difficult portion of the task - cycling control modes to rotate and orient the fork\(^5\). This inability to confidently predict a goal until the fork was oriented caused predict-then-act methods to provide no assistance for the first 29.4 seconds on average - which is greater than the total average time of our method (18.5s). We believe users were more willing to give up control authority if they did not need to do multiple mode switches and orient the fork, which subjectively felt more tedious than moving the position.

For human-robot teaming, the total task time was dominated by the robot, with the user generally finishing before the robot. In situations like this, augmenting the cost function to be more aggressive with robot motion, even at the cost of responsiveness to the user, may be beneficial. Additionally, incorporating more optimal robot policies may enable faster robot motions within the current framework.

Finally, though we believe these results show great promise for shared control teleoperation and teaming, we note users varied greatly in their preferences and desires. Prior works in shared control teleoperation have been mixed on whether users prefer control authority or more assistance [YH11; Kim+12; DS13a]. Our own experiments were also mixed. Even within a task, users had high variance, with users fairly split for grasping (fig. 7.3), and a high variance for user responses for full autonomy for eating (fig. 7.10). For teaming, users were similarly mixed in their rating for an algorithm depending on whether or not they collided with the robot (table 7.2). This variance suggests a need for the algorithm to adapt to each individual user, learning their particular preferences.

\(^5\) These mode switches have been identified as a significant contributor to operator difficulty and time consumption [Her+16]
Our experiments in chapter 7 indicate that users had varied preferences for when and how they would like to be assisted. In section 7.1.1.6, we examined how users with different preferences changed their inputs in different ways for each assistance strategy. Users who preferred blend often opposed assistance, providing inputs to counteract the shared autonomy system even when it made progress for their goal. Users who preferred policy often provided no inputs while assistance was going towards their goal, letting the shared autonomy system do the bulk of the work. See fig. 7.6.

In this chapter, we provide a method for learning a model of user behavior during assistance, and incorporating this model into cost minimization. These models are learned for each individual user as they use our shared autonomy system. The shared autonomy system then updates with the new predictor, which we use at the next iteration. This process repeats. See fig. 8.1.

In order to learn how a user responds to assistance quickly, we use our previously described predictor (section 6.3.1) as a prior. Intuitively, we believe that user behavior during assistance will closely resemble how they act without it. Our method learns a new distribution of user actions by minimizing the Kullback-Leibler (KL) divergence with a distribution for user behavior without assistance. Using this prior enables us to learn a good model with fewer data points.

With this predictor, we perform a short rollout of actions the user and robot would select during shared autonomy. If we predict an assistance action will cause a user to oppose assistance, we account for the additional cost the user would incur, and penalize that action. If we predict an assistance action will enable a user to achieve their goal without providing input, we predict they incur less cost, and prefer that action.

We implement this method for a discrete gridworld scenario with modal control [Her+16] (section 5.3). To simulate the cost of mode

Figure 8.1: Our algorithm pipeline. Given the current predictor, we utilize the framework from chapter 6 to provide assistance under goal uncertainty. Once the system achieves the user’s goal, we add this trial to all previous data. We use this entire dataset, along with a prior model of user behavior with no assistance, to learn a new predictor. This induces a different assistance policy, which we use at the next iteration. We repeat this process.
switching in shared-control teleoperation, users experienced a time
delay when they switched modes. At each timestep, our shared au-
tonomy system can provide assistance by automatically switching
modes for the user. In a study with users on mechanical turk, we
find that using this new predictor enabled users to achieve their goals
while incurring less cost, and with less fighting against assistance.

8.1 Learning the User Policy with Assistance

Intuitively, we believe that user behavior during assistance will re-
semble the way they act without assistance. Let $p_{me}$ be a predictor
of user behavior without assistance, e.g. learned through maximum
entropy inverse optimal control (MaxEnt IOC) [Zie+08]. Notably, this
predictor is goal-driven, as it models the user as an agent stochas-
tically minimizing a cost function for a particular goal. Thus, this
captures how a user would attempt to achieve a goal without assist-
tance.

To learn a predictor with assistance, we employ the principle of
minimum cross-entropy [SJ80], matching the observed data while min-
imizing the Kullback-Leibler (KL) divergence to this prior distribu-
tion.

Let $f_{xu}$ be some features of user input $u$ and trajectory so far $\xi$. Let
$\overline{f_{xu}}$ be the average feature observed in the data:

$$\arg \min_{p_{kl}} KL(p_{kl} \| p_{me})$$

s.t. $\sum_{\xi \in \text{Data}} P(\xi) \sum_u p_{kl}(u|\xi)f_{xu} = \overline{f_{xu}}$

That is, the average feature of the data $\overline{f_{xu}}$ should match the expected
feature predicted by our learned distribution $p_{kl}$ on the trajectories
observed in the data.

As our prior models how a user would achieve a goal, we choose
features $f_{xu}$ to model how users respond to assistance. For computa-
tional purposes, we follow Nikolaidis et al. [Nik+16] and utilize
a bounded memory model, incorporating features of only a short
history $k$. For these past $k$ timesteps, we select features to indicate
whether the user opposed assistance. We also use a feature for the
difference between the optimal cost-to-go for the user acting alone
minus if assistance were optimal for that goal. This feature captures
how likely a user is to select actions which rely on assistance, going
to states that are useful if assistance reacts optimally.
8.2 Assistance Action Selection

Given a predictor, we can compute the value function for a known goal similarly to our formulation in chapter 6. Once we have the value function for each goal, we combine through QMDP/Hindsight Optimization [Cho+00; Yoo+08; Lit+95] to select actions under goal uncertainty. Once the user supplies an input, we use the new predictor to update our distribution over goals.

Following eq. (6.1), we compute the value function for a single goal:

\[ V^\pi_t(s) = \mathbb{E} \left[ \sum_t C^t(s_t, u_t, a_t) \mid s_0 = s \right] \]

where \( u_t \sim p_{kl}(\cdot \mid s_t, f_{st}^z) \)
\( a_t \sim \pi^t(\cdot \mid s_t, u_t) \)
\( s_{t+1} \sim T(\cdot \mid s_t, u_t, a_t) \)

Where \( f_{st}^z \) are trajectory features of up until time \( t \). In section 6.2, we discuss various approximations to this, where we do not need to roll out \( u_t \) for all time steps. In particular, we utilized the robot takes over approximation in our experiments (chapter 7), which corresponds to:

\[ V_{\text{section 6.2}}(s) = \min_{\pi^t} \mathbb{E} \left[ \sum_t C^t(s_t, 0, a_t) \mid s_0 = s \right] \]  

This assumption was made for computational purposes - rolling out the user and robot policies while selecting assistance actions is computationally difficult. However, if we wish to incorporate the user model into action selection - for example, to avoid fighting the user - we must relax this assumption.

Instead, we approximate by rolling out our policy and predictor for a short horizon, and utilize a heuristic thereafter:

\[ V(s) \approx \min_{\pi^t} \mathbb{E} \left[ \sum_t C^t(s_t, u_t, a_t) + \tilde{V}(x_T) \mid s_0 = s \right] \]

Where \( \tilde{V} \) is some estimate of the cost-to-go, e.g. \( \tilde{V} = V_{\text{section 6.2}} \). In practice, we let the horizon \( T \) equal the history our predictor uses.

8.3 Iterating Learning and Policy Updates

The above learning problem assumes that that the training and testing distributions are independent and identically distributed (iid) - that is, the training histories \( \zeta \in \text{Data} \) are drawn from the same distribution of histories we will see during testing. However, updating
our model of the user causes our shared autonomy policy to change, and therefore the histories to be different, violating this assumption.

This common problem in reinforcement learning is addressed by the DAgger method [Ros+11; RB12]. The solution is intuitively simple - iteratively update your policy, get a new set of data with the current policy, and train the predictor with all data, including data from previous policies. See fig. 8.1. In addition to providing theoretical no-regret guarantees in this setting, this has the empirical benefit of continuously adapting to the user’s behavior during assistance.

8.4 Experiments

We implement this method for a discrete gridworld scenario intended to mimic shared-control teleoperation for modal control [Her+16] (section 5.3). Briefly, modal control addresses the problem of controlling high degree of freedom systems with lower degree of freedom inputs by defining a discrete set of control modes, each of which controls a subset of the robot degrees of freedom.

We aimed to evaluate each user on the same set of maps using three methods in randomized order:

1. Direct, where no assistance was provided
2. MaxEnt, where we used a predictor which assumed no assistance (chapter 6)
3. Rollout, where we learn a predictor with assistance (section 8.1) and a short rollout to estimate the new value function (section 8.2)

Prior to MaxEnt or Rollout, users went through a different set of maps, which we use to learn the user policy $P_{me}$ based on MaxEnt IOC [Zie+08]. When using the Rollout method, we initialize our predictor by running the optimization of section 8.1 with our data without assistance. As described in section 8.3, we update this predictor after every iteration, adding the trial data to our current dataset, computing $P_{kl}$ using our $P_{me}$ as a prior, use this to compute a new $V$, which leads to a new shared autonomy assistance policy. See fig. 8.1. We repeat this process through all Rollout trials between every iteration.

In this experiment, our MaxEnt and Rollout could only provide assistance by automatically switching modes for users. In assistive robotics, mode switching is a key cause of both cognitive load and execution time, consuming about 17.4% of execution time [Herlant et al. [Her+16]. Thus, automatically mode switching provides useful non-intrusive assistance, and has been shown to be an effective form of assistance [Her+16]. Unlike the continuous control space from chapter 7, this discrete assistance cannot easily be modelled by

---

1 We can view our learning problem as one of system identification, where user inputs cause stochastic transitions and costs. Ross and Bagnell [RB12] show that the DAgger method provides guarantees and good performance in this setting.
blending [DS13a].

As the robot cannot complete the task on its own, we use the user takes over approximation, detailed in section 6.2, when estimating the value functions $V_{\text{section 6.2}}$ (eq. (8.1)) and $V$ (eq. (8.2)).

8.4.1 Metrics

Our experiments aim to evaluate the effectiveness and user satisfaction of each method.

Objective measures. We measure the objective efficiency of the system in two ways. Cost measures the total cost a user incurs in order to achieve their goal. This cost was proportional to time, as users experienced a time delay proportional to the cost$^2$. Assistance Fight Ratio measures how often the user and shared autonomy system both mode switch at the same iteration, undoing the action while causing the user to incur a large cost. We assess this metric only for methods that can provide assistance.

Subjective measures. We also evaluated user satisfaction with the system through subjective measures. After all trials with each method, we asked users to respond to questions about each system using a five point Likert scale. These questions, specified in section 8.4.3, assessed a user’s perceived ability to achieve their goal, and feeling of whether they were in control.

8.4.2 Hypotheses

We aim to evaluate through objective and subjective measures if users were able to achieve their goals better using our prediction with assistance framework. We formulate the following hypothesis regarding the efficiency of our methods, based on our objective measures:

$H_{4a}$ Participants will achieve their goals while incurring less cost when using assistance with better predictors

$H_{4b}$ Participants will fight with assistance methods less while using assistance with better predictors

In line with our hypotheses in section 7.1.2.2, we also formulate the following hypotheses about the subjective measures:

$H_{4c}$ Participants will more strongly agree on feeling in control for methods with less autonomous assistance

$H_{4d}$ Participants will more strongly agree preference and usability subjective measures for assistance methods with better predictors

8.4.2.1 Experiment Design

Users saw a map and controls, as shown in fig. 8.2. Their objective was to navigate to the displayed goal using the on-screen controls.

Figure 8.2: Our experimental setup. (a) Users must navigate the robot to the specified goal, which the system does not know a priori. The grid includes fast-moving squares (white), slow-moving squares (green), and walls (blue). (b) Some maps were designed to distinguish between users who were willing to rely on assistance to automatically mode switch for a shorter path. (c) User controls and displayed target, where they move the robot through modal control: left-right and up-down, and can switch modes by rotating the robot. The active control mode is depicted by the orientation of the robot, and the opaque controls.

$^a$ Users frequently took breaks, so we do not assess task completion time.
To simulate modal control, the robot had two control modes: left-right and up-down. This was indicated by the orientation of the robot itself, and the opacity of control buttons. When navigating, users experienced a *time delay* proportional to the cost. The time delay for moving on white and green squares was 80ms and 240ms, respectively. The additional time delay for mode switching on either square was 800ms. Blue squares on the screen represented walls, which the users could not move onto.

We manually created 26 maps, each having between 1 and 3 goals. Each map and goal is treated as a separate trial, with a total of 53 trials. For each user, we randomly assign 26 of these trials to training the predictor for MaxEnt IOC, and 27 trials for testing all methods. Maps consisted of a mix of simpler maps (e.g. fig. 8.2a), and maps aimed at distinguishing between users who liked to rely on assistance and those who preferred direct teleoperation (e.g. fig. 8.2b).

### 8.4.3 Procedure

We conducted a within-subjects study with one independent variable (control method) that had three conditions (*Direct*, *MaxEnt*, *Rollout*). Because each participant saw all control methods, we counteract the effects of novelty and practice by counterbalancing the order of conditions.

We recruited users through Amazon’s Mechanical Turk service. In order to ensure reliable results, all participants were located in the USA to avoid language barriers, and we required an approval rate of over 95%. We asked a control question to ensure they paid attention to the task, and eliminated users with incorrect answers to this question. In addition, as our task was very long, many users did not complete it. With those users removed, we ended up with 55 users who satisfied this criteria.

Users were first given instructions and 3 randomly selected practice trials with no assistance. After that, all users completed all trials for training the MaxEnt IOC predictor. Once the predictor was learned, users used our three control methods (*Direct*, *MaxEnt*, *Rollout*) in random order. Prior to using either *MaxEnt* or *Rollout*, users also had 3 random trials with assistance, using the *MaxEnt* method. The ordering and trials for these methods as identical. Upon completing all trials for one method, users completed a short survey with the following questions:

1. “I felt in *control*”
2. “I was able to accomplish tasks *quickly*”
3. “The robot did what I *wanted*”
4. “If I were going to use a system, I would *like* to use the system”
8.4.4 Results

Our metrics are detailed in section 8.4.1. For each participant, we computed the total cost accumulated for each method for each trial. We average across all 27 test trials in each condition, enabling us to treat each user as one independent datapoint in our analyses. Differences in our metrics across conditions were analyzed with a significance threshold of $\alpha = 0.05$. If a significant main effect was found, a post-hoc analysis was used to identify which conditions were statistically different from each other, with Holm-Bonferroni corrections for multiple comparisons.

We analyzed cost with a repeated measures ANOVA. As our data violated the assumption of sphericity, we used a Greenhouse-Geisser correction. We found that cost differed significantly between methods ($F(1.311, 70.813) = 509.34, p < 0.0001$). Post-hoc analysis revealed a significant differences of cost between all pairs: Direct and MaxEnt ($p < 0.0001$), Direct and Rollout ($p < 0.0001$), and MaxEnt and Rollout ($p = 0.039$). We found that using the Rollout policy resulted in users completing their task with less cost on average (18.108 ± 0.408) than MaxEnt (18.804 ± 0.399) and Direct (35.706 ± 0.569). Thus, we found support for $H_4a$. As Rollout learns through iterations, we also plot how the average cost changes across trials. See fig. 8.3.

As there are only two conditions for the assistance fight ratio, we analyzed with a paired sample t-test. We found a significant difference between MaxEnt (0.0566 ± 0.0097) and Rollout (0.0331 ± 0.0048), $t(54) = 2.512, p = 0.015$. Thus, we found support...
for \( H_4b \). As \textsc{Rollout} learns through iterations, we also plot how the assistance fight ratio changes across trials. See \textit{fig. 8.3}.

User reported subjective measures for the survey questions are assessed using a Friedman’s test and a significance threshold of \( \alpha = 0.05 \). If significance was found, a post-hoc analysis was performed, comparing all pairs using a Wilcoxon signed-rank test with Holm-Bonferroni corrections.

User agreement on \textit{control} differed significantly between methods, \( \chi^2(2) = 19.169, p < 0.001 \). Post-hoc analysis revealed that only one pair was significant, with users feeling more in control with \textsc{Direct} than \textsc{Rollout} (\( p = 0.005 \)). Thus, we find partial support for \( H_4c \).

User agreement on their perceived ability to achieve goals \textit{quickly} differed significantly between methods, \( \chi^2(2) = 9.529, p = 0.009 \). Post-hoc analysis revealed that both assistance methods were perceived to enable users to achieve goals more quickly than \textsc{Direct}, using either \textsc{MaxEnt} (\( p = 0.017 \)) or \textsc{Rollout} (\( p = 0.020 \)). We found no significant difference between \textsc{MaxEnt} and \textsc{Rollout}. Thus, we find partial support for \( H_4d \).

We found significant differences for the subjective measures of \textit{want} and \textit{like}. See \textit{fig. 8.4}.

\subsection*{8.5 Discussion}

In this chapter, we presented a method for modelling how users change their behavior during shared autonomy, and using this model to provide better shared autonomy. We were motivated by our observations that users had varied preferences, and these preferences correlated with how users reacted to assistance. In \textit{section 8.1}, we presented a method to learn a distribution over user actions by using a predictor learned without shared autonomy as a prior, enabling learning with few iterations. In \textit{section 8.2}, we demonstrated how to incorporate this user model into action selection. In \textit{section 8.3}, we discuss how this affects learning our predictor, and present our solu-
tion for learning through multiple iterations using DAgger [Ros+11; RB12]. Finally, in section 8.4, we showed that this method enabled users to achieve their goals while incurring less cost, and while fighting the assistance strategy less.

While these results are a promising first step to incorporating models of how assistance affects user action selection, there is still much to be explored. Computationally, our method here is limited to discrete problems or sampling techniques, due to the rollout required for estimating the value function.

From an ideological point of view, we learn our predictor as if the user action selection depends only on a short history of states, user actions, and robot actions. In reality, we see that users will learn and adapt their behavior through iterations (fig. 8.3). We hope to explore methods that can model this user learning, and use it to provide better shared autonomy assistance.
9

Final Thoughts

This thesis presented methods for acting under uncertainty that are goal-directed, dealing with uncertainty only as required to achieve a goal. They are connected by the insight that not all uncertainty impedes gaining utility - even when uncertainty is high, there often exist actions which gain utility over the entire distribution. This insight enabled us to formulate and implement methods for active information gathering and shared autonomy for real-world problems.

For active information gathering, we first drew a connection between information gathering in robotics and adaptive submodularity (chapter 3), enabling us to provide near-optimality guarantees with an efficient lazy-greedy algorithm. This method gathered uncertainty indiscriminately, without considering the goal. To alleviate this, we formulated the Decision Region Determination (DRD) problem, with the goal of reducing uncertainty just enough to make a decision (chapter 4). We presented two adaptive submodular methods this problem, each providing rigorous guarantees and improved empirical performance compared to state-of-the-art active information gathering methods. Experimentally, we found this method outperformed those which reduce uncertainty indiscriminately, such as the commonly used reduction of Shannon entropy [Cas+96; Bur+97; Fox+98; Bou+02; Zhe+05; Fu+07; Eri+08; Hsi+08; Heb+13; Sad+16b].

We next formulated shared autonomy as a general problem of acting under uncertainty (chapter 6). This formulation enabled us to use hindsight optimization to make progress for a distribution of user goals, rather than requiring the confident prediction in a single goal prior to assisting. In user studies for both shared-control teleoperation and human-robot teaming, we showed our method enabled faster performance with less user effort compared to methods which predict a single user goal (chapter 7). Though objective measures of performance were improved, we found users were mixed in their preference. To address this, we extended our shared autonomy framework to learn user-specific models for how they react to
assistance, and utilize this to minimize user cost (chapter 8).

Based on our experiences developing and implementing these ideas, we now discuss exciting areas of future work.

9.1 Active Information Gathering Future Work

Decisions as tests

Our Decision Region Determination (DRD) framework modelled information gathering with three distinct components: hypotheses representing the possible state of the world, tests to gather information, and decisions valid for specific subsets of hypotheses. We showed how a variety of information gathering problems could be split into these three components (table 4.2).

However, decisions and tests need not be separate - in some situations, decisions can also be used to gather information. For example, in touch-based localization (sections 3.3, 4.3.2 and 4.5.1), we could attempt to accomplish the task prior to reducing uncertainty to a decision region. If we succeed, utility is gained and the problem is solved. If not, an observation is received, uncertainty is updated, and the method continues.

We could consider adding the set of decisions to tests, and terminating if a decision succeeds. However, this would be suboptimal - it would not capture the cost difference between reducing uncertainty to a decision region and then performing the decision, and accomplishing the task. One promising avenue for future work would be an extension of our framework to incorporate the utility gained by performing a decision, while maintaining adaptive submodularity. This would provide improved performance and near-optimality guarantees compared to the policy that can gather information and gain utility simultaneously.

Learning through trials

In our experiments, we found that our method often selected the same tests, especially at the beginning of a trial. This is no surprise - the initial uncertainty was similar\(^1\), and some outcomes occur with much higher probability. Performing the same test and receiving a similar outcome often resulted in selecting the same next test.

Incorporating a method which learns commonly occurring sequences of tests, observations, and the next selected test would enable us to sometimes skip test selection and gather information faster. Choudhury et al. [Cho+17a] present a promising line of work with a similar idea, using computationally intensive information gathering offline as an expert, and imitating this policy for online use.

\(^1\) We generally sampled uncertainty from a continuous distribution to start, so while the set of hypotheses was different, they modelled the same underlying distribution.
However, this method relies on learning a complicated policy for any situation.

Instead, it might be better to combine this learned policy with the ability to recompute test selection when necessary, e.g. when the observed outcomes are not similar enough to data. Ideally, learning is focused on commonly occurring sequences, enabling fast computation and good performance for most scenarios.

We could also consider only learning this set for a small number of initial tests, as each additional observation makes it less likely that the entire sequence was observed. Additionally, computation is faster after some information has been gathered, as the expectations are computed over smaller sets.

9.2 Shared Autonomy Future Work

Capturing user preference and feeling of control

One of our unexpected findings in our shared autonomy experiments was the variance in user preference. For disabled users, it has been hypothesized that users view assistive robots as extensions of themselves, enabling them to interact with the world in a way that they could not [Kim+12]. When objective performance was equal, users tended to prefer full control [Kim+12]. However, prior work has found that users subjectively prefer more assistance when it leads to more efficient task completion [YH11; DS13a].

Our findings here were mixed - for the easier grasping experiment, users tended to prefer less assistance, though not significantly so (section 7.1.1). For the more difficult feeding experiment, users preferred our shared autonomy framework to blending or direct tele-operation (section 7.1.2). For our gift-wrapping experiment, users preferred our shared autonomy method only when we conditioned on whether they collided with the robot (section 7.2.1). This suggests that user preference varies not simply on the amount of autonomy, but the situation and task at hand, and the kind of assistance.

One potential next step would be to learn a model of when certain assistance actions cause users to feel less control authority, and penalize those actions differently. For example, users seemed to appreciate extra-modal assistance in our feeding experiment, either because it was more helpful, or because assistance in a mode the user does not control affects their feeling of control authority less. Investigating this difference, and altering the assistance cost based on the findings, may enable greater user satisfaction.

Another variable that affected user preference was the task itself, where users prefer assistance for more difficult tasks. This is not sur-
prising, as we would expect users to give up control authority more willingly for tasks they may not prefer. One possibility indicator of how willing a user is to accept assistance is the difference in cost with and without assistance - that is, the difference in value function from a state if assistance were present or not. We utilized this difference as a feature in our user-specific adaptation experiments (chapter 8), and found it helped performance.

**User-specific adaptation**

Recent works in shared autonomy, described in this thesis and otherwise [YH11; Lee+12; DS13a; Hau13], suggests that individual users respond to assistance differently. New work by Nikolaidis et al. [Nik+16; Nik+17c] captures these ideas through the user’s adaptability, a parameter representing how likely a user is to change their strategy based on the robot’s actions. Similarly, Sadigh et al. [Sad+16b] explore how to learn a model for how a robot’s actions affect users, and use this model for robot action planning for autonomous driving. Sadigh et al. [Sad+16a; Sad+17] also explore how to actively gather information about the user’s state and preferences.

However, we believe more general models exploring how autonomous assistance affects users should be explored and incorporated into action selection. In particular, learning user-specific models can be greatly beneficial, as we observed high variance of preference across users in our experiments (sections 7.1.1, 7.1.2 and 7.2.1). We began exploring this idea in chapter 8, and how we could incorporate this model into providing assistance. However, this work was limited to a simple gridworld example due to computational limitations. Scaling these ideas to larger domains may lead to even more improved performance, as assistance is often more helpful in more complicated problems.

**Incorporating better cost functions**

In our experiments, we used a simple distance-based cost function, for which we could compute value functions in closed form. This enabled us to compute prediction and assistance 50 times a second, making the system feel responsive and reactive. However, this simple cost function could only provide simple assistance, with the objective of minimizing the time to reach a goal. Ideally, more complicated notions of the user’s cost would be incorporated into action selection. Importantly, these methods should avoid performing rollouts online, as we require very efficient policies for use in shared autonomy.

Recent successes in reinforcement learning, and in particular deep learning, have largely focused on learning policies directly, instead
of value functions. Finn et al. [Fin+16] show a method for simultaneously learning a cost function and policy through Maximum Entropy Inverse Optimal Control (MaxEnt IOC) [Zie+08], which would enable more complicated policies to be learned. Interestingly, their experiments suggest that directly using the learned policy outperforms using the learned cost function. While using the learned policy directly is applicable for learning a single robot policy to imitate demonstrations, using hindsight optimization over a distribution of user goals requires learning a value function for each goal. An interesting avenue of future exploration would be to take learned policies and compute value functions for them, enabling our framework to utilize hindsight optimization over these policies.

9.3 Acting Under Uncertainty Future Work

Ideally, methods for acting under uncertainty simultaneously optimize over both information gathering and task accomplishing during the selection of each action. This is captured generally by POMDP solvers [Roy+05; SS05; Kur+08; SV10; Sha+12; Som+13; Sei+15]. However, optimizing for both simultaneously is often intractable.

Instead, the work in this thesis presents methods on two extremes of acting under uncertainty. For active information gathering, we focused on gathering information efficiently prior to making a decision, believing goal-directed progress could not be made until some uncertainty was resolved. For shared autonomy, our method instead tries to gain as much utility as possible under uncertainty, hoping uncertainty resolves itself over time. We believe many problems of acting under uncertainty falls into one of these categories.

We chose these extremes due to the contexts of each particular problem, where we believed one approach presented the right trade-offs. Ideally, a method would be imbued with some notion of which is useful, and could select one method for action selection to optimize over until some criteria was met. For example, our DRD framework could use for a decision region some set of uncertainty for which hindsight optimization would be successful. Methods which could compute some criteria like this may enable systems to both gather information and act under uncertainty without solving the full POMDP.
Appendix

Here we provide the proofs and details for our theorems throughout this thesis.

A. Hypothesis Pruning Proofs

We present proofs for our Hypothesis Pruning (HP) theorems from chapter 3, showing the guarantee of near-optimal performance. To do so, we prove our metrics are adaptive submodular, strongly adaptive monotone, and self-certifying. Note that the bounds on adaptive submodular functions require that observations are not noisy - that is, for a fixed hypothesis \( h \), a test can result in only one observation deterministically. In our case, we would like to model a distribution of observations for each test \( t \) and hypothesis \( h \), as our sensors are noisy. Thus, we first construct a non-noisy problem by creating many weighted “noisy” copies of each hypothesis \( h \). We then show how to compute our objective on the original problem. Finally, we prove our performance guarantee.

A.1.1 Constructing the Non-Noisy Problem

Similar to previous approaches in active learning, we construct a non-noisy problem by creating “noisy” copies of each hypothesis \( h \) for every possible noisy observation \([Gol+10; Bel+12]\). Let \( \Omega_t(h) = \{\hat{h}_1, \ldots, \hat{h}_K\} \) be the function that creates \( K \) noisy copies for test \( t \). Here, the original probability of \( h \) is distributed among all the noisy copies, \( P(h) = \sum_{\hat{h} \in \Omega_t(h)} P(\hat{h}) \). For convenience, we will also consider overloading \( \Omega \) to take sets of tests, and sets of realizations. Let \( \Omega_T(h) \) recursively apply \( \Omega \) for each \( t \in T \). That is, if we let \( T = \{t_1, t_2, \ldots\} \) we apply \( \Omega_{t_1} \) to \( h \), then \( \Omega_{t_2} \) to every output of \( \Omega_{t_1}(h) \), and so on. Note that we still have \( P(h) = \sum_{\hat{h} \in \Omega_T(h)} P(\hat{h}) \). Additionally, let \( \Omega_T(H) \) apply \( \Omega_T \) to each \( h \in H \) and combine the set.

The probability of each noisy copy comes from our weighting functions defined in section 3.2.2:

\[
\Omega_T(h) = \{\hat{h}_1, \ldots, \hat{h}_K\} \\
P(\hat{h}) = P(h) \sum_{h' \in \Omega_t(h)} \frac{\omega_{t}(h)}{\omega_{t'}(h)} \\
P(\hat{h}) = P(h) \prod_{t \in T} \frac{\omega_{t}(h)}{\sum_{h' \in \Omega_t(h)} \omega_{t'}(h)} (\text{multiple tests})
\]
For simplicity, we also assume that the maximum value of our weighting function is equal to one for any test. We note that our weighting functions in section 3.2.2 have this property for the non-noisy observation where \( t_h = t_h \):

\[
\max_{\hat{h} \in \Omega_T(h)} \omega_h(t_h) = 1 \quad \forall h, t
\]  

(A.1)

We build our set of non-noisy realizations \( \hat{H} = \Omega_T(H) \). Our objective function is over \( \hat{h} \), specifying the probability mass removed from the original problem. One property we desire is if our observations are consistent with one noisy copy of \( h \), then we keep some proportion of all of the noisy copy (proportional to our weighting function \( \omega_{HP} \) or \( \omega_{WHP} \)). In our HP algorithm for example, if any noisy copy of \( h \) remains, the objective function acts as if all of the probability mass remains.

We define our utility function here in a slightly different form: if the evidence \( S \) is generated by running tests \( T \), with observations generated by \( \hat{h} \), let \( \hat{f}(T, \hat{h}) = f(S) \). Note that we can always do this mapping, as \( \hat{h} \) generates observations deterministically, and we can set \( T = S_T \). We compute our objective as:

\[
f(\hat{T}, \hat{h}) = 1 - \sum_{h \in H} \left( \prod_{t \in T} \frac{P(h)}{\max P(\Omega_T(h))} \right) \left( \sum_{\hat{h}' \in \Omega_T(h)} \sum_{t' \in T} \delta_{\hat{h}' t'} P(\hat{h}') \right)
\]

Where \( \delta_{\hat{h}' t'} \) is the Kronecker delta function, equal to 1 if \( t_h = t_h \) and 0 otherwise, \( h \) is the original hypothesis from which \( \hat{h} \) was produced, and \( \max P(\Omega_T(h)) \) is the highest probability of the “noisy” copies. By construction, any test will keep at most \( \max P(\Omega_T(h)) \) probability mass per test, since at most one noisy copy from \( \Omega_T(h) \) will be consistent with the observation. Intuitively, multiplying by \( \frac{P(h)}{\max P(\Omega_T(h))} \) will make it so if we kept the highest weighted noisy copy of \( h \), our objective would be equivalent to keeping the entire hypothesis \( h \).

\[
f(\hat{T}, \hat{h}) = 1 - \sum_{h \in H} \left( \prod_{t \in T} \frac{P(h)}{\max P(\Omega_T(h))} \right) \left( \sum_{\hat{h}' \in \Omega_T(h)} \sum_{t' \in T} \delta_{\hat{h}' t'} P(\hat{h}') \right)
\]

\[
= 1 - \sum_{h \in H} \left( \prod_{t \in T} \frac{P(h)}{\max P(\Omega_T(h))} \right) \left( \sum_{\hat{h}' \in \Omega_T(h)} \sum_{\hat{h}'' \in \Omega_T(h)} \sum_{t' \in T} \delta_{\hat{h}' t'} P(\hat{h}'') \right)
\]

\[
= 1 - \sum_{h \in H} \left( \prod_{t \in T} \frac{P(h)}{\max P(\Omega_T(h))} \right) \left( \sum_{\hat{h}' \in \Omega_T(h)} \sum_{\hat{h}'' \in \Omega_T(h)} \delta_{\hat{h}' t'} P(\hat{h}'') \right)
\]

Here, we separate the recursive splitting over the hypothesis \( h \) into those split based on tests in \( T \) and those split from other tests. Since \( \prod_{t \in T} \delta_{\hat{h}' t'} \) only depends on the response to tests in \( T \), it only depends on noisy copies made from \( \Omega_T \). Thus, we can factor those out. Additionally, we marginalize over the copies of \( \hat{h}' \) as \( \sum_{\hat{h}' \in \Omega_T(h')} P(\hat{h}'') = P(\hat{h}') \). Overall, this simplification enables us to only consider the copies from the
tests in $T$. We further simplify:

$$
\hat{f}(T, \hat{h}) = 1 - \sum_{h \in H} \left( \prod_{t \in T} \frac{P(h)}{\max_{\hat{h}' \in \Omega_T(h)} P(\hat{h}')} \right) \left( \sum_{\hat{h}' \in \Omega_T(h)} P(\hat{h}') \prod_{t \in T} \delta_{h't'} \right)
$$

$$
= 1 - \sum_{h \in H} \left( \prod_{t \in T} P(h) \left( \frac{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{h't'}(t_h)}{\max_{\hat{h}' \in \Omega_T(h)} \omega_{h't'}(t_h) P(h)} \right) \right) \left( \sum_{\hat{h}' \in \Omega_T(h)} P(\hat{h}') \prod_{t \in T} \delta_{h't'} \right)
$$

$$
= 1 - \sum_{h \in H} \left( \prod_{t \in T} \prod_{\hat{h}' \in \Omega_T(h)} \omega_{h't'}(t_h) \right) \left( \sum_{\hat{h}' \in \Omega_T(h)} P(\hat{h}') \prod_{t \in T} \frac{\omega_{h't'}(t_h)}{\sum_{\hat{h}'' \in \Omega_T(h)} \omega_{h''t'}(t_h)} \prod_{t \in T} \delta_{h't'} \right)
$$

$$
= 1 - \sum_{h \in H} P(h) \sum_{\hat{h}' \in \Omega_T(h)} \left( \prod_{t \in T} \omega_{h't'}(t_h) \right) \prod_{t \in T} \delta_{h't'}
$$

$$
= 1 - \sum_{h \in H} P(h) \sum_{\hat{h}' \in \Omega_T(h)} \prod_{t \in T} \omega_{h't'}(t_h) \delta_{h't'}
$$

Where eq. (A.2) corresponds to plugging in the value of $\Omega_T(h)$, eq. (A.3) used eq. (A.1) above. Now we consider how the function $\Omega$ generates noisy copies. We require that exactly one noisy copy $\hat{h}' \in \Omega_T(h)$ agree with every observation received so far, and thus only one term will have a nonzero product $\prod_{t \in T} \delta_{h't'}$. We defer further specific details of $\Omega$ until the next section. We get:

$$
\hat{f}(T, \hat{h}) = 1 - \sum_{h \in H} P(h) \sum_{\hat{h}' \in \Omega_T(h)} \prod_{t \in T} \omega_{h't'}(t_h) \delta_{h't'}
$$

$$
= 1 - \sum_{h \in H} P(h) \prod_{t \in T} \omega_{h't'}(t_h)
$$

At this point we can see how this equals the objective function $f(S)$ from section 3.2.2. Here again, we let $S$ be the evidence gathered if we ran tests $T$ and received observations generated by $\hat{h}$. We get:

$$
\hat{f}(T, \hat{h}) = 1 - \sum_{h \in H} P(h) \prod_{t \in T} \omega_{h't'}(t_h)
$$

$$
= 1 - \sum_{h \in H} \omega_{h't'}(t_h)
$$

$$
= 1 - M_S
$$

$$
= f(S)
$$

### A.1.2 Observation Probabilities

To compute expected marginal utilities, we will need to define our space of possible observations, and the corresponding probability for these observations. Recall that $S = \{S_T, S_O\}$, where $S_T$ are the tests in $S$, and $S_O$ are the observations. We call $P(t_H = o | S)$ the probability of receiving observation $o$ from
performing test $t$ conditioned on evidence $S$, over all hypotheses $\mathcal{H}$. Intuitively, this will correspond to how much probability mass agrees with the observation. More formally:

$$P(t_H = o|S) \propto \sum_{h \in \mathcal{H}} \sum_{\hat{h} \in \Omega_T(h)} P(\hat{h}) \delta_{\hat{h},o} \prod_{\{i,o\} \in S} \delta_{i,o}$$

Similar to before, we will be able to consider noisy copies made from only tests in $S_T$ and $t$ (the derivation follows exactly as in appendix A.1.1). This will simplify to:

$$P(t_H = o|S) \propto \sum_{h \in \mathcal{H}} \sum_{\hat{h} \in \Omega_T(h)} P(\hat{h}) \delta_{\hat{h},o} \prod_{\{i,o\} \in S} \delta_{i,o}$$

$$= \sum_{h \in \mathcal{H}} P(h) \sum_{\hat{h} \in \Omega_T(h)} \sum_{\hat{h}' \in \Omega_T(\hat{h})} \left( \frac{\omega_{t_H}(t_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \delta_{\hat{h}',o} \right) \left( \prod_{\{i,o\} \in S} \frac{\omega_{t_H}(\hat{h}_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \delta_{i,o} \right)$$

The first term in parenthesis comes from the weighting of performing test $t$ and receiving observation $o$, where we would like the only noisy copy of $h$ that agrees with that observation. The second term comes from that same operation, but for all tests and observations in $S$. Again, we know by construction that exactly one noisy copy agrees with all observations. Hence, we can write this as:

$$P(t_H = o|S) \propto \sum_{h \in \mathcal{H}} P(h) \sum_{\hat{h} \in \Omega_T(h)} \sum_{\hat{h}' \in \Omega_T(\hat{h})} \left( \frac{\omega_{t_H}(t_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \delta_{\hat{h}',o} \right) \left( \prod_{\{i,o\} \in S} \frac{\omega_{t_H}(\hat{h}_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \delta_{i,o} \right)$$

$$= \sum_{h \in \mathcal{H}} P(h) \left( \frac{\omega_o(t_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \right) \left( \prod_{\{i,o\} \in S} \frac{\omega_o(\hat{h}_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \right)$$

Finally, we would also like for $\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)$ to be constant for all tests $t$ and realizations $h$, enabling us to factor those terms out. To approximately achieve this, we generate noisy copies by discretizing the trajectory uniformly along the path, and generate a noisy copy of each hypothesis $h$ at every discrete location. We approximate our hypothesis to be set at one of the discrete locations, such that $t_h$ is equal to the nearest discrete location. For many locations, the weighting function will be less than some negligible $\varepsilon$. Let there be $K$ discrete locations for any $h$ and $t$ where $\omega_{t_h} > \varepsilon$. We say that $|\Omega_t(h)| = K$. Thus, we can fix the value of $\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h) = \kappa \ \forall t,h$. Note that we also need to be consistent with observations corresponding to not contacting an object anywhere along the trajectory. Therefore, we also consider $K$ noisy copies for this case. Under these assumptions, we can further simplify:

$$P(t_H = o|S) \propto \sum_{h \in \mathcal{H}} P(h) \left( \frac{\omega_o(t_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \right) \left( \prod_{\{i,o\} \in S} \frac{\omega_o(\hat{h}_h)}{\sum_{\hat{h}' \in \Omega_T(h)} \omega_{t_H}(\hat{h}_h)} \right)$$

$$\approx \sum_{h \in \mathcal{H}} P(h) \left( \frac{\omega_o(t_h)}{\kappa} \right) \left( \prod_{\{i,o\} \in S} \frac{\omega_o(\hat{h}_h)}{\kappa} \right)$$

$$\propto \sum_{h \in \mathcal{H}} P(h) \omega_o(t_h) \prod_{\{i,o\} \in S} \omega_o(\hat{h}_h)$$

$$= \sum_{h \in \mathcal{H}} w_S(h) \omega_o(t_h)$$

$$= m_{S,t,o}$$
Finally, we need to normalize all observations to get:

\[ P(t_H = o | S) = \frac{m_{S,t,o}}{\sum_{o' \in O_t} m_{S,t,o'}} \]

Where \( O_t \) consists of all the discrete stopping points sampled, and the \( K \) observations for non-contact.

### A.1.3 Proving the Bound

We showed that our utility function is equivalent to the mass removed from the original \( H \):

\[ f(S) = 1 - M_S. \]

This function can utilize either of the two reweighting functions \( \omega^{HP} \) or \( \omega^{WHP} \) defined in section 3.2.2. Our objective is a truncated version of this: \( f_Q(S) = \min \{ Q, f(S) \} \), where \( Q \) is the target value for how much probability mass we wish to remove. We assume that the set of all tests \( T \) is sufficient such that, if \( \hat{f}(T, \hat{h}) \geq Q, \forall \hat{h} \in \hat{H} \). Note that adaptive monotone submodularity is preserved by truncation, so showing these properties for \( f \) implies them for \( f_Q \).

Using our utility function and observation probability, it is not hard to see that the expected marginal benefit of an test is given by:

\[
\Delta_f(t | S) = \mathbb{E}_{o \in O_t} [f(S \cup \{(t,o)\}) - f(S) | S] \\
= \sum_{o \in O_t} P(o | S) [(1 - m_{S,t,o}) - (1 - M_S)] \\
= \sum_{o \in O_t} \frac{m_{S,t,o}}{\sum_{o' \in O_t} m_{S,t,o'}} [M_S - m_{S,t,o}] \tag{A.5}
\]

This shows the derivation of the marginal utility, as defined in section 3.2.2. We now provide the proof for theorem 3, by showing that this utility function is adaptive submodular, strongly adaptive monotone, and self-certifying:

**Lemma 1.** Let \( \Lambda \subseteq T \), which result in partial realizations \( S_{\Lambda} \). Our objective function defined above is strongly adaptive monotone.

**Proof.** We need to show that for any test and observation, our objective function will not decrease in value. Intuitively, our objective is strongly adaptive monotone, since we only remove probability mass and never add hypotheses. More formally:

\[
f(S) \leq \mathbb{E} [f(S \cup \{(t,o)\}) | S, \hat{h}(t) = o] \\
\Leftrightarrow 1 - M_S \leq 1 - M_{S \cup \{(t,o)\}} \\
\Leftrightarrow 1 - M_S \leq 1 - m_{S,t,o} \\
\Leftrightarrow m_{S,t,o} \leq M_S \\
\Leftrightarrow \sum_{h \in H} p_S(h) \omega_o(t_{H'}) \leq \sum_{h \in H} p_S(h)
\]

As noted before, both of the weighting functions defined in Section 3.2.2 never have a value greater than one. Thus each term in the sum from the LHS is smaller than the equivalent term in the RHS. \( \Box \)

**Lemma 2.** Let \( S \subseteq S' \subseteq T \times O \). Our objective function defined above is adaptive submodular.
Proof. For the utility function $f$ to be adaptive submodular, it is required that the following holds over expected marginal utilities:

$$\Delta(a|S') \leq \Delta(a|S)$$

$$\sum_{o \in O} \frac{m_{S', t, o}}{\sum_{o' \in O'} m_{S', t, o'}} [M_{S'} - m_{S', t, o}] \leq \sum_{o \in O} \frac{m_{S, t, o}}{\sum_{o' \in O'} m_{S, t, o'}} [M_S - m_{S, t, o}]$$

We simplify notation a bit for the purposes of this proof. As the test is fixed, we will replace $O_t$ with $O$. For a fixed evidence $S$ and test $t$, let $m_{S, t, o} = m_o$. Let $k_o = m_{S, t, o} - m_{S', t, o}$, which represents the difference of probability mass remaining between partial realizations $S'$ and $S$ if we performed test $t$ and received observation $o$. We note that $k_o \geq 0 \ \forall o$, which follows from the strong adaptive monotonicity, and $k_o \leq m_{S, t, o}$ which follows from $m_{S', t, o} \geq 0$. Rewriting the equation above:

$$\sum_{o \in O} \frac{m_o - k_o}{\sum_{o' \in O'} m_{o'} - k_{o'}} [M_{S'} - m_0 + k_0] \leq \sum_{o \in O} \frac{m_o}{\sum_{o' \in O'} m_{o'}} [M_S - m_0]$$

$$\Leftrightarrow \left( \sum_{o \in O} M_{S'} m_0 - m_0^2 + m_o k_o - M_{S'} k_o + m_o k_o - k_o^2 \right) \left( \sum_{o' \in O'} m_{o'} \right) \leq \left( \sum_{o \in O} M_S m_0 - m_0^2 \right) \left( \sum_{o' \in O'} m_{o'} - k_{o'} \right)$$

$$\Leftrightarrow \sum_{o \in O} \sum_{o' \in O'} M_{S'} (m_o m_{o'} - m_o k_o) + 2m_o m_{o'} k_o - m_o k_o^2 \leq \sum_{o \in O} \sum_{o' \in O'} M_S (m_o m_{o'} - m_o k_{o'}) + m_o k_{o'}^2$$

We also note that $M_S - M_{S'} \geq \max_{o \in O} (k_o)$. That is, the total difference in probability mass is greater than or equal to the difference of probability mass remaining if we received any single observation, for any observation.

$$\Leftrightarrow \sum_{o \in O} \sum_{o' \in O'} 2m_o m_{o'} k_o - m_{o'} k_o^2 \leq \sum_{o \in O} \sum_{o' \in O'} (M_S - M_{S'})(m_o m_{o'} - m_o k_{o'}) + m_o k_{o'}^2$$

$$\Leftrightarrow \sum_{o \in O} \sum_{o' \in O'} 2m_o m_{o'} k_o - m_{o'} k_o^2 \leq \sum_{o \in O} \sum_{o' \in O'} \max_{o \in O} (k_o) (m_o m_{o'} - m_o k_{o'}) + m_o k_{o'}^2$$

$$\Leftrightarrow \sum_{o \in O} \sum_{o' \in O'} 2m_o m_{o'} k_o - m_{o'} k_o^2 \leq \sum_{o \in O} \sum_{o' \in O'} \max_{o \in O} (k_{o'}) (m_o m_{o'} - m_o k_{o'}) + m_o k_{o'}^2$$

In order to show the inequality for the sum, we will show it holds for any pair $o, o'$. First, if $o = o'$, than we have an equality and it holds trivially. For the case when $o \neq o'$, we assume that $k_o > k_{o'}$ WLOG, and show the inequality for the sum:

$$2m_o m_{o'} (k_o + k_{o'}) - m_{o'} k_{o'}^2 - m_o k_o^2 \leq 2m_o m_{o'} k_o - m_o k_o k_o - m_{o'} k_{o'}^2 + m_{o'}^2 k_o$$

$$\Leftrightarrow 2m_o m_{o'} k_o - m_{o'} k_{o'}^2 \leq m_o^2 k_o + m_{o'}^2 k_o - m_o k_o k_{o'}$$

$$\Leftrightarrow 0 \leq k_{o'} (m_o - m_{o'})^2 - (k_o - k_{o'}) k_o (m_o - m_{o'}) + (k_o - k_{o'}) m_{o'} (m_{o'} - k_{o'})$$

$$\Leftrightarrow 0 \leq k_{o'} (m_o - m_{o'})^2 - (k_o - k_{o'}) k_{o'} (m_o - m_{o'}) + (k_o - k_{o'}) k_{o'} (m_{o'} - k_{o'})$$

We split into 3 cases:

$A.1.4 \ k_{o'} = 0$

This holds trivially, since the RHS is zero
A.1.5 \( k_{o'} \neq 0, m_o \leq 2m_{o'} - k_{o'} \)

Since \( k_{o'} \neq 0 \), we can rewrite:

\[
0 \leq (m_o - m_{o'})^2 - (k_o - k_{o'})(m_o - m_{o'}) + (k_o - k_{o'})(m_{o'} - k_{o'}) \\
\iff 0 \leq -(k_o - k_{o'})(m_o - m_{o'}) + (k_o - k_{o'})(m_{o'} - k_{o'}) \\
\iff (m_o - m_{o'}) \leq (m_{o'} - k_{o'})
\]

Which follows from the assumption for this case.

A.1.6 \( m_o \geq 2m_{o'} - k_{o'} \)

We show this step by induction. Let \( m_o = 2m_{o'} - k_{o'} + x, x \geq 0 \)

**Base Case:** \( x = 0 \), which we showed in the previous case.

**Induction** Assume this inequality holds for \( m_o = 2m_{o'} - k_{o'} + x \). Let \( m_o' = m_o + 1 \). We now show that this holds for \( m_o' \):

\[
0 \leq (m_o' - m_{o'})^2 - (k_o - k_{o'})(m_o' - m_{o'}) + (k_o - k_{o'})(m_{o'} - k_{o'}) \\
\iff 0 \leq (m_o - m_{o'} + 1)^2 - (k_o - k_{o'})(m_o - m_{o'} + 1) + (k_o - k_{o'})(m_{o'} - k_{o'}) \\
\iff 0 \leq 2m_o - 2m_{o'} + 1 - k_o + k_{o'} \quad \text{by inductive hypothesis} \\
\iff 0 \leq m_o + 1 - k_o \quad \text{by assumption from case} \\
\iff 0 \leq 1
\]

And thus, we have shown the inequality holds for any pair \( o, o' \). ■

Finally, it is easy to see that the sum can be decomposed into pairs of \( o, o' \). Therefore, we can see the inequality over the sum also holds.

**Lemma 3.** The utility function \( f \) defined above is self-certifying.

**Proof.** Golovin and Krause [GK11] define an instance as self-certifying if whenever the maximum value is achieved for the utility function \( f \), it is achieved for all realizations consistent with the observation. See Golovin and Krause [GK11] for a more rigorous definition. They point out that any instance which only depends on the state of items in \( S \) is automatically self-certifying (Proposition 5.6 in Golovin and Krause [GK11]). That is the case here, since the objective function \( f = \min \{ Q, 1 - M_S \} \) only depends on the elements of \( S \). Therefore, our instance is self-certifying. ■

We are now ready to provide the guarantees for our bound, which we restate here:

**Theorem 3 (Performance Bound of HP and WHP).** Let our objective function be \( f \) as defined in eq. (3.1), utilizing either weighting function \( \omega^\text{HP} \) or \( \omega^\text{WHP} \). Define a threshold \( Q \) for the total weight of hypotheses we wish to remove. Let \( \eta \) be any value such that \( f(S) > Q - \eta \) implies \( f(S) \geq Q \) for all \( S \). Let \( \pi^*_{\text{avg}} \) and \( \pi^*_{\text{wc}} \) be the optimal
policies minimizing the expected and worst-case cost of tests selected, respectively. The greedy policy \( \pi^g \) satisfies:

\[
C(\pi^g) \leq C(\pi^*) \left( \ln \frac{Q}{\eta} + 1 \right)
\]

\[
C_{wc}(\pi^g) \leq C_{wc}(\pi^*) \left( \ln \frac{Q}{\delta \eta} + 1 \right)
\]

With \( \delta \) a constant based on the underlying non-noisy problem, described in appendix A.1.3.

As we have shown our objective is adaptive submodular, strongly adaptive monotone, and self-certifying, follows from Theorems 5.8 and 5.9 of Golovin and Krause [GK11]. Following their notation, we let \( \eta \) be any value such that \( f(S) > Q - \eta \) implies \( f(S) \geq Q \) for all \( S \). For Hypothesis Pruning, for example, we have \( \eta = \min_h P(h) \). Additionally, the bound on the worst case cost includes \( d = \min_{\hat{h}} \hat{h} (P(\hat{h}) \hat{h} \in \hat{H}) \). The specific values of these constants are related to the weighting function and how discretization is done. Nonetheless, for either weighting function and any way we discretize, we can guarantee the greedy algorithm selects a near-optimal sequence.

A.2 HyperEdge Cutting (HEC) Proofs

In this section, we provide proofs for the theorems stated in section 4.2.

A.2.1 \( k \) for Bounds

We start by showing that for a properly defined \( k \), the DRD problem is solved (\( \mathcal{V}(S) \subseteq r \)) if and only if the HEC objective is maximized. However, we sometimes require a slightly greater \( k \) to ensure the objective \( f_{\text{HEC}} \) is adaptive submodular. We define these below.

Let \( R \) be a set of regions, the length of which is related to \( k \). To get equivalence of the DRD and HEC, we require that for every region in \( R \), there is some hypothesis in all but one region of \( R \).

\[
R_{\text{iff}} = \arg \max_R |R| \quad \text{s.t.} \quad \forall r \in R, \exists h : h \notin r, h \in R \setminus r
\]

\[
k_{\text{iff}} = |R_{\text{iff}}|
\]

Sometimes, this is not sufficient for adaptive submodularity. For this, we also require that there is some hypothesis in every region of \( R \), and we also add one to the length of \( R \).

\[
R_{\text{as}} = \arg \max_R |R| \quad \text{s.t.} \quad 1) \exists \bar{h} \in R \quad 2) \forall r \in R, \exists h : h \notin r, h \in R \setminus r
\]

\[
k_{\text{as}} = |R_{\text{as}}| + 1
\]

Before moving on, we prove that \( k_{\text{as}} \geq k_{\text{iff}} \).

Proposition 2. \( k_{\text{as}} \geq k_{\text{iff}} \)

Proof. There are two cases:

1. \( \exists h \in R_{\text{iff}} \). In this case, \( R_{\text{as}} = R_{\text{iff}} \) and \( k_{\text{as}} = |R_{\text{as}}| + 1 = k_{\text{iff}} + 1 \).

2. \( \forall h \in R_{\text{iff}} \). Define \( \bar{R} = R_{\text{iff}} \setminus r \) for some \( r \in R_{\text{iff}} \). We know by definition of \( R_{\text{iff}} \) that \( \exists h \in \bar{R} \). Additionally, we know by definition of \( k_{\text{iff}} \) that \( \forall r \in \bar{R}, \exists \bar{h}, \bar{h} \notin r, h \in R_{\text{iff}} \setminus r \), so it follows that \( h \in \bar{R} \setminus r \). Therefore, we know \( \bar{R} \) satisfies the constraints for \( R_{\text{as}} \), and \( k_{\text{as}} \geq |\bar{R}| + 1 = |R_{\text{iff}}| = k_{\text{iff}} \).
Our algorithm actually utilizes $k = \min \left( \max_{h \in H} \left| \{ r : h \in r \} \right|, \max_{r \in R} \left| \{ g : g \in r \} \right| \right) + 1$. We briefly show that each of these also upper bound $k_{as}$.

**Proposition 3.** $\max_{h \in H} \left| \{ r : h \in r \} \right| + 1 \geq k_{as}$

**Proof.** Note that condition (1) in $R_{as}$ bounds $|R_{as}|$ by $\max_{h \in H} \left| \{ r : h \in r \} \right|$. The result follows.

**Proposition 4.** $\max_{r \in R} \left| \{ g : g \in r \} \right| + 1 \geq k_{as}$

**Proof.** Let $r$ be an element of $R_{as}$. By definition, it is required that at least $|R_{as}|$ different subregions $g_1 \cdots g_{|R_{as}|}$ be in that region - one which is in every other region in $R_{as}$ to satisfy condition (1), and $|R_{as}| - 1$ which are in all but one of the $R_{as} - 1$ other regions to satisfy condition (2). The result follows.

Thus, we can utilize $k = \min \left( \max_{h \in H} \left| \{ r : h \in r \} \right|, \max_{r \in R} \left| \{ g : g \in r \} \right| \right) + 1$ and apply the proofs using cardinality at least $k_{as}$ and $k_{eff}$. While our bounds and algorithm are better if we knew the correct $k_{as}$ to use, finding that value is itself hard to compute - thus, our implementation uses the value defined in section 4.2 and copied above.

### A.2.2 Theorem 4: Equivalence of DRD and HEC

**Theorem 4 (Relation of DRD and HEC).** Suppose we construct a splitting hypergraph by drawing hyperedges of cardinality $k$ according to eq. (4.3). Let $S \subseteq T \times O$ be a set of evidence. All consistent hypotheses lie in some decision region if and only if all hyperedges are cut, i.e.,

$$\mathcal{E}(S) = \emptyset \iff \exists r : \mathcal{V}(S) \subseteq r$$

**Proof.** We first prove that if all $h$ are contained in one region, then all edges are cut, i.e. $\exists r : \mathcal{V}(S) \subseteq r \Rightarrow \mathcal{E}(S) = \emptyset$. This is by construction, since a hyperedge $e \in \mathcal{E}(S)$ is only between subregions (or hypotheses) that do not share any regions. More concretely, our definition of $e$ requires $\beta r$ s.t. $\forall h \in e : h \in r$. Since all remaining nodes $\mathcal{V}(S) \subseteq r$, there will be no such such set of hypotheses.

Next, we prove that if all edges are removed, then all $h$ are contained in one region, i.e., $\mathcal{E}(S) = \emptyset \Rightarrow \exists r : \mathcal{V}(S) \subseteq r$. Clearly, if we set $|\mathcal{V}(S)| \leq k$, this condition would be met - $\mathcal{E}(S)$ would check every subset of $\mathcal{V}(S)$ to see if they shared a region, and would draw a hyperedge i.f.f. they do not. To complete the proof, we will make use of the following lemma:

**Lemma 4.** Define $\beta$ as some constant s.t. $\beta \geq k$. $\forall H \subseteq \mathcal{H}, |H| = \beta, \exists r : H \subseteq r \Rightarrow \forall \{H \cup h\} \subseteq \mathcal{H}, \exists r : \{H \cup h\} \in r$

**Proof.** For the sake of contradiction, suppose $\not\exists r : \{H \cup h\} \in r$. This must mean $h \notin H$. Let $\{H \cup h\} = \{h_1, h_2, \ldots, h_{\beta+1}\}$. Let $H_i$ be the subset of $\{H \cup h\}$ which does not include the $i$th $h$ from $\{H \cup h\}$, i.e. $H_i = \{h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_{\beta+1}\}$. By assumption, we know $\exists r : H_i \in r$. Let $r_i$ be that region for $H_i$. If $r_i = r_j$ for any $i, j$, this would imply $\{H_i \cup H_j\} = \{H \cup h\} \in r_i$. Thus, each $r_i$ must be unique if $\not\exists r : \{H \cup h\} \in r$. Furthermore, this implies $h_i \notin r_i$ and $h \in r_j, \forall j \neq i$. Let $R_{\beta+1} = \{r_1 \ldots r_{\beta+1}\}$. By definition of $\beta$, we know $\beta \geq k \geq k_{eff}$. But this causes a contradiction - by definition of $k_{eff}$, the maximum set of regions $R$ where $h_i \notin r_i, h_i \in r_j, \forall j \neq i$ is $k_{eff}$. But $R_{\beta+1}$ would require such a set of regions where $|R_{\beta+1}| = \beta + 1 \geq k_{eff} + 1$. Thus, we have a contradiction, and have shown $\exists r : \{H \cup h\} \in r$. 

By construction, we know that if $\mathcal{E}(\mathcal{S}) = \emptyset \Rightarrow \forall H \subseteq \mathcal{H}, |H| \leq k, \exists r : H \subseteq r$. Applying lemma 4 inductively, this implies, $\forall \{H \cup h_1\} \subseteq \mathcal{V}(\mathcal{S}), \exists r : \{H \cup h_1 \cup h_2\} \subseteq \mathcal{V}(\mathcal{S}), \exists r : \{H \cup h_1 \cup h_2\} \subseteq r \Rightarrow \cdots \Rightarrow \exists r : \mathcal{V}(\mathcal{S}) \subseteq r$.

A.2.3 Theorem 5: Strong Adaptive Monotonicity and Adaptive Submodularity

Theorem 5 (Adaptive Submodularity of HEC). The objective function $f_{\text{HEC}}$ defined in eq. (4.6) is adaptive submodular and strongly adaptive monotone.

Proof. We start with showing our formulation is strongly adaptive monotone.

Lemma 5. The function $f_{\text{HEC}}$ described above is strongly adaptive monotone, i.e.

$$f_{\text{HEC}}(\mathcal{S} \cup \{(t, h(t))\}) - f_{\text{HEC}}(\mathcal{S}) \geq 0 \quad \forall t, h$$

Proof. This states that our utility function must always increase as we take additional actions and receive observations. Intuitively, we can see that additional action observation pairs can only cut edges, and thus our utility function always increases. More concretely:

$$f_{\text{HEC}}(\mathcal{S} \cup \{(t, h(t))\}) - f_{\text{HEC}}(\mathcal{S})$$

$$= (w(\mathcal{E}) - w(\mathcal{E} \cup \{(t, h(t))\})) - (w(\mathcal{E}) - w(\mathcal{E}(\mathcal{S})))$$

$$= w(\mathcal{E}(\mathcal{S})) - w(\mathcal{E} \cup \{(t, h(t))\})$$

$$= w(\{e \in \mathcal{E} : \forall (i, o) \in \mathcal{S} \exists h \in e, \exists h(i) = o\})$$

$$- w(\{e \in \mathcal{E} : \forall (i, o) \in \mathcal{S} \exists h \in e, \exists h(i) = o, h(t) = h(t)\})$$

$$= w(\{e \in \mathcal{E} : \forall (i, o) \in \mathcal{S} \exists h \in e, \exists h(i) = o, h(t) \neq h(t)\})$$

$$\geq 0$$

since $w(e) \geq 0 \forall e$

Lemma 6. The function $f_{\text{HEC}}$ described above is adaptive submodular for any prior with rational values, i.e. for $\mathcal{S} \subseteq \hat{\mathcal{S}} \subseteq \mathcal{T} \times \mathcal{O}$

$$\Delta_{f_{\text{HEC}}}(t | \mathcal{S}) \geq \Delta_{f_{\text{HEC}}}(t | \hat{\mathcal{S}}) \quad \forall t \in \mathcal{T} \setminus \mathcal{S}$$

where $\mathcal{S} \mathcal{T}$ are the set of tests in $\mathcal{S}$.

Proof. This states that our expected utility for a fixed action $t$ decreases as we take additional actions and receive observations. We rewrite our expected marginal utility in a more convenient form:

$$\Delta_{f_{\text{HEC}}}(t | \mathcal{S}) = \sum_h P(h | \mathcal{S}) \left( f_{\text{HEC}}(\mathcal{S} \cup \{(t, h(t))\}) - f_{\text{HEC}}(\mathcal{S}) \right)$$

$$= \sum_h P(h | \mathcal{S}) \left( [w(\mathcal{E}) - w(\mathcal{E} \cup \{(t, h(t))\})] - [w(\mathcal{E}) - w(\mathcal{E}(\mathcal{S}))] \right)$$

$$= \sum_h P(h | \mathcal{S}) \left( w(\mathcal{E}(\mathcal{S})) - w(\mathcal{E} \cup \{(t, h(t))\}) \right)$$
For convenience, we define \( n_i \) to be the total probability mass in \( \mathcal{G}_i \) consistent with all evidence in \( \mathcal{S} \) and observation \( o \). We define \( n^o \) similarly. More formally:

\[
\begin{align*}
\hat{n}_i^o &= \sum_{h \in \mathcal{G}_i} P(h) \mathbb{I}(h \in \mathcal{V}(\mathcal{S} \cup \{(t, o)\})) \\
n_i &= \sum_{o \in \Omega} n_i^o \\
n^o &= \sum_{\mathcal{G}_i \in \mathcal{U}} n_i^o \\
N &= \sum_{\mathcal{G}_i \in \mathcal{U}} \sum_{o \in \Omega} n_i^o \\
\hat{w}(\mathcal{E}(\mathcal{S})) &= \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i
\end{align*}
\]

Similarly, we can also write \( \hat{w}(\mathcal{E}(\mathcal{S} \cup \{(t, o)\})) = \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^o \). We can rewrite our objective as:

\[
\begin{align*}
\Delta_{f_{\text{hec}}} (t \mid \mathcal{S}) &= \sum_{h} P(h \mid \mathcal{S}) \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i - \sum_{e \in \mathcal{E}} \prod_{i \in e} \hat{n}_i^o (t) \right) \\
&= \sum_{o} \frac{n^o}{N} \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i - \sum_{e \in \mathcal{E}} \prod_{i \in e} \hat{n}_i^o \right) \\
&= \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i - \sum_{o} \frac{n^o}{N} \sum_{e \in \mathcal{E}} \prod_{i \in e} \hat{n}_i^o
\end{align*}
\]

Similarly, we define variables for the evidence \( \hat{S} \), i.e. \( \hat{n}_i^o \) for the total probability mass in \( \mathcal{G}_i \) consistent with all evidence in \( \hat{S} \) and observation \( o \):

\[
\Delta_{f_{\text{hec}}} (t \mid \hat{S}) = \sum_{e \in \mathcal{E}} \prod_{i \in e} \hat{n}_i - \sum_{o} \frac{\hat{n}_i^o}{\hat{N}} \sum_{e \in \mathcal{E}} \prod_{i \in e} \hat{n}_i^o
\]

We rewrite what we would like to show as:

\[
\begin{align*}
\Delta_{f_{\text{hec}}} (t \mid \mathcal{S}) - \Delta_{f_{\text{hec}}} (t \mid \hat{S}) &= \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i - \sum_{o} \frac{n^o}{N} \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^o \right) - \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} \hat{n}_i - \sum_{o} \frac{\hat{n}_i^o}{\hat{N}} \sum_{e \in \mathcal{E}} \prod_{i \in e} \hat{n}_i^o \right) \\
&\geq 0
\end{align*}
\]

We will show that for any single action observation pair, which corresponds to eliminating a single hypothesis, the expected utility of a test will always decrease. General adaptive submodularity, which states the expected utility decreases with any additional evidence, follows easily. For convenience, we consider rescaling our function so that all \( n_i^o \) are integers, which is possible since we assumed a rational prior. Note that a function \( f \) is adaptive submodular i.f.f. \( cf \) is adaptive submodular for any constant \( c > 0 \), so showing adaptive submodularity in the rescaled setting implies adaptive submodularity for our setting.

**Lemma 6.1.** If we remove one hypothesis from subregion \( k \) which agrees with observation \( c \), i.e.

\[
\hat{n}_i^o = \begin{cases} 
    n_i^o - 1 & \text{if } i = 1 \text{ and } o = c \\
    n_i^o & \text{else}
\end{cases}
\]
then

\[
\Delta = \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i - \sum_{o} \frac{n^o}{N} \sum_{e \in \mathcal{E}} \prod_{i \in e} n^o_i \right) - \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} \tilde{n}_i - \sum_{o} \frac{\tilde{n}^o}{N} \sum_{e \in \mathcal{E}} \prod_{i \in e} \tilde{n}^o_i \right) \geq 0
\]

Proof. Based on our definitions, it follows that:

\[
\hat{n}_i = \begin{cases} n_i - 1 & \text{if } i = l \\ n_i & \text{else} \end{cases}
\]

\[
\hat{n}^o = \begin{cases} n^o - 1 & \text{if } o = c \\ n^o & \text{else} \end{cases}
\]

\[
\hat{N} = N - 1
\]

We split the difference into three terms:

\[
\Delta^a = \sum_{e \in \mathcal{E}} \left( \prod_{i \in e} n_i - \prod_{i \in e} \hat{n}_i \right)
\]

\[
\Delta^b = \sum_{o \in \mathcal{C} \setminus \{} \sum_{e \in \mathcal{E}^o} \left( -\frac{n^o}{N} \prod_{i \in e} n^o_i + \frac{\hat{n}^o}{N} \prod_{i \in e} \hat{n}^o_i \right)
\]

\[
\Delta^c = \sum_{e \in \mathcal{E}} \left( \frac{n^c}{N} \prod_{i \in e} n^c_i + \frac{\hat{n}^c}{N} \prod_{i \in e} \hat{n}^c_i \right)
\]

\[
\Delta^a + \Delta^b + \Delta^c = \Delta
\]

To aid in notation, we define \( \mathcal{E}_l = \{ e \in \mathcal{E} : g_l \in e \} \), hyperedges that contain region \( l \), and \( \mathcal{E}_l = \mathcal{E} \setminus \mathcal{E}_l \), all other hyperedges. Additionally, let \( |e_l| \) be the number of times \( g_l \) appears in the multiset \( e \).

First term:

\[
\Delta^a = \sum_{e \in \mathcal{E}} \left( \prod_{i \in e} n_i - \prod_{i \in e} \hat{n}_i \right)
\]

\[
= \sum_{e \in \mathcal{E}_l} \left[ \prod_{i \in e} n_i - \prod_{i \in e} \hat{n}_i \right] + \sum_{e \in \mathcal{E}_l^o} \left[ \prod_{i \in e} n_i - \prod_{i \in e} \hat{n}_i \right]
\]

\[
= \sum_{e \in \mathcal{E}_l} \left[ \prod_{i \in e} n_i - \prod_{i \in e} n_i \right] + \sum_{e \in \mathcal{E}_l^o} \left[ \left( \prod_{i \in e, i \neq l} n_i \right) n^{|e_l|} - \left( \prod_{i \in e, i \neq l} n_i \right) \left( n_l - 1 \right)^{|e_l|} \right]
\]

\[
= \sum_{e \in \mathcal{E}_l} \left[ \prod_{i \in e, i \neq l} n_i \right] \left( n_l^{|e_l|} - \left( n_l - 1 \right)^{|e_l|} \right)
\]

\[
\geq 0 \quad \text{(since } n_l \geq 1)\]
Second term:

\[
\Delta^b = \sum_{o \in O \setminus \epsilon} \sum_{e \in \mathcal{E}} \left( -\frac{n^o}{N} \prod_{i \in e} n_i^o + \frac{\hat{n}^o}{N} \prod_{i \in e} n_i^o \right) \\
= \sum_{o \in O \setminus \epsilon} \sum_{e \in \mathcal{E}} \left( -\frac{n^o}{N} \prod_{i \in e} n_i^o + \frac{n^o}{N - 1} \prod_{i \in e} n_i^o \right) \\
= \sum_{o \in O \setminus \epsilon} \sum_{e \in \mathcal{E}} \frac{n^o}{N(N - 1)} \prod_{i \in e} n_i^o \\
\geq 0 \quad \text{(since each term } \geq 0) 
\]

Third term:

\[
\Delta^c = \sum_{e \in \mathcal{E}} \left( -\frac{n^c}{N} \prod_{i \in e} n_i^c + \frac{\hat{n}^c}{N} \prod_{i \in e} n_i^c \right) \\
= -\frac{n^c}{N} \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^c + \frac{n^c - 1}{N - 1} \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^c \right) (n_i^c - 1)^{|e|} \\
= -\frac{n^c}{N} \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^c + \frac{n^c - 1}{N - 1} \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^c \right) \left( (n_i^c)^{|e|} - (n_i^c - 1)^{|e|} \right) \\
= -\frac{n^c}{N} \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^c + \frac{n^c - 1}{N - 1} \left( \sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^c \right) \left( (n_i^c)^{|e|} - (n_i^c - 1)^{|e|} \right) \\
\leq 0 \quad \text{(since each term } \leq 0) 
\]

We also define:

\[
\Delta^c = \left( \frac{N - n^c}{N(N - 1)} \right) \Delta_1^c + \left( \frac{n^c - 1}{N - 1} \right) \Delta_2^c \\
\Delta_1^c = -\sum_{e \in \mathcal{E}} \prod_{i \in e} n_i^c \\
\Delta_2^c = -\left( \sum_{e \in \mathcal{E}} \prod_{i \in e, i \neq l} n_i^c \right) \left( (n_i^c)^{|e|} - (n_i^c - 1)^{|e|} \right) 
\]

\[
\Delta^a = \left( \frac{N(N - n^c)}{N(N - 1)} + \frac{n^c - 1}{N - 1} \right) \Delta^a \\
= \left( \frac{N - n^c}{N(N - 1)} \right) \Delta_1^a + \left( \frac{n^c - 1}{N - 1} \right) \Delta_2^a \\
\Delta_1^a = N \sum_{e \in \mathcal{E}} \left( \prod_{i \in e, i \neq l} n_i \right) \left( n_i^{|e|} - (n_i - 1)^{|e|} \right) \\
\Delta_2^a = \sum_{e \in \mathcal{E}} \left( \prod_{i \in e, i \neq l} n_i \right) \left( n_i^{|e|} - (n_i - 1)^{|e|} \right) 
\]
The constants in front of the sum for $\Delta^a_1$ and $\Delta^a_2$ were from the equation, and $\Delta^a$ was split up to include the same constants. Now we will show that $\Delta^a_1 + \Delta^a_1 \geq 0$ and $\Delta^a_2 + \Delta^a_2 \geq 0$. We start with the latter:

$$\Delta^a_2 + \Delta^a_2 = \sum_{e \in \mathcal{E}_i} \left[ \prod_{j \in e, j \neq i} n_i \right] \left( n_i^{|e|} - (n_i - 1)^{|e|} \right) - \prod_{j \in e} n_i \left( n_i^{|e|} - (n_i - 1)^{|e|} \right)$$

$$\geq \sum_{e \in \mathcal{E}_i} \left[ \prod_{j \in e, j \neq i} n_i \right] \left( n_i^{|e|} - (n_i - 1)^{|e|} \right) - \prod_{j \in e} n_i \left( n_i^{|e|} - (n_i - 1)^{|e|} \right)$$

$$= 0$$

Where (A.6) follows from $n_i \geq n_i^c \forall i$.

$$\Delta^a_1 + \Delta^a_1 = N \sum_{e \in \mathcal{E}_i} \left[ \prod_{j \in e, j \neq i} n_i \right] \left( n_i^{|e|} - (n_i - 1)^{|e|} \right) - \sum_{e \in \mathcal{E}_i} n_i$$

$$\geq N \sum_{e \in \mathcal{E}_i} \left[ \prod_{j \in e, j \neq i} n_i \right] n_i^{|e| - 1} - \sum_{e \in \mathcal{E}_i} n_i$$

$$= (N - n_i) \sum_{e \in \mathcal{E}_i} \left[ \prod_{j \in e, j \neq i} n_i \right] n_i^{|e| - 1} + \sum_{e \in \mathcal{E}_i} \left[ \prod_{j \in e, j \neq i} n_i \right] n_i^{|e|} - \sum_{e \in \mathcal{E}_i} n_i$$

$$= (N - n_i) \sum_{e \in \mathcal{E}_i \setminus \mathcal{E}_j} \left[ \prod_{j \in e, j \neq i} n_i \right] n_i^{|e| - 1} - \sum_{e \in \mathcal{E}_i \setminus \mathcal{E}_j} n_i$$

$$\geq (N - n_i) \sum_{e \in \mathcal{E}_i \setminus \mathcal{E}_j} \left[ \prod_{j \in e, j \neq i} n_i \right] n_i - \sum_{e \in \mathcal{E}_i \setminus \mathcal{E}_j} n_i$$

Where (A.7) follows from $n_i \geq n_i^c \forall i$, (A.8) follows from $n_i^{|e|} - (n_i - 1)^{|e|} \geq n_i^{|e|} - n_i^{|e| - 1} (n_i - 1) = n_i^{|e| - 1}$, and (A.9) cancels edges in $\mathcal{E}_i$ exactly, leaving only edges in $\mathcal{E}_j$.

We again want to separate out terms that cancel. We define:

$$\mathcal{E}_j^k = \{ e : |e| = \widehat{k} \land \exists j \text{ s.t. } \forall g \in e : g \subseteq r_j \}$$

$$\mathcal{E}_j^{\min} = \{ e : e \in \mathcal{E}, \widehat{\exists} e \subset e : \widehat{\exists} \in \mathcal{E}_j^{k - 1} \}$$

$$\overline{\mathcal{E}_j^{\min}} = \mathcal{E} \setminus \mathcal{E}_j^{\min}$$

We defined $\mathcal{E}_j^k$ as the hyperedges for any specified cardinality $\widehat{k}$. We call $\mathcal{E}_j^{\min}$ the minimal hyperedges if $k$ is the minimal cardinality at which these regions should be seperated. Thus, these are the hyperedges where no subset of subregions $\{s_1 \cdots s_{k-1}\} \subset e$ would have a seperation hyperedge. All other hyperedges are called non-minimal. We also define $\mathcal{E}_j^{\min}, \mathcal{E}_j^{\min}, \mathcal{E}_j^{\min}, \mathcal{E}_j^{\min}$ as the minimal and non-minimal hyperedges
of \( \mathcal{E}_l \) and \( \overline{\mathcal{E}_l} \):

\[
\begin{align*}
\mathcal{E}_l^{\text{min}} &= \{ e : e \in \mathcal{E}_l, \mathcal{E} \subset e : e \notin \mathcal{E}^{k-1} \}, \\
\overline{\mathcal{E}_l}^{\text{min}} &= \{ e : e \in \overline{\mathcal{E}_l}, \mathcal{E} \subset e : e \notin \mathcal{E}^{k-1} \}, \\
\mathcal{E}_l^{\text{min}} &= \mathcal{E}_l \setminus \mathcal{E}_l^{\text{min}} \\
\overline{\mathcal{E}_l}^{\text{min}} &= \overline{\mathcal{E}_l} \setminus \overline{\mathcal{E}_l}^{\text{min}}
\end{align*}
\]

We also note that:

\[
\sum_{e \in \mathcal{E}_l^{\text{min}}} \prod_{i \notin e} n_i \leq \sum_{g \in \mathcal{G} \setminus \mathcal{G}_l} n_j \sum_{e \in \overline{\mathcal{E}_l}^{k-1}} \prod_{i \notin e} n_i
\]

\[
= (N - n_l) \sum_{e \in \mathcal{E}_l^{k-1}} \prod_{i \notin e} n_i
\]

For convenience, we define one additional set of hyperedges \( \mathcal{E}_{l}^{\text{c}} \). These are hyperedges in \( \mathcal{E}_l \) such that no subset of \( k - 1 \) elements which do not include \( k \) are in \( \overline{\mathcal{E}_l} \).

\[
\mathcal{E}_{l}^{\text{c}} = \{ e : e \in \mathcal{E}_l \land \mathcal{E}_{k-1} \subset e \text{ s.t. } e_{k-1} \in \mathcal{E}_l^{k-1} \}
\]

This enables us to split the set \( \mathcal{E}_l \) up into edges where \( \mathcal{E}_l^{k-1} \) are a subset, and \( \mathcal{E}_{l}^{\text{c}} \). We note that since there is no region shared by all elements \( e^{k-1} \in \mathcal{E}_l^{k-1} \), then there will be no region shared by \( e = e^{k-1} \cup g_l \). Thus, this will be an element of \( \mathcal{E}_l^{\text{c}} \). This gives us:

\[
\sum_{e \in \mathcal{E}_l^{\text{c}}} \prod_{i \notin e} n_i = \sum_{e \in \mathcal{E}_l^{k-1}} \prod_{i \notin e} n_i + \sum_{e \in \mathcal{E}_{l}^{\text{c}}} \prod_{i \notin e} n_i
\]

Applying these:

\[
\Delta_l^0 + \Delta_l^1 \geq (N - n_l) \sum_{e \in \mathcal{E}_l^{\text{c}}} \prod_{i \notin e} n_i - \sum_{e \in \overline{\mathcal{E}_l}^{\text{min}}} \prod_{i \notin e} n_i
\]

\[
= (N - n_l) \left( \sum_{e \in \mathcal{E}_l^{k-1}} \prod_{i \in e} n_i + \sum_{e \in \mathcal{E}_l^{\text{c}}} \prod_{i \notin e} n_i \right) - \sum_{e \in \mathcal{E}_l^{\text{min}}} \prod_{i \notin e} n_i - \sum_{e \in \mathcal{E}_l^{\text{min}}} \prod_{i \notin e} n_i
\]

\[
\geq (N - n_l) \left( \sum_{e \in \mathcal{E}_l^{k-1}} \prod_{i \in e} n_i + \sum_{e \in \mathcal{E}_l^{\text{c}}} \prod_{i \notin e} n_i \right) - (N - n_l) \sum_{e \in \mathcal{E}_l^{k-1}} \prod_{i \in e} n_i - \sum_{e \in \mathcal{E}_l^{\text{min}}} \prod_{i \notin e} n_i
\]

\[
= (N - n_l) \left( \sum_{e \in \mathcal{E}_l^{\text{c}}} \prod_{i \notin e} n_i \right) - \sum_{e \in \mathcal{E}_l^{\text{min}}} \prod_{i \notin e} n_i
\]

At this point, we use the structure of our edge construction and definition of \( k \) to show this sum is \( \geq 0 \).

We have a positive term, consisting of edges which include \( k \), and a negative term, consisting of edges that do not include \( k \). We will show that for every product in the negative term, there is a corresponding product in the positive term.

To do so, we show that for any \( e \in \mathcal{E}_l^{\text{min}} \), there is at least one corresponding \( e' \in \mathcal{E}_{l}^{\text{c}} \) to cancel the terms out. More concretely:
Lemma 6.1.1. Let $e \in \mathcal{E}^\text{min}_I$. There exists some $e^{k-1} \subseteq e, |e^{k-1}| = k - 1$ such that $e' = (e^{k-1} \cup g_I) \subseteq \hat{E}_I$.

Proof. Recall that $e$ is a multiset of subregions. It is straightforward to see that because $e$ is minimal, there can be no repeated elements in the multiset - and thus it is equivalent to a set. Define this set as $e = \{\hat{g}_1, \ldots, \hat{g}_k\}$. Define each distinct subset which does not include $\hat{g}_i$ as $e_i = e\setminus \hat{g}_i, 1 \leq i \leq k$. By our definition of minimal hyperedges $e^{\text{min}}_I$, we know that $\forall e_i, \exists r_i : e_i \subseteq r_i$, which implies that $e_i \not\subseteq \mathcal{E}^{k-1}_I$. Note that each $r_i$ must be distinct. If $r_i = r_j$, for any $i, j$, this would imply $(e_i \cup e_j) = e \in r_i$. But since there exists a separating hyperedge $e$,$/\exists r : e \subseteq r$. This implies $\hat{g}_i \not\subseteq r_i$. Combining this with our definition of $\hat{E}_I$, if $\hat{E}_I$ is a subset of $\mathcal{E}^{k-1}_I$, then it cannot be true: the largest such $\hat{E}_I$ cannot exist for all $e_i$. If $g_I \not\subseteq r_i \Rightarrow e_i \cup g_I \not\subseteq r_i$. For the sake of contradiction, suppose $g_i \subseteq r_i \forall i$. Let $R = \{r_1 \ldots r_k\}$. For this to be true, it must be that: (1) $\forall h \in g_i, h \in R (2) \forall r_i \in R, \exists h \in \hat{g}_i : h \not\in r_i$, $h \in R \setminus r_i$ where $|R| = k$. However, by definition of $k$ this cannot be true: the largest such $R$ where this holds $|R| = k - 1$. Thus, we have a contradiction, and have shown such a set of regions $\{r_1 \ldots r_k\} = R : g_I \subseteq r_i \forall r_i$ cannot exist. Therefore, $\exists e_i : (e_i \cup g_I) \subseteq \hat{E}_I$.

In order to apply Lemma 6.1.1, we split every $e \in \mathcal{E}^{\text{min}}_I$ it up into $e^{k-1}$ and $\overline{e}$, where $e^{k-1}$ is the subset of $e$ such that $(e^{k-1} \cup g_I) \subseteq \hat{E}_I$, and $\overline{e} = e \setminus e^{k-1}$. Let $\overline{n}$ be the number of particles in subregion $\overline{e}$, which we will use in eq. (A.10):

$$\Delta^a_i + \Delta^b_i \geq (N - n_i) \left( \sum_{e \in \hat{E}_I} \prod_{i \notin e} n_i \right) - \sum_{e \in \mathcal{E}^{\text{min}}_I} \prod_{i \notin e} n_i$$

$$= (N - n_i) \left( \sum_{e \in \hat{E}_I} \prod_{i \notin e} n_i \right) - \sum_{e \in \mathcal{E}^{\text{min}}_I} \prod_{i \notin e^{k-1}} n_i$$

$$\geq (N - n_i) \left( \sum_{e \in \hat{E}_I} \prod_{i \notin e} n_i \right) - \sum_{g_i \in \overline{e} \setminus g_I} n_j \left( \sum_{e \in \hat{E}_I} \prod_{i \notin e} n_i \right)$$

$$= (N - n_i) \left( \sum_{e \in \hat{E}_I} \prod_{i \notin e} n_i \right) - (N - n_i) \left( \sum_{e \in \hat{E}_I} \prod_{i \notin e} n_i \right)$$

$$= 0$$


At this point, we have shown that $\Delta = \Delta^a + \Delta^b + \Delta^c \geq 0$, since $\Delta^b \geq 0$ and $\Delta^a + \Delta^c \geq 0$, which is what we needed to show.

It is not hard to see that for any $S \subseteq \hat{S} \subseteq T \times O$, we could show that $\Delta_{\text{fHIC}}(t \mid S) \geq \Delta_{\text{fHIC}}(t \mid \hat{S}_1) \geq \Delta_{\text{fHIC}}(t \mid \hat{S}_2) \cdots \geq \Delta_{\text{fHIC}}(t \mid \hat{S})$. In other words, we can always find a sequence of removing one hypothesis at a time to get from $S$ to $\hat{S}$ when $S \subseteq \hat{S} \subseteq T \times O$. 

A.2.4 **Theorem 6: Greedy Performance Bound**

**Theorem 6 (HEC Performance Bound).** Assume that the prior probability distribution $P$ on the set of hypotheses is rational. Then, the performance of $\pi_{\text{HEC}}$ is bounded as follows:

$$C(\pi_{\text{HEC}}) \leq (k \ln(1/p_{\text{min}}) + 1)C(\pi^*),$$

where $p_{\text{min}} = \min_{h \in H} P(h)$ and $\pi^*$ is the optimal policy.

We would like to apply Theorem 5.8 of $[\text{GK}11]$. We have already shown adaptive submodularity and strong adaptive monotonicity in appendix A.2.3. The theorem also requires that instances are self-certifying, which means that when the policy knows it has obtained the maximum possible objective value immediately upon doing so. See $[\text{GK}11]$ for details. As our objective is equivalent for all remaining hypotheses in $V(S)$, our function $f_{\text{HEC}}$ is self-certifying.

The performance bound now follows directly from Theorem 5.8 of $[\text{GK}11]$. To apply the theorem, we needed to define two constants: a bound on the maximum value of $f_{\text{HEC}}(S)$, $Q = 1$, and the minimum our objective function can change by, which corresponds to removing one hyperedge, $\eta = p_{\text{min}}^k$. Plugging those into Theorem 5.8 of $[\text{GK}11]$ gives $C(\pi_{\text{HEC}}) \leq (k \ln(1/p_{\text{min}}) + 1)C(\pi^*)$.

A.3 **Multi-Target MDPs**

Below we provide the proofs for decomposing the value functions for MDPs with multiple targets, as introduced in section 6.4.

A.3.1 **Theorem 10: Decomposing value functions**

Here, we show the proof for our theorem that we can decompose the value functions over that the targets for deterministic MDPs:

**Theorem 10.** Let $V_k$ be the value function for target $k$. Define the cost for the goal as in eq. (6.3). For an MDP with deterministic transitions, and a deterministic user policy $\pi^u$, the value and action-value functions $V_g$ and $Q_g$ can be computed as:

$$Q_g(x, u, a) = Q_{k^*}(x, u, a) \quad \kappa^* = \arg \min_k V_k(x')$$

$$V_g(x) = \min_{x'} V_k(x')$$

**Proof.** We show how the standard value iteration algorithm, computing $Q_g$ and $V_g$ backwards, breaks down at each time step. At the final timestep $T$, we get:

$$Q^T(x, a) = C_\kappa(x, a)$$

$$V^T(x) = \min_{a} C_\kappa(x, a)$$

for any $\kappa$
Since $V^T_k(x) = \min_a C_{x,a}(x,a)$ by definition. Now, we show the recursive step:

\[
Q^{t-1}(x,a) = C_{x,a}(x,a) + V^t(x') \\
= C(x,a) + \min_\kappa V_k^t(x') \quad \kappa^* = \arg\min V_k(x') \\
= C(x,a) + V_k^t(x') \quad \kappa^* = \arg\min V_k(x') \\
V^{t-1}(x) = \min_a Q^{t-1}(x,a) \\
= \min_a C(x,a) + V_k^t(x') \quad \kappa^* = \arg\min V_k(x') \\
\geq \min_a (C(x,a) + V_k^t(x')) \\
= \min_k V_k^{t-1}(x)
\]

Additionally, we know that $V(x) \leq \min_k V_k(x)$, since $V_k(x)$ measures the cost-to-go for a specific target, and the total cost-to-go is bounded by this value for a deterministic system. Therefore, $V(x) = \min_k V_k(x)$.

\[\square\]

A.3.2 Theorem 11: Decomposing soft value functions

Here, we show the proof for our theorem that we can decompose the soft value functions over that the targets for deterministic MDPs:

**Theorem 11.** Define the probability of a trajectory and target as $p(\xi, k) \propto \exp(-C_k(\xi))$. Let $V^\text{soft}_k$ and $Q^\text{soft}_k$ be the soft-value functions for target $\kappa$. For an MDP with deterministic transitions, the soft value functions for goal $g$, $V^\text{soft}_g$, and $Q^\text{soft}_g$, can be computed as:

\[
V^\text{soft}_g(x) = \arg\min_\kappa V^\text{soft}_k(x) \\
Q^\text{soft}_g(x, u) = \arg\min_\kappa Q^\text{soft}_k(x, u)
\]

**Proof.** As the cost is additive along the trajectory, we can expand out $\exp(-C_k(\xi))$ and marginalize over future inputs to get the probability of an input now:

\[
\pi^g(u_t, k|x_t) = \frac{\exp(-C_k(x_t, u_t)) \int \exp(-C_k(\xi_{t+1}^{T}))}{\sum_{k'} \int \exp(-C_k(\xi_{t+1}^{T}))}
\]

Where the integrals are over all trajectories. By definition, $\exp(-V^\text{soft}_g(x_t)) = \int \exp(-C_k(\xi_{t+1}^{T}))$:

\[
\begin{align*}
\pi^g(u_t, k|x_t) &= \frac{\exp(-C_k(x_t, u_t)) \exp(-V^\text{soft}_g(x_{t+1}))}{\sum_{k'} \exp(-V^\text{soft}_g(x_{t+1}))} \\
&= \frac{\exp(-Q^\text{soft}_g(x_t, u_t))}{\sum_{k'} \exp(-V^\text{soft}_g(x_t))}
\end{align*}
\]
Marginalizing out $\kappa$ and simplifying:

$$
\pi^u(u_t|x_t) = \frac{\sum_k \exp(-Q^{\text{soft}}_{\kappa,t}(x_t, u_t))}{\sum_k \exp(-V^{\text{soft}}_{\kappa,t}(x_t))}
$$

$$
= \exp \left( \log \left( \frac{\sum_k \exp(-Q^{\text{soft}}_{\kappa,t}(x_t, u_t))}{\sum_k \exp(-V^{\text{soft}}_{\kappa,t}(x_t))} \right) \right)
$$

$$
= \exp \left( \text{softmin}_k V^{\text{soft}}_{\kappa,t}(x_t) - \text{softmin}_k Q^{\text{soft}}_{\kappa,t}(x_t, u_t) \right)
$$

As $V^{\text{soft}}_{g,t}$ and $Q^{\text{soft}}_{g,t}$ are defined such that $\pi^u(u|x,g) = \exp(V^{\text{soft}}_{g,t}(x) - Q^{\text{soft}}_{g,t}(x,u))$, our proof is complete. \hfill \square
List of Figures

1.1 Example Manipulation Task Information Acting Under Uncertainty 9
1.2 Example Shared Autonomy Task for Acting Under Uncertainty 9
1.3 Goal-Directed Active Information Gathering 10
1.4 Assistance with Ambiguous Goals in Shared Control Teleoperation 12
1.5 Assistance with Ambiguous Goals in Human-Robot Team- ing 12

3.1 Hypothesis Pruning for Door Opening 28
3.2 Touch-Based Localization as Set Cover 29
3.3 Observation Model for Touch-Based Localization 31
3.4 Hypothesis Pruning Covariance Evolution 35

4.1 The Decision Region Determination (DRD) problem 41
4.2 A decision region for button pushing 42
4.3 HEC with two decision regions 46
4.4 A depiction of our method as hyperedges. (a) The equivalent hyperedges of $CHP_3(\mathcal{G})$. (b) First iteration of algorithm 1 which removes all $|\xi| = 1$ (light edges) by subtracting $g_1CHP_2(\{g_1\}) + g_2CHP_2(\{g_2\}) + g_3CHP_2(\{g_3\})$. (c) Second iteration of algorithm 1 which removes all $|\xi| = 2$ (light edges) by subtracting $g_1g_2CHP_1(\{g_1,g_2\}) + g_2g_3CHP_1(\{g_2,g_3\})$ 49
4.5 HEC Results on MovieLens 100k Experiments 51
4.6 Illustration of Decision Regions for MovieLens 100k for HEC Experiments 51
4.7 Decision Regions for Microwave Touch Based Localization 52
4.8 HEC Robot Experiments Barchart 53
4.9 Noisy-Or Problem Instance 54
4.10 Noisy-Or Graph Coloring Decomposition 55
4.11 Noisy-Or Touch Based Localization Experimental Setup 58
4.12 Noisy-Or Robot Simulation Results 58
4.13 DiRECt Experimental results for MovieLens, EMPCranes, and Risky Choice Theory 59

5.1 Teleoperation Interfaces 63
5.2 Modal Control 65
5.3 Shared Autonomy Blend Diagram 66
5.4 Shared Autonomy Policy Diagram 66

6.1 Shared Autonomy Framework Diagram 71
6.2 Arbitration as a function of confidence in Policy Method 72
6.3 Multi-Target Value Functions 81

7.1 Example of Hindsight Optimization and Prediction with User Inputs 84
7.2 Grasping Study Experimental Setup 86
7.3 Grasping Study Survey Response 88
7.4 Grasping Study User Preference vs. Like Ratings 88
7.5 Grasping Study Time and Control Plots 89
7.6 Grasp Study Trajectories for Two Users 90
7.7 Eating Study Experiment Setup 92
7.8 Eating Study Results 94
7.9 Eating Study Time vs. User Inputs and Assistance Ratio 96
7.10 Eating Study Survey Responses 97
7.11 Gift Wrapping Study Experiment Setup 102
7.12 Gift Wrapping Study Distance Metrics Boxplots 104
7.13 Gift Wrapping Study Idle Time and Percentage Boxplots 104
7.14 Gift Wrapping Study Trial Duration Boxplots 105

8.1 Prediction with Assistance Method Overview 109
8.2 Prediction with Assistance Experimental Setup 113
8.3 Prediction with Assistance Experiment Objective Measures 115
8.4 Prediction with Assistance Survey Responses 116
## List of Tables

2.1 Variable Definitions for Adaptive Submodularity 21

3.1 Hypothesis Pruning Test Selection Time 36
3.2 Example Tests for Hypothesis Pruning Experiment 37
3.3 Hypothesis Pruning Real Robot Experiment 38

4.1 HEC Experiments Running Times on MovieLens 100k 52
4.2 DRD Problem Applications, and the Corresponding Tests and Decisions 57

6.1 Shared Autonomy variable definitions 73

7.1 Eating Study Post-Hoc P-Values for Every Hypothesis 99
7.2 Gift Wrapping Study Subjective Ratings Depending on Collision 106
Bibliography


Andrew Guillory and Jeff Bilmes. “Average-Case Active Learning with Costs”. In: The International Conference on Algorithmic Learning Theory. 2009.


Yanan Li, Keng Peng Tee, Wei Liang Chan, Rui Yan, Yuanwei Chua, and Dilip Kumar Limbu. “Role adaptation of human and robot in collaborative tasks”. In: *icra*. 2015.

Jue Kun Li, David Hsu, and Wee Sun Lee. “Act to see and see to act: POMDP planning for objects search in clutter”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2016.


