4-2015

Reflective, Deliberative Agent-Based Information Gathering

Adam D. Eck
University of Nebraska - Lincoln, aeck@cse.unl.edu

REFLECTIVE, DELIBERATIVE AGENT-BASED INFORMATION GATHERING

by

Adam Dewane Eck

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Leen-Kiat Soh

Lincoln, Nebraska

April, 2015

REFLECTIVE, DELIBERATIVE AGENT-BASED INFORMATION GATHERING

Adam Dewane Eck, Ph.D.

University of Nebraska, 2015

Advisor: Leen-Kiat Soh

As computational devices and entities become further established as routine, omnipresent components of our everyday lives (e.g., wearable sensors, smart homes, cyber-physical systems, embodied agents, human-robot interactions), such systems face an increased pressure to perpetually understand the complex, noisy, uncertain world around them in real-time. This environmental knowledge enables computational systems to intelligently decide how to best behave in response to the current situation, adapt to the ever-changing conditions of the dynamic world, and accomplish system goals that ultimately aim to improve our daily experience. However, achieving and maintaining such knowledge is very complicated due to the complexities and challenging properties of real-world environments.

In this research, we study how to improve environment knowledge in intelligent agents and multiagent systems through reflective, deliberative information gathering. By being deliberative, an agent intentionally and selectively chooses how to gather information. By being reflective, an agent can self-evaluate its informational needs and performance in order to understand its needs and past sensing outcomes to best guide deliberative information gathering, as well as adapt and learn in an uncertain environment.

Within reflective, deliberative information gathering, this dissertation addresses two key problems: (1) the Analysis Problem, whereby an agent must determine how to measure and balance sensing benefits and costs in order to reflect and improve deliberative information gathering, (2) the Information Sharing Problem, whereby multiple agents must determine how to cooperatively sense together and share information to update collective beliefs.

For the Analysis Problem, we propose two improvements to a popular framework for reasoning under uncertainty—partially observable Markov decision processes (POMDPs): (1) Potential-based Reward Shaping (PBRS) providing metareasoning about information gathering within time-constrained planning, and (2) Difference-based Heuristic Selection (DHS) with Long Sequence Entropy Minimization (LSEM) for situationally-aware planning capable of balancing knowledge improvement and costs minimization. For the Information Sharing Problem, we propose two solutions for improving large team information sharing observing localized, non-stationary phenomena: (3) cooperative change detection and response and (4) forgetting-based adaptation of information sharing. We also propose: (5) a learning-based approach for ad hoc information gathering that enables agents to learn how to share information without requiring pre-coordination.

DEDICATION

To Sophia: Never stop reaching for your dreams.

ACKNOWLEDGEMENTS

I would like to thank those who have assisted with the creation of this dissertation. First, I am very grateful for my advisor Dr. Leen-Kiat Soh, whose guidance, mentoring, thought provoking discussions, meaningful feedback, and careful review have helped me to grow as a researcher, a scholar, and a person. Second, I appreciate the help of my committee members Dr. Stephen Scott, Dr. Ashok Samal, Dr. Steven Dunbar, and Dr. Milind Tambe for their willingness to assist with my research and thoughtful review of my work. Third, I thank my colleagues for the many discussions and critiques that have improved my research, especially L.D. Miller and Nobel Khandaker. Fourth, I acknowledge the resources available at the University of Nebraska used to complete my research, including the Holland Computing Center.

Further, I am forever indebted to my wonderful wife Liz, whose love, support, patience, and sacrifice have allowed me to reach for my dreams. Finally, I thank God for the many blessings in my life and for the gifts He has bestowed upon me. Without either, this research would not be possible.

GRANT INFORMATION

This research has been supported by a NSF Graduate Research Fellowship (DGE-054850), a U.S. Department of Education GAANN Fellowship (P200A040150), as well as grants from the NSF (DBI-0743783, DEB-0953766, SBES-1228937, SES-1132015), and has been partially completed utilizing the Holland Computing Center of the University of Nebraska.

CHAPTER 1 INTRODUCTION 1

1.1. Reflective, Deliberative Information Gathering 3
1.2. Initial Research 5
1.3. Dissertation Problems 8
1.4. Solutions to Dissertation Problems 11
1.5. Dissertation Contributions 15
1.6. Dissertation Outline 17

CHAPTER 2 BACKGROUND AND RELATED WORK 18

2.1. Deliberative Information Gathering 18
2.2. Deliberative Information Gathering with Active Sensing POMDPs 22
2.2.1. Markov Decision Process **..... 23**
2.2.2. Partially Observable Markov Decision Process **..... 23**
2.2.3. Active Sensing POMDP **..... 26**
2.2.4. Applications of the Active Sensing POMDP **..... 27**
2.3. Reflective Information Gathering 30
2.3.1. Reflection for Deliberative Information Gathering **..... 30**
2.3.2. Reflection for the Active Sensing POMDP **..... 33**
2.4. Multiagent Information Gathering with Limited Sensors 36
2.5. Comparison of our Research to Prior Work 38

CHAPTER 3 POTENTIAL-BASED REWARD SHAPING FOR POMDPS (SOLUTION 1 TO THE ANALYSIS PROBLEM) 42

3.1. Introduction 43
3.2. Background 50
3.2.1. Online POMDP Planning **..... 50**
3.2.2. Potential-Based Reward Shaping **..... 54**
3.3. Potential-Based POMDP Planning 58
3.3.1. Extending PBRS to Online POMDP Planning **..... 58**
3.3.2. Impact of PBRS on Online Planning **..... 70**
3.4. Experimental Setup 78
3.4.1. Benchmark Problems **..... 81**
3.4.1.1. Tag **..... 81**

3.4.1.2. RockSample 82
3.4.1.3. AUVNavigation 85
3.5. Results 87
3.5.1. Tag Results 90
3.5.1.1. Comparison of Full Breadth Planning With and Without Reward Shaping 90
3.5.1.2. Comparison Between Potential Function Types 92
3.5.1.3. Comparison of PBRS with Depth-Focused, State-of-the-Art Planning Algorithms 93
3.5.2. RockSample Results 95
3.5.2.1. Comparison of Full Breadth Planning With and Without Reward Shaping 95
3.5.2.2. Comparison Between Potential Function Types 96
3.5.2.3. Comparison of PBRS with Depth-Focused, State-of-the-Art Planning Algorithms 98
3.5.3. AUVNavigation Results 99
3.5.3.1. Comparison of Full Breadth Planning With and Without Reward Shaping 99
3.5.3.2. Comparison Between Potential Function Types 102
3.5.3.3. Comparison of PBRS with Depth-Focused, State-of-the-Art Planning Algorithms 103
3.5.4. Discussion 106
3.6. Conclusions and Future Work 114

CHAPTER 4 SITUATIONALLY-AWARE ONLINE HEURISTIC PLANNING FOR HIGHLY UNCERTAIN ENVIRONMENTS (SOLUTION 2 TO THE ANALYSIS PROBLEM) 117

4.1. Introduction 118
4.2. Background 123
4.2.1. *Online POMDP Planning* 123
4.2.2. *Heuristic Search Algorithms for Online POMDP Planning* 127
4.3. Problem 132
4.4. Solution Approach 136
4.4.1. *Planning Stages* 137
4.4.2. *LSEM Heuristic* 139
4.4.3. *DHS Situational-Awareness* 142
4.4.4. *Theoretical Analysis* 148
4.5. Experimental Setup 153
4.6. Results 160
4.6.1. *AUVNavigation Results* 160

4.6.2. *Tag Results* 164
4.6.3. *RockSample Results* 167
4.6.4. *Discussion* 168
4.7. Conclusions 170

CHAPTER 5 INTELLIGENT INFORMATION SHARING WITH LOCALIZED, NON-STATIONARY PHENOMENA (SOLUTIONS 1 AND 2 TO THE INFORMATION SHARING PROBLEM) 173

5.1. Introduction 174
5.2. LTIS 176
5.2.1. LTIS Model 176
5.2.2. Prior LTIS Research 178
5.3. Non-Stationary Phenomena 180
5.3.1. Modeling Non-Stationarity in LTIS 180
5.3.2. Analyzing the Effect of Non-Stationarity 181
5.4. Change Detection and Response 185
5.5. Forgetting Outdated Beliefs 188
5.6. Experimental Setup 192
5.7. Results 194
5.8. Conclusions 200

CHAPTER 6 AD HOC INFORMATION GATHERING (SOLUTION 3 TO THE INFORMATION SHARING PROBLEM) 201

6.1. Introduction 201
6.2. Problem 205
6.2.1. AHIG Formulation 205
6.2.2. Related Work 208
6.3. POMDP Formulation 209
6.3.1. AHIG as a POMDP 210
6.3.2. Problems with POMDP Formulation 212
6.4. Knowledge State MDP 214
6.4.1. Incorporating Shared Information 214
6.4.2. Knowledge State MDP Transformation 215
6.4.3. Learning Knowledge State Dynamics 218

6.5. Experimental Setup 221
6.6. Results 223
6.7. Conclusions 226

CHAPTER 7 CONCLUSIONS AND FUTURE WORK 228

7.1. Summary 228
7.2. Future Work 232
7.3. Contributions 234

REFERENCES 235

TABLE OF FIGURES

Figure 1.1: Summary of Research 11
Figure 2.1: Comparison to Prior Reflective, Deliberative Information Gathering Research within the Analysis Problem 38
Figure 2.2: Comparison to Prior Multiagent Information Gathering Research within the Information Sharing Problem 40
Figure 2.3: Comparison to Prior Research on Resource Usage during Information Gathering within the Environment Impact Problem 41
Figure 3.1: Performance of Planning Algorithms as Planning Time Increased on the Tag Benchmark Problem for Select Approaches 94
Figure 3.2: Performance of Planning Algorithms as Planning Time Increased on the RockSample Benchmark Problem for Select Approaches 98
Figure 3.3: Performance of Planning Algorithms as Planning Time Increased on the AUVNavigation Benchmark Problem for Select Approaches 104
Figure 3.4: Proportion of AUVNavigation Runs Successfully Ending at a Goal Location as Planning Time Increased for Select Approaches 104
Figure 4.1: (a) Example π Tree with Two Actions and Two Observations with Depth 1, (b) Example Path with Depth n 125
Figure 4.2: Stages of Planning in Highly Uncertain Environments 138
Figure 5.1: Agent Belief Updates 182
Figure 5.2: Impact of Non-Stationarity 184
Figure 5.3: Example of Performing Belief Decay (a) Only Upon Receipt of Information vs. (b) Every Tick 191
Figure 5.4: Impact of Malicious/Faulty Agents under Periodic Sequences of Phenomenon Values 198
Figure 5.5: Impact of Malicious/Faulty Agents under Random Sequences of Phenomenon Values 198
Figure 6.1: Average Belief Certainty 224
Figure 6.2: Average Proportion of Correct Agents 224
Figure 6.3: Average Total Reward 226
Figure 7.1: Summary of Research 228

TABLE OF TABLES

Table 2.1: Related Deliberative Information Gathering Research 19
Table 2.2: Related Active Sensing POMDP Research 28
Table 2.3: Related Reflective Information Gathering Research 31
Table 2.4: Related Multiagent Information Sharing Research 36
Table 3.1: Types of Potential Functions for POMDPs 60
Table 3.2: Summary of Potential Functions Used in Each Benchmark Problem 80
Table 3.3: Results from Tag Benchmark Problem with 95% Confidence Intervals 90
Table 3.4: Results from RockSample Benchmark Problem with 95% Confidence Intervals 95
Table 3.5: Results from AUVNavigation Benchmark Problem with 95% Confidence Intervals 100
Table 3.6: Proportion of AUVNavigation Runs Successfully Ending at a Goal Location with 95% Confidence Intervals 102
Table 4.1: Results on AUVNavigation Benchmark with 95% Confidence Intervals 161
Table 4.2: Results on Tag Benchmark with 95% Confidence Intervals 164
Table 4.3: Results on RockSample Benchmark with 95% Confidence Intervals 167
Table 5.1: Comparison of Solutions with Different Phenomenon and Networks with 95% Confidence Intervals 195
Table 6.1: POMDP Formulation of AHIG Problem 211
Table 6.2: Knowledge State MDP Formulation 217

CHAPTER 1 INTRODUCTION

Many real-world applications of computer systems benefit from the use of artificial intelligence (AI) and **multiagent systems** (MAS). For example, intelligent agents have found wide-ranging uses from intelligent tutoring systems and collaborative learning environments in education (e.g., D'Mello & Graesser, 2012; Khandaker *et al*., 2011) to mixed-initiative systems supporting human users with routine tasks (e.g., Chalupsky *et al*., 2001; Myers *et al*., 2007; Yorke-Smith *et al*., 2009) to search and rescue robots that help discover human victims after disasters (e.g., Casper & Murphy, 2003; Calisi *et al*., 2007).

In particular, an intelligent agent is a unit situated in a specified environment capable of autonomously (1) sensing its environment to gather information about its current situation, (2) using this information to decide how to behave in the environment (e.g., based on internal goals), and (3) taking action to change the environment according to its decisions in order to complete tasks. Through intelligence, hardware or software agents provide features such as reactivity to changing environments, proactive behavior aimed to accomplish goals, learning to improve performance over time, and social behavior to work together to solve complex problems (Wooldridge, 1999). Together, these features enable a system to achieve valuable properties such as reliability, scalability, robustness, consistency, efficiency, and effectiveness.

Achieving these benefits requires an agent to consistently make correct decisions appropriate to its current situation. However, the quality of an agent's decision making depends on the information gathered by the agent from its environment through *sensing*: without good information, even a rational agent could make wrong decisions and thus fail

to accomplish its goals and complete its tasks. Unfortunately, proper sensing is made especially difficult due to challenging properties of environments common to many real-world applications of intelligent agents, including noise, partial observability, non-stationarity, and limited resources.

For example, in a search and rescue robotics application, individual robots could be responsible for autonomously navigating a physical space to discover trapped victims in need of assistance within collapsed buildings after a powerful earthquake. These robotic agents must be able to gather high quality information during sensing in order to know how to navigate through the space and identify all victims so that they can be freed from the rubble. However, the quality of information gathered during sensing by these robots is negatively influenced by their environment. For instance, smoldering fires might resemble the heat signature of a person to an agent's infrared sensor, returning noisy, inaccurate information to the agent. Additionally, the agents' sensors can only view a limited portion of the disaster area at once, so the environment is only partially observable (with portions of the true state of the environment hidden from the agent at any particular point in time). Furthermore, the environment can change while each agent is sensing (e.g., new buildings collapse), causing the prior information collected by agents to become outdated and in need of refresh to maintain accurate, up-to-date beliefs. Finally, the robots are powered by battery supplies and must therefore be careful when consuming limited energy to maximize the amount of area covered and/or their operational time in order to find the most victims. Given that there are also multiple agents (i.e., robots) operating in the same environment, their actions can also work against one another, making sensing even more difficult. For instance, robots might

move in front of each other's sensors, adding noise to the resulting information gathered. Likewise, agents can otherwise change the environment (e.g., creating extra rubble by running into obstacles), making the environment even more non-stationary and requiring more sensing to maintain up-to-date beliefs.

Given the challenges of sensing in complex environments, special care must be taken to make sure that agents appropriately sense to gather information with sufficient quality and quantity to inform their decisions, achieve goals, and complete tasks. We next outline our research vision to address this necessity.

1.1. Reflective, Deliberative Information Gathering

To improve agent sensing in order to benefit agent reasoning and actuation, as well as overall system performance, this research focuses on **reflective, deliberative information gathering**[1] by intelligent agents. By being ***deliberative***, an agent *intentionally* and *selectively* chooses how to gather information, as opposed to considering sensing as a *secondary* behavior, which could instead potentially lead to suboptimal information gathering in complex environments. By being ***reflective***, an agent *self-evaluates* its informational needs and performance in order to understand its needs and past sensing outcomes to best guide deliberative information gathering, as well as *adapt* and *learn* as it faces new decisions in an uncertain environment. Together, these qualities enable an intelligent agent to carefully consider its current knowledge, the knowledge required of its decisions, and the state of its environment in order to know

[1] By "information gathering", we mean both the gathering of raw data/observations from the environment, as well as the transformation of such data into information useful for the agent's reasoning. We use the terms "sensing" and "information gathering" interchangeably throughout this dissertation.

how, when, and where to sense so that it improves the way it gathers the necessary information for its reasoning in an efficient and effective manner.

In contrast, a non-deliberative (i.e., passive) information gathering agent would focus its reasoning *solely* on completing tasks and not explicitly think about how to act to perform good sensing now with the hope of potentially later benefitting its tasks. For instance, a search and rescue robot that pre-computes a path to take through the disaster area and does not periodically adjust its movement or sensor positioning would be a non-deliberative information gathering agent. Furthermore, a non-reflective yet deliberative information gathering agent would not self-evaluate its sensing performance or learn over time how to improve its sensing from past experience. For instance, a non-reflective search and rescue robot might not recognize that continually adjusting its vision camera isn't helping it find new victims due to a lack of ambient light in the collapsed building, and thus the agent would not switch to focus its limited energy resources on more effective infrared sensing in order to better find victims.

Overall, this research both (1) extends classical metareasoning (e.g., Cox & Raja, 2011; Raja & Lesser, 2007; Zilberstein, 2008) from decisions about *reasoning control* to decisions about *sensing control* which benefits both sensing and the agent's task-level decisions, and (2) extends prior research on deliberative information gathering, sometimes called active sensing/perception (e.g., Weyns, Steegmans, & Holvoet, 2004), to be more introspective about agent performance and needs in order to encourage improved adaptation over time.

For instance, in our search and rescue running example, a robotic agent should deliberatively manage its sensors to maintain high quality sensing while moving through

the complex environment terrain. This could include frequently re-aiming its visual camera and infrared sensors to best scan for victims, as well as planning routes to intentionally navigate through areas where the agent has the least knowledge of the presence of victims. To determine how to best deliberatively sense over time, the agent should reflect on what it already knows about the complex environment, as well as the potential benefits of different types of actions (e.g., choosing to enter a room, pointing its camera in a different direction) and the costs of these actions (e.g., consumed battery power, wasted time, possible noise which could corrupt its current knowledge). Following such behavior, the robot should then be able to gather both higher quality information (through choosing the best sensing actions) as well as a greater quantity of information (by lasting longer in the environment before its battery expires). Together, such information better informs the agent's decisions and enables it to find the most victims to rescue.

1.2. Initial Research

Our research on reflective, deliberative information gathering for intelligent agents and multiagent systems was initially inspired by our earlier research (Eck, 2010) studying the Environment Impact Problem:

> **Environment Impact Problem**: How can an agent *mitigate any changes to its environment* caused by sensing that have *lasting impacts* on both the information gathered and the ability of the agent to accomplish its tasks in order to avoid corrupting the environment?

In the Environment Impact Problem, actions taken by agents for the purpose of sensing not only result in gathered information used to *change the agent's knowledge*, but these actions can also *change the agent's environment* and affect its future behavior. In

the Environment Impact Problem, we studied how an agent can reflect to anticipate these changes to the environment and predict their consequences, and then determine how to deliberatively act in order to mitigate or avoid problems caused by environment changes.

One type of environment change we have studied involves the use of *stateful resources* by agents to gather information. As an agent interacts with a stateful resource, the agent can change the state of the resource, causing dynamic (rather than fixed) costs to the agent based on the state of the resource. Furthermore, the quality and quantity of information gathered by a stateful resource depends on its current state, providing greater accuracy or more information in some states than others. We call this effect the **Observer Effect of agent sensing**. Overall, agents must be mindful of the internal state of resources used during sensing (and how its actions change the state) in order to gather the best information at the lowest cost, and we have studied solutions for both modeling stateful resource behavior, as well as approaches for managing usage of such resources.

For example, in a mixed-initiative system application where an intelligent agent works alongside a human user to support the user's daily tasks (e.g., an office worker scheduling meetings (Chalupsky *et al*., 2001; Myers *et al*., 2007; Yorke-Smith *et al*., 2009) or a student learner performing educational assignments (D'Mello & Graesser, 2012; Khandaker *et al*., 2011)), the agent might need to interact directly with the human user (a stateful resource) to gather information and understand the user's preferences so that it can best support the user and her tasks. Such interactions can interrupt and distract the user from her current activities. If done at inopportune times, these interruptions can disrupt the user's cognitive processes (Mark, Gudith, & Klocke, 2008) and increase user frustration (Adamczyk & Bailey, 2004) (the resource states), and cause the user to want

to return quickly to her current activities or even quit using the system (Klein, Moon, & Picard, 2002), altogether affecting the quality and quantity of information provided back to the mixed-initiative agent. Properly managing human-agent interactions to gather information in this example enables us to construct more efficient and effective agents, as well as improve the end-user experience and productivity.

Alternatively, sensing actions taken by agents can also have permanent effects on the environment. That is, an action can produce a change in the environment state that could prevent the agent from ever gathering necessary information or achieving certain tasks and goals. For instance, in our search and rescue example, if a robot chooses to navigate through a dangerous hallway to search for victims, its movement through the hallway could further weaken the structure of the building and collapse other paths, preventing the robot from exploring nearby areas or rescuing other victims in the future. Thus, current actions have an influence on the future abilities of the robot, including its ability to gather information and/or accomplish its goals.

As part of studying reflective, deliberative information gathering, we have also extended our Master's thesis research on the Environment Impact Problem and the Observer Effect (Eck, 2010) separate from this dissertation. First, we have enhanced the formalization of the problem of modeling this effect. We have also improved our POMDP-based solution framework for metacognitively managing agent sensing, which allows the agent to reflect on the impacts of sensing actions with respect to changing both stateful resources and the agent's knowledge, then deliberatively choose sensing actions expected to best improve the agent's knowledge under the Observer Effect. This research has been published in the Journal of Autonomous Agents and Multiagent Systems

(JAAMAS) (Eck & Soh, 2013c). We have also improved our MineralMiner simulation for studying environment impacts from sensing (including both the Observer Effect and permanent effects on the environment), amongst many other environment properties that make sensing a challenging activity. This research has been published in the Multiagent and Grid-Based Systems (MAGS) journal (Eck & Soh, 2013b).

1.3. Dissertation Problems

To better understand both (1) how to produce reflective, deliberative information gathering in intelligent agents, as well as (2) the benefits of this approach for agent-based sensing, this dissertation focus on two core problems: the Analysis Problem, and the Information Sharing Problem.

> **Analysis Problem**: How should an agent *measure or predict the benefits and costs* of performing various sensing actions with respect to gathering information, then *analyze the resulting tradeoff*, in order to best guide its deliberative sensing?

First, the Analysis Problem is at the core of reflective, deliberative sensing: an agent must be able to measure and/or predict the benefits and costs of its actions with respect to its current knowledge and informational needs in order to achieve reflective sensing behavior. Within this problem, we study different methods for performing such measurement and prediction at different levels of agent reasoning. We also study different techniques and approaches for analyzing these measures and predictions in order to best guide deliberative sensing and balance the tradeoffs between sensing benefits and costs.

For instance, in our search and rescue running example, one possible useful measure of sensing benefits is the improvement in the certainty of an agent's beliefs after

gathering new information. Since the location of victims is inherently uncertain, the agent wants to know with high certainty whether a victim is nearby before moving on to another room (lest it accidentally leave victims behind undiscovered). In contrast, the agent might measure the costs of sensing based on the amount of time different sensing actions take (e.g., slowly moving deeper into the room vs. quickly re-aiming its sensors), as well as the limited battery energy required for each action (e.g., a low cost for forward movement vs. a high cost for turning around). Then, analyzing this information, the agent can deliberatively choose the action to continue its sensing that best balances benefits and costs and boost its overall performance.

> **Information Sharing Problem**: How can agents leverage multiagent cooperation in order to *share information when information gathering is limited* (e.g., agents have limited sensors or resources)?

Second, rather than looking at intelligent agents as isolated individuals responsible for their own independent information gathering, we can also look at how cooperative agents can help one another in the sensing process. By combining multiple agents, we can achieve benefits such as increased coverage of the environment (when individual agents suffer from a limited world view through partial observability), timelier sensing (especially in dynamic, non-stationary environments), higher accuracy and faster uncertainty reduction (by combining multiple viewpoints of the environment to avoid noise), as well as better limited resource management.

Towards information sharing, in this research we study the dynamics of information flow through multiple cooperative agents working together as they share information, as well as solutions for (1) determining when and where each agent should sense, (2) how agents should share information with their neighbors, (3) how to incorporate shared information in agent beliefs, and (4) how to share or conserve limited

resources for sensing between cooperative agents. As a team, agents can reflect together on their collective knowledge and informational needs, as well as either cooperatively or individually plan how to deliberatively sense in order to carry out team goals and achieve better sensing as a group rather than as individual agents. In particular, we are interested in environments where the sensing capabilities of agents are limited compared to the size of the team of cooperative agents (e.g., only a few agents have sensors to directly observe the environment). We are also interested in environments, called ad hoc environments, where agents have no prior knowledge of each other or their peers' capabilities and willingness to cooperate, preventing pre-coordination of information sharing behavior.

For instance, in our search and rescue robotics example, a small group of robots developed by different organizations could work together to canvas a damaged building at once, and they belong to a larger team of agents (e.g., emergency responders, dispatchers) that cannot otherwise observe the disaster area. These robots could cooperatively compare their initial knowledge, and then decide how to divide up the area for exploration in order to speed up identification of victims, as well as redundantly overlap their sensing areas to provide additional information to increase overall certainty after searching through the environment for victims. Agents could learn how to weight their own observations versus how much they should trust shared information from their teammates when updating their beliefs. Depending on the circumstances, the robots might frequently communicate with each other to maintain up-to-date beliefs, or they might conserve energy by communicating infrequently to maximize how long they can operate in the environment.

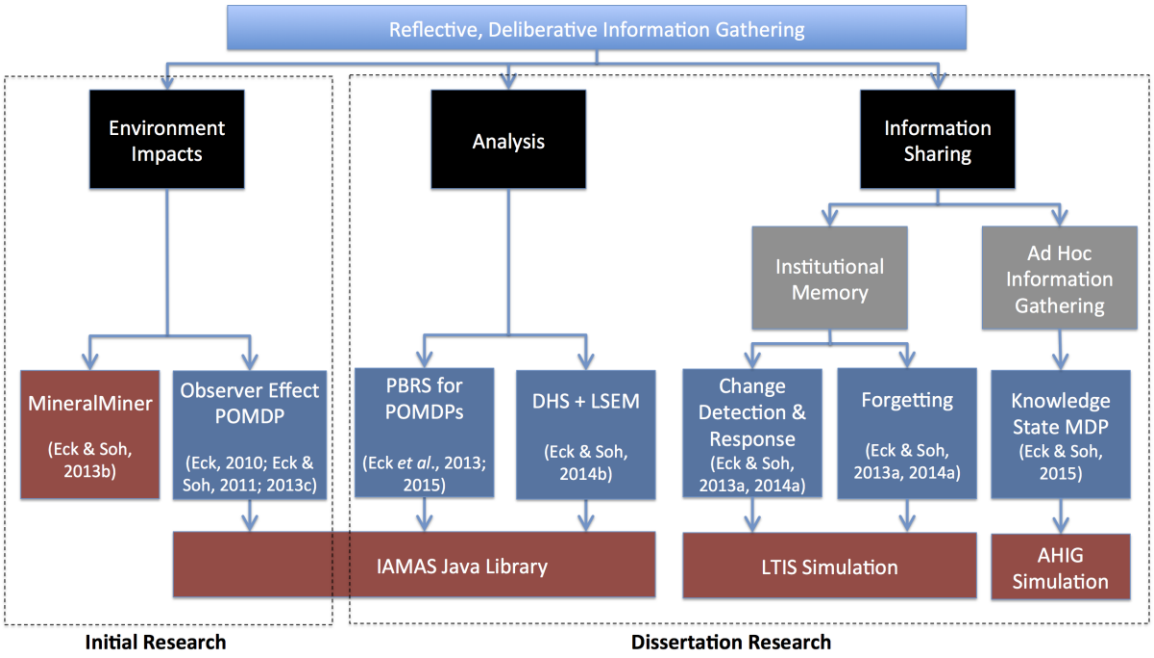

Figure 1.1: Summary of Research

1.4. Solutions to Dissertation Problems

Towards solving these two core problems—the Analysis Problem and the Information Sharing Problem—and better understanding reflective, deliberative information gathering, the research presented in this dissertation has accomplished the following, summarized in Figure 1.1 and described in more detail below.

To address the Analysis Problem, we propose two novel approaches to reflecting on the benefits and costs of sensing actions, then optimizing the resulting tradeoff within a popular framework for agent reasoning (e.g., Boutilier, 2002; Doshi & Roy, 2008; Spaan, Veiga, & Lima, 2010; Williams & Young, 2007): the partially observable Markov Decision process (POMDP) (Kaelbling, Littman, & Cassandra, 1998). These two solutions include: (1) potential-based reward shaping (PBRS) (Ng, Harada, & Russell, 1999; Asmuth, Littman, & Zinkov, 2008) for POMDPs, and (2) difference-based

heuristic selection (DHS) with the long sequence entropy minimization (LSEM) heuristic for situationally-aware heuristic search-based online planning.

First, our PBRS for POMDPs solution is an approach to embed additional measures reflecting action benefits and costs (including with respect to sensing) in reward optimization by agents to produce agent behavior that best addresses the tradeoff between benefits and costs to improve overall agent behavior. Unlike past attempts to include similar information to guide action selection in POMDPs (e.g., Mihaylova *et al*., 2002; Araya-Lopez *et al*., 2010), our approach offers important theoretical guarantees on agent performance. As an additional benefit, this approach also generalizes to a solution for improving agent planning in devices with constrained computational resources (e.g., wireless sensors, robots) by guiding the agent towards large rewards beyond the myopic planning (i.e., limited number of planning steps) caused by a lack of computational power. It also represents a novel technique for adding metareasoning to agent reasoning with POMDPs without increasing the size of the agent's state space (and thus does not increase the computational complexity of the reasoning process). Overall, PBRS both addresses the Analysis Problem studied in this dissertation, as well as offers broader impacts for agent reasoning in general. This research has been published both as an extended abstract (Eck *et al*., 2013) at AAMAS 2013 and more recently as an article in JAAMAS (Eck *et al*., 2015). This solution will be discussed in greater detail in Chapter 3 of this dissertation.

Second, DHS + LSEM represents a novel heuristic search algorithm for online planning in POMDPs. In particular, the LSEM heuristic guides agent planning towards policies (i.e., action plans) that quickly gather the necessary information to operate in

highly uncertain environments (such as those commonly found in real-world applications of multiagent systems). It does so by reflecting on the expected certainty in agent knowledge (a measure of sensing benefit, directly addressing the high uncertainty in the domain) after taking an action in order to determine which action sequences to consider during planning and find a good policy. This work differs from our PBRS solution in that LSEM reflects on sensing benefits when *choosing how to search through the policy space while planning*, whereas PBRS reflects on sensing benefits *during the choice of sensing action to take during execution of plans* (and thus at a different level of agent reasoning). Additionally, DHS provides situationally-aware planning that enables the agent to select between different heuristics measuring different types of information when choosing how to expand planning during plan construction. As such, DHS enables the agent to consider both the benefits of sensing (revealed through LSEM) with other heuristics (reflecting sensing costs) to quickly find approximately optimal policies. Altogether, DHS + LSEM can find good policies two orders of magnitude faster than the best previously reported heuristic search online POMDP planning algorithms. This research was published as a full paper at the AAMAS 2014 conference (Eck & Soh, 2014b) and will be discussed in greater detail in Chapter 4 of this dissertation.

Third, to address the Information Sharing Problem, we first focus on challenging domains with localized phenomenon observed by only a small subset of the agents within a large cooperative team (e.g., observing individual users of a large mixed initiative software system), requiring large team information sharing (LTIS) (Glinton, Scerri, & Sycara, 2009, 2010, 2011; Pryymak, Rogers, & Jennings, 2012) to achieve and maintain consistent and accurate shared beliefs. We produce solutions to overcome a challenging

problem caused by environment non-stationarity: the *institutional memory problem* where large portions of the team of agents become stuck with outdated beliefs as the environment changes (e.g., newly collapsed buildings, changing user preferences or goals), no matter how much additional information enters the team through additional sensing. In particular, we develop two algorithms for mitigating this problem: (1) a change detection and response algorithm where agents work together within local sub-teams to quickly detect changes to the observed phenomenon, and (2) a forgetting-based algorithm, where agents independently use belief decay to maintain up-to-date beliefs to avoid problems caused by faulty agents or malicious information. Both solutions successfully avoid the institutional memory problem and lead to consistent, accurate beliefs through the team as the environment changes. This research has been published as an extended abstract at AAMAS 2013 (Eck & Soh, 2013a) and as a full paper with the WEIN workshop at AAMAS 2014 (Eck & Soh, 2014a). This work will be discussed in greater detail in Chapter 5 of this dissertation.

Fourth, to further address the Information Sharing Problem, we also focus on *ad hoc environments* where agents can either sense on their own or share information with peers, except the agents have no advance knowledge of their peers' capabilities and willingness to work together. Thus, agents cannot pre-coordinate their joint behavior in advance, and instead must learn both when to work together (through sharing) and when to work independently (through sensing with the agent's own sensors) in order best update agent knowledge over time. We propose a solution called the Knowledge State MDP where agents individually learn the benefits of relying on each type of source to maximize knowledge improvement. This research was accepted for publication as a full

paper at the AAMAS 2015 conference (Eck & Soh, 2015) and will be discussed in greater detail in Chapter 6 of this dissertation.

1.5. Dissertation Contributions

The research for this dissertation has made several important contributions to the fields of artificial intelligence and multiagent systems, including:

1. A better fundamental understanding of agent-based sensing in complex environments, valuable for a wide range of intelligent agents and multiagent systems domains. This knowledge can be applied to improve agent reasoning and actuation in different applications, as well as improves our overall understanding of general artificial intelligence.
2. A set of solutions to provide reflective, deliberative information gathering to improve agent-based sensing, including single-agent POMDP solutions and cooperative agent team-based solutions.
3. New techniques for metareasoning by intelligent agents with broader impacts beyond sensing control.
4. Implemented simulation environments mimicking real-world scenarios and applications for studying agent-based sensing.
5. The addition of implementations of many of our solutions to a Java library for artificial intelligence that can be reused for other AI and agent-based projects.

First, from a fundamental research perspective, the dissertation both (1) explores difficult aspects of agent-based sensing in complex environments in order to improve our scientific understanding of the relationship between information gathering and agent reasoning and actuation, as well as (2) produces general-purpose, domain-independent solutions that can be used to engineer agent-based sensing systems in a wide range of domains and real-world applications of intelligent agents and multiagent systems. For example, this research could be applied to applications in autonomic computing; computer supported, collaborative learning; cyber-physical systems; mixed-initiative

systems; robotics; survey systems; ubiquitous and pervasive computing; and wireless sensor networks.

Second, from a broader impacts perspective, the dissertation includes solutions that not only improve agent sensing through reflection and deliberation, but can also improve other aspects of agent reasoning. Specifically, our PBRS for POMDPs solution represents a general-purpose approach to adding metareasoning to the popular POMDP agent reasoning framework. This solution allows not only reflections on the benefits and costs of agent *sensing* to be used to guide action selection, but any measure of benefits and costs across *any* agent goal. Chapter 3 details some other types of measures that the agent can use to reflect on its overall needs and future expectations to improve reward maximization in complex environments. Additionally, our DHS heuristic selection approach to improve online POMDP planning can work with any set of heuristics, not just those maximizing uncertainty reduction to improve agent sensing (e.g., LSEM).

Finally, from a software perspective, the research for this dissertation has resulted in two types of products. First, this research has produced and enhanced simulation environments for evaluating agent-based sensing, including the simulations for large team information sharing and ad hoc information gathering, as well the implementation of many popular POMDP benchmark problems in a unified framework and programming language (Java). Second, combined with the other research activities of the authors, this research has also contributed implementations of our solutions to a Java-based library for general artificial intelligence techniques called IAMAS (which we intend to release as open source software for general, free availability to other programmers and researchers).

1.6. Dissertation Outline

The rest of this dissertation is organized as follows. First, in Chapter 2, we discuss prior work within the agent-based sensing literature in order to frame our research on reflective, deliberative information gathering in the context of the state-of-the-art. We also provide some general background on concepts and techniques used throughout the dissertation. Next, we describe our two solutions for the Analysis Problem in Chapters 3 and 4, respectively: (1) PBRS for POMDPs and (2) the DHS + LSEM heuristic search algorithm for online POMDP planning. Then, in Chapter 5, we detail our research on the institutional memory subproblem of the Information Sharing Problem with solutions. Afterwards, in Chapter 6, we detail our learning-based Knowledge State MDP solution to ad hoc information gathering subproblem of the Information Sharing Problem. In each of these four solution chapters, we also present experimental studies used to evaluate our solutions, as well as investigate the benefits of reflective, deliberative information gathering in agent-based sensing. Please note that these four chapters are each based on our prior publications (aforementioned in Section 1.3). Finally, we conclude in Chapter 7 by summarizing our dissertation research, as well as we outline future work we intend to continue.

CHAPTER 2 BACKGROUND AND RELATED WORK

In this chapter, we describe related research from the intelligent agents and multiagent systems literature to our overall focus of reflective, deliberative information gathering. First, we introduce some general work on deliberative information gathering in Section 2.1. Next, in Section 2.2, we detail more specific work using active sensing POMDPs for deliberative information gathering, which is closely related to our PBRS for POMDPs and DHS + LSEM solution approaches presented in Chapters 3 and 4, respectively, as well as our Knowledge State MDP solution in Chapter 6. Then, we describe prior work that initially added reflectiveness to deliberative information gathering in Section 2.3. Afterwards, we discuss related work from the multiagent sensing literature in Section 2.4. Finally, in Section 2.5, we conclude by discussing how our research on reflective, deliberative information gathering (both from this dissertation and our prior work on the Environment Impact Problem) fits within the context of the state-of-the-art introduced in this chapter.

Along the way, we also introduce some background, including an overview of MDPs and POMDPs in Section 2.2.1-2.2.2, which is relevant to both the related work in Section 2.2, as well as our solutions in Chapters 3, 4, and 6. Background or related work only relevant to specific parts of our research will be introduced later in the appropriate chapters.

2.1. Deliberative Information Gathering

Although the vast majority of intelligent agents and multiagent systems research focuses primarily on the reasoning and actuation components of agent behavior (and thus generally relegates sensing to a *by-product* of other agent activities), research focusing on

Table 2.1: Related Deliberative Information Gathering Research

Reference	Contributions
(Bajcsy, 1988)	Bajcsy advocated for the use of active perception to control information gathering for robotics, which represented one of the earliest calls for deliberative information gathering in agents. They developed a hierarchical approach to improve sensing both locally and globally.
(Floreano & Mondada, 1994)	Floreano & Mondada studied the use of neural networks and genetic algorithms to learn controllers to guide active perception in robotics. Their algorithms resulted in learned automated behavior such as targeted exploration for missing information.
(Grass & Zilberstein, 1997; 2000)	Grass & Zilberstein developed Value-Driven Information Gathering (VDIG) for automating information gathering from the internet to support human users' decisions.
(Lesser *et al*., 2000)	Lesser *et al*. studied resource-Bounded Information Gathering (BIG), including an agent for (goal oriented and opportunistic) planning for information gathering from sources distributed across the internet.
(Weyns, Steegmans, & Holvoet, 2004)	Weyns, Steegmans, & Holvoet developed one of the first *domain-independent* frameworks for active sensing by agents. They studied this framework in the context of situated agents (researching the relationship and connections between an agent and its environment).
(Weyns, Helleboogh, & Holvoet, 2005)	Weyns, Helleboogh, & Holvoet implemented a simulation environment called Packet-World for their study of active sensing.
(So & Sonenberg, 2009)	So & Sonenberg studied the application of active perception for situation awareness in intelligent agents in order to direct an agent's attention to the most interesting or relevant features of the environment for information gathering.

agent sensing as a *primary objective* has recently begun growing in popularity in the literature. In this subsection, we review some of the general history of deliberative information gathering within the agents literature in order to place our research in the context of the state-of-the-art. We summarize this history in Table 2.1. We will further elaborate in Section 2.2 on recent deliberate sensing research using a similar type of solution to our solutions in Chapters 3 and 4.

To begin, Bajcsy (1988) and Floreano & Mondada (1994) were two of the first researchers to explore the needs for (and benefits of) deliberately choosing how to perform sensing in order to improve the quality and quantity of information gathered by

agents. Specifically, both explored an area of research called *active perception*[2] whereby a robotic agent makes control decisions about gathering information used to model the environment, controlling either (1) what raw data to collect as observations during sensing (e.g., active control of vision cameras (Bajcsy, 1988)), or (2) what information to extract from raw data when processing observations from the agent's sensors. Using active perception, Bajcsy (1988) and Floreano & Mondada (1994) advocated that autonomous, intelligent agents could improve their understanding of the world around them, which in turn would improve their ability to complete tasks in the environment. To perform active perception, Bajcsy (1988) considered a hierarchical approach that improved information gathering both locally with respect to individual models of the environment, as well as globally across components used for sensing. Floreano & Mondada (1994), on the other hand, used neural networks and genetic algorithms to learn how to sense in complex environments, resulting in automated behavior such as targeted exploration for missing information.

A few years later, in response to the growing amount of information valuable to human users offered through various web pages and services, Grass & Zilberstein (1997; 2000) developed an agent-based framework called Value-Driven Information Gathering (VDIG) using software agents to choose what information to collect for users, as well as how to collect it, in order to support human users' decisions (e.g., purchasing software online). Similarly, Lesser *et al*. (2000) developed an autonomous, intelligent software agent called BIG (resource-Bounded Information Gathering) that was capable of multilevel planning to choose how to deliberatively gather information from the internet

[2] Recall (c.f., Section 1.1) that in this research, active perception and active sensing are synonymous with deliberative information gathering. "Active" refers to the agent conscientiously (i.e., deliberatively) choosing actions for their sensing or information gathering value.

for human users. Details of how VDIG and BIG performed deliberative information gathering are provided in Section 2.3.

More generally, Weyns, Steegmans, & Holvoet (2004) were one of the first to study the need for general purpose, domain independent approaches for deliberative information gathering by agents. In particular, they studied what they called *active sensing*[3] and focused on improving information gathering for agents as part of their research studying situated agents (i.e., the relationship and connections between agents and their environments). They developed an extensible framework that divides information gathering into three components: (1) *sensing*, which collects raw values from the environment, (2) *interpreting*, where raw observations are converted into domain-specific representations for knowledge, and (3) *filtering*, where only the relevant and/or important observations are retained for knowledge refinement. We take a similar perspective[4] to information gathering in our research (as a process of collecting and transforming raw observations into useful information for refining agent knowledge to support agent reasoning). To control information gathering in a deliberative manner, Weyns, Steegmans, & Holvoet propose that domain-specific optimizations over sensing benefits and costs should be embedded by the developer in the selection of which raw observations to collect in the sensing component, as well as in the filtering of processed observations in the filtering component. As part of this research, Weyns, Helleboogh, &

[3] Again, recall (c.f., Section 1.1) that in this research, active perception and active sensing are synonymous with deliberative information gathering. "Active" refers to the agent conscientiously (i.e., deliberatively) choosing actions for their sensing or information gathering value.

[4] However, we use the terms "sensing" and "information gathering" interchangeably and do not limit the meaning of the term "sensing" to be collecting raw observations, as done by Weyns, Steegmans, & Holvoet (2004)

Holvoet (2005) also developed one of the first testbed environments for deliberative information gathering in their Packet-World simulation.

Similar to Weyns, Steegmans, and Holvoet's (2004) research on situated agents, So & Sonenberg (2009) explored the use of active perception to improve situation awareness within intelligent agents. That is, in order to best understand the agent's current situation in its situated environment, So & Sonenberg advocated the use of active perception to proactively direct the agent's attention to the most relevant or important aspects of its environment for observation (e.g., interesting events or to fill in missing information from the agent's knowledge) and improve upon the traditional belief-desire-intention (BDI) framework (Rao & Georgeff, 1995) for agent reasoning. To guide active perception, So & Sonenberg considered the use of a logical events calculus.

2.2. Deliberative Information Gathering with Active Sensing POMDPs

One popular solution approach to performing deliberative information gathering in the intelligent agent literature is the active sensing (or active perception) POMDP. In particular, the active sensing POMDP has been commonly used to (1) model the dynamics and goals of the deliberative information gathering problem for agents and (2) generate dynamic plans for choosing sensing actions to perform based on the agent's current situation (e.g., Doshi and Roy, 2008; Guo, 2003; Spaan et. al, 2010; Williams and Young, 2007). In this subsection, we first formalize the general POMDP (and the related fully observable MDP) to provide the background necessary for understanding both (1) important prior work in deliberative information gathering, as well as (2) three of our solution techniques for reflective, deliberative information gathering (presented in Chapters 3, 4, and 6 later in this dissertation). Then, we discuss how the deliberative

information gathering problem is commonly modeled within a POMDP. Finally, we provide examples of prior work using active sensing POMDPs for deliberative information gathering. We summarize the related work on active sensing POMDPs in Table 2.2.

2.2.1. Markov Decision Process

Formally, a (discounted, finite state) MDP can be represented mathematically as a tuple $\langle S, A, T, R, \gamma \rangle$ [Kaelbling, Littman, & Cassandra, 1998]. Within this model, $S = \{s\}$ represents the set of *states* of the agent's environment. Since the environment is fully observable, the agent always knows the current state s in an MDP. The agent can perform *actions* from $A = \{a\}$. Taking an action a in state s both (1) earns the agent a reward $R(s, a)$ according to a *reward function* $R: S \times A \rightarrow \mathbb{R}$ and (2) stochastically changes the state of the environment to a next state s'. The *transition function* $T: S \times A \times S \rightarrow [0,1]$ models the probability that action a changes the dynamic environment from state s to s': $T(s, a, s') = P(s' \mid s, a)$.

The agent's goal is to determine a plan of actions called a policy $\pi: S \rightarrow A$ that controls what action the agent takes based on its current state in order to maximize cumulative, discounted rewards:

$$E[\sum_{t=0}^{n} \gamma^t r_t] \tag{2.1}$$

where r_t is the reward received at time t, n is the *planning horizon* (i.e., number of steps to plan ahead), and $\gamma \in [0,1)$ is a discount factor for weighting future, uncertain rewards.

2.2.2. Partially Observable Markov Decision Process

The POMDP, on the other hand, is an extension of the MDP to *partially observable* environments. Formally, a POMDP can be represented mathematically as a

tuple $\langle S, A, Z, T, O, R, \gamma, b_0 \rangle$ with S, A, T, R, γ as in the MDP (Kaelbling, Littman, & Cassandra, 1998). Since POMDPs are used in partially observable environments, the current state of the environment s is assumed to be hidden from the agent. Instead, after each action, the agent receives an *observation* from the set $Z = \{z\}$ that reveals some information about the next state of the environment s'. The *observation function* $O: S \times A \times Z \rightarrow [0,1]$ models the probability that next state s' and action a produce observation z: $O(s', a, z) = P(z \mid s', a)$.

Since the environment state is hidden from the agent at any point in time, the agent faces uncertainty about the current state of the environment. This type of uncertainty is addressed by the agent through maintaining a probability distribution over possible states called a *belief state* $b: S \rightarrow [0,1]$ such that

$$\sum_{s \in S} b(s) = 1 \tag{2.2}$$

$$b(s) \geq 0, \forall s \in S \tag{2.3}$$

so that $b \in \Pi(S)$, where $\Pi(S)$ denotes the set of probability distributions over S.

After taking action a and receiving observation z, the agent's belief state probability distribution b is updated to incorporate the new information using a Bayesian update:

$$b^{a,z}(s') = P(s' \mid a, z, b) = \frac{P(z \mid s', a, b) P(s' \mid a, b)}{P(z \mid a, b)} = \frac{1}{P(z \mid a, b)} O(s', a, z) \sum_{s \in S} T(s, a, s') b(s) \tag{2.4}$$

where $P(z \mid a, b)$ normalizes belief state $b^{a,z}$ so that it remains a valid probability distribution under Eq. 2.2. As the agent performs more and more actions and thus receives more and more observations, its beliefs change from the initial belief state b_0 (the prior distribution over environment states, often a uniform distribution) to a posterior

belief state b_t (after taking t actions and receiving t observations) in order to reduce the agent's uncertainty about the current environment state.

Using the POMDP model, the agent's goal is to maximize the cumulative rewards it earns for taking actions while operating in the environment. Since the agent is uncertain about the current state of the environment, it aims to maximize *expected* rewards:

$$E[r_t] = R(b_t, a_t) = E[R(s_t, a_t)|\, b_t] = \sum_{s_t \in S} b_t(s_t) R(s_t, a_t) \tag{2.5}$$

In order to accomplish this goal, the agent plans a policy $\pi: \Pi(S) \rightarrow A$ (over belief states instead of states, as in an MDP) prescribing an action a to take dependent on the agent's belief state b. The policy is calculated by recursively or iteratively solving the set of Bellman equations to calculate the agent's expected cumulative rewards:

$$V(b_0, \pi) = E[\sum_{t=0}^{n} \gamma^t r_t] \tag{2.6}$$

$$V(b) = \max_{a \in A} Q(b, a) \tag{2.7}$$

$$Q(b, a) = R(b, a) + \gamma \sum_{s \in S} b(s) \sum_{s' \in S} T(s, a, s') \sum_{z \in Z} O(s', a, z) V(b^{a,z}) \tag{2.8}$$

then choosing

$$\pi(b) = \operatorname{argmax}_{a \in A} Q(b, a) \tag{2.9}$$

To plan a policy π satisfying Eq. 2.9, an agent must recursively solve Eqs. 2.7-2.9. This entails iteratively computing values of $Q(b, a)$ for additional belief states $b^{a,z}$ that the agent might experience in the future to accurately calculate the long-term cumulative value from its initial belief state b_0. The tradeoff is that the farther into the future the agent plans, the more accurately it will account for future rewards and thus choose better actions, but deeper planning requires more time and the number of possible future belief states grows exponentially with planning depth n.

Due to the computational complexity of computing policies for large POMDPs, finding exact solutions can be quite difficult. Thus, approximate solutions are commonly employed, which estimate the exact policy the agent should perform. Examples of popular approximate solutions include point-based methods (Shani, Pineau, & Kaplow, 2013) that determine appropriate actions around select belief states the agent might encounter, such as PBVI (Pineau et. al, 2003), Perseus (Spaan and Vlassis, 2005), HSVI (Smith and Simmons, 2004), and SARSOP (Kurniawati *et al*., 2008). An agent can build its policy maximizing expected rewards offline, allowing for more computational time and resources to build a larger policy, then follow the policy while operating online in the environment. Alternatively, an agent can also use more recent methods to interleave planning and execution online to adapt to unforeseen situations, such as state-of-the-art online POMDP planning algorithms (Ross & Chaib-draa, 2007; Ross *et al*., 2008; Zhang & Chen, 2012). We will provide background on online algorithms for POMDPs in Sections 3.2 and 4.2.

2.2.3. Active Sensing POMDP

Most often, the information variables the agent is trying to discern through sensing are represented by the hidden states S in an active sensing POMDP (e.g., Guo, 2003; Doshi and Roy, 2008). Furthermore, factors internal to the agent or external in the environment that can influence the observations gathered by sensing are also represented in the state space, such as user behavior history (Williams and Young, 2007), bookkeeping variables for controlling reasoning (Spaan, Veiga, & Lima, 2010) and remembering history, as well as the state of stateful resources that can corrupt gathered information (Eck, 2010; Eck & Soh, 2011; 2013c). The different sensing actions the

agent can perform to gather information are represented by the POMDP's actions A, and the observations Z reflect information collected that help the agent refine its beliefs about which state is the correct one (i.e., what the true value of the information variables the agent intends to know through sensing). How the agent chooses actions to achieve its goals (e.g., uncertainty reduction, balancing the tradeoff between sensing costs vs. task accomplishment) is controlled by the reward function ρ used in the POMDP. Most commonly, ρ is chosen to be Eq. 2.4 and causes the agent to choose sensing actions that both (1) lead the agent to large future task-based rewards and (2) have low cost. However, other types of reward functions have recently been proposed that add some level of reflection to the agent's sensing action selection, which we will discuss in more detail in Section 2.3. We also propose a more principled way to add reflection to the active sensing POMDP using PBRS for POMDPs in Chapter 3.

2.2.4. Applications of the Active Sensing POMDP

One popular application of active sensing POMDPs is user preference elicitation, whereby the agent gathers information about a human user's preference over a set of items (e.g., products, interest, goals). Such interactions with humans are important for a range of environments, including recommendation systems (e.g., Adomavicius and Tuzhulin, 2005), computer supported collaborative learning systems (e.g., Khandaker et. al, 2011), and personal assistant agents (e.g., Eck & Soh, 2012b; Myers et. al, 2007; Yorke-Smith et. al, 2009). For example, Boutilier (2002) considered an active sensing POMDP for determining user utility functions over a range of items. Additionally, Doshi and Roy (2008) described an active sensing POMDP for first discovering a user's current goal, then acting on the goal to provide intelligent user support. Similarly, Williams and

Table 2.2: Related Active Sensing POMDP Research

Reference	Contributions
(Boutilier, 2002)	Boutilier studied the preference elicitation POMDP for modeling deliberatively gathering information about a human user's preferences.
(Guo, 2003)	Guo cast the classification problem (identifying an unknown object) as a POMDP in order to deliberately choose how to gather information to result in accurate classification.
(Sabbadin, Lang, & Ravoanjanahary, 2007)	Sabbadin, Lang, & Ravoanjanahary developed the epistemic MDP, a specific form of the active sensing POMDP (with no state transitions and only information gathering actions).
(Williams & Young, 2007)	Williams & Young applied POMDPs to the problem of understanding human user speech in an automated telephone dialog system.
(Doshi & Roy, 2008)	Doshi & Roy developed improved solutions for solving the preference elicitation POMDP used to gather information during human-agent interactions.
(Spaan, 2008; Spaan, Veiga, & Lima, 2010)	Spaan studied the use of POMDPs to control information gathering by a team of cooperating robotic and sensor agents in order to enable the team to appropriately respond to events in the local area.
(Cohn *et al*., 2010; Cohn, Durfee, & Singh, 2011)	Cohn *et al*. proposed expected myopic gain algorithms for choosing queries (i.e., information gathering actions) to ask human operators to learn how to act autonomously for the human operator in an MDP using Bayesian inverse reinforcement learning.
(Eck, 2010; Eck & Soh, 2011; 2013c)	Eck & Soh developed the Observer Effect POMDP for controlling information gathering to appropriately use stateful resources and avoid/mitigate the Observer Effect during agent sensing.

Young (2007) considered the problem of determining and responding to user goals during human-agent dialog management. In these problems, the goal of the agent when choosing sensing actions is often to minimize costs from sensing and failed intelligent support (Doshi and Roy, 2008; Williams and Young, 2007), or maximizing the value of information collected during sensing with respect to the user's task (Boutilier, 2002).

Active sensing POMDPs have also been used for other applications of intelligent agent-based systems. For example, Guo (2003) used an active sensing POMDP to control sensing actions used to classify the label of objects in the agent's environment while minimizing sensing costs. Moreover, Spaan (2008; et. al, 2010) used an active sensing POMDP to integrate observations from fixed position cameras and control the movements of a mobile robot to best observe a common area and respond to events and

users in need of assistance. Also, Eck and Soh (2013c) used the Observer Effect POMDP to control sensing with stateful resources to maximize knowledge refinement and minimize distortions in observations from changing the state of resources during sensing.

Furthermore, another model very similar to the active sensing POMDP has also been proposed in the literature. Specifically, *epistemic MDPs* (Sabbadin, Lang, & Ravoanjanahary, 2007) model the environment similar to active sensing POMDPs but exclusively consider *epistemic actions* that only gather information from the environment but do not change the state of the environment. Thus, epistemic MDPs are appropriate for active sensing applications where the primary goal of the agent is to discern the correct state of the environment without having to worry about affecting the environment during sensing. To account for this difference from general active sensing, the state transition probabilities are removed from the standard POMDP model. However, although this relaxation of the POMDP is more concise and has fewer terms in its calculations, Sabbadin, Lang, & Ravoanjanahary prove that the relaxation does not improve the model's complexity except under certain strict conditions (e.g., observations are deterministic[5]). Thus, an epistemic MDP can be represented as an active sensing POMDP without affecting the complexity of the solution by using deterministic state transitions (c.f., Section 2.2.1):

$$T(s,a,s') = \begin{cases} 1 & \text{if } s = s' \\ 0 & \text{else} \end{cases} \tag{2.10}$$

In fact, actions have already been assumed to be purely epistemic in some applications of active sensing POMDPs (e.g., Guo, 2003).

[5] That is, taking the same action resulting in the same state always returns the same observation, but different states and actions can produce the same observation

Finally, similar research has also been proposed in the setting of Bayesian inverse reinforcement learning (with an MDP model of the environment) for deliberatively choosing information gathering actions. Specifically, when an agent needs to learn how to act autonomously in lieu of a human operator according to the human's preferences, Cohn *et al*. (2010; Cohn, Durfee, & Singh, 2011) propose expected myopic gain algorithms for choosing queries to ask the human operator to myopically improve the agent's understanding of either (1) the unknown environment dynamics modeled by the transition function T (Cohn *et al*., 2010), (2) the unknown reward function R (Cohn *et al*., 2010), or (3) the preferred action a for a given state s (Cohn, Durfee, & Singh, 2011).

2.3. Reflective Information Gathering

In this subsection, we next review some of the general history of reflective information gathering. We summarize this history in Table 2.3.

2.3.1. Reflection for Deliberative Information Gathering

In some of the earliest work on reflection in information gathering, Zilberstein (1996; with Russell, 1993) studied how to allocate resources within information gathering in autonomous robots to support the robot's tasks (e.g., movement to a location). In particular, they considered the observation *processing* component of information gathering (i.e., transforming raw observations into useful information for reasoning, such as raw vision pixels into information about the agent's surroundings). The goal of this research was to control how much time was spent on processing information during information gathering to avoid consuming computational resources that could instead be used by the agent's task-oriented reasoning. Thus, the agent faced a tradeoff between time available for reasoning vs. the quality of information necessary for reasoning

Table 2.3: Related Reflective Information Gathering Research

Reference	Contributions
(Zilberstein & Russell, 1993; Zilberstein, 1996)	Zilberstein studied the use of performance profiles to reflect on the computational resources used to process gathered information in order to develop anytime algorithms to control information gathering.
(Grass & Zilberstein, 1997; 2000)	Grass & Zilberstein calculated the value of information collected by sensing actions and reflectively weighed this benefit against sensing costs to control information gathering in VDIG.
(Lesser *et al*., 2000)	Lesser *et al*. evaluated the results of sensing (both goal directed and opportunistic, e.g., costs and uncertainty) in order to plan sensing actions in BIG.
(Padhy *et al*., 2006)	Padhy *et al*. developed an algorithm for sensing frequency control that reflectively compared observations to agent knowledge in order to know when to speed up sensing to understand the dynamic environment vs. when to slow down sensing to conserve limited energy resources in agent-based wireless sensors.
(Krause & Guestrin, 2005; 2007; 2009; Krause *et al*., 2008)	Krause *et al*. studied the Observation Selection Problem to optimize various objective functions (e.g., contamination detection, variance minimization) over gathered information according to cost constraints.
(Mihaylova *et al*., 2002)	Mihaylova *et al*. proposed the use of hybrid reward functions for active sensing POMDPs that consider not only the task-oriented costs and benefits of actions, but also reflectively evaluate expected improvements in agent knowledge (i.e., its belief state).
(Sabbadin, Lang, & Ravoanjanahary, 2007)	Sabbadin, Lang, & Ravoanjanahary proposed several reward functions for their epistemic MDP (a variant of the active sensing POMDP) that reflect on the benefits and costs of sensing actions in order to guide deliberative information gathering.
(Araya-Lopez *et al*., 2010)	Araya-Lopez *et al*. introduced belief-based reward functions for active sensing POMDPs that *exclusively* reflect on the benefits of sensing actions with respect to agent knowledge. They also prove several important theoretical properties of the use of such non-traditional reward functions within POMDPs (e.g., convexity for optimization in POMDP solvers).

(requiring time spent instead on information gathering). Using performance profiles to reflectively model the benefits of sensing per unit of time consumption, Zilberstein developed anytime algorithms to control sensing and optimize the overall behavior of the robot.

Within the VDIG framework (c.f., Section 2.1), Grass & Zilberstein (1997, 2000) compared the agent's *a priori* knowledge about the supported human user's decision with the information available from sources across the internet in order to calculate the value of information with respect to the user's decision (based on the expected utility to the

user of gaining such information), and chose to continue retrieving information for the user so long as the value of information collected continued to exceed the costs (e.g., time to retrieve the information, money paid to an information source on the internet) of gathering such information. Similarly, BIG (Lesser *et al*., 2000) considered the results of sensing to determine how to deliberatively gather information, both (1) from the top down during its multilevel planning by evaluating important properties of its generated plans such as costs (e.g., time and money) and uncertainty, as well as (2) from the bottom up to discover opportunities for low cost sensing to meet its overall information gathering objectives. Together, both VDIG and BIG represent domain-specific frameworks for reflecting on deliberative information gathering that could possibly be extended to more generic approaches for domain-independent, reflective, deliberative information gathering.

Elsewhere in the intelligent agents literature, Padhy *et al*. (2006) created a reflective solution for sensing frequency control within the context of agent-based wireless sensor networks. In an effort to minimize unnecessary limited energy consumption during environment monitoring, they developed an algorithm that compared recent observations to the agent's knowledge about the environment to determine whether or not its observations (and the thus environment being monitored) were dynamically changing. When the observations remained static, the agent's knowledge was still up-to-date, so an agent reduced its sensing frequency to also reduce energy consumption and extend the lifetime of the sensor network. On the other hand, when new observations were unexpected based on the agent's knowledge, the agent increased

the sensing frequency to quickly adapt and build a more up-to-date model of the dynamic environment under observation.

Beyond sensing only with intelligent agents, Krause *et al.* (2008; with Guestrin, 2005; 2007; 2009) studied the Observation Selection Problem (OSP), which looked at how to gather information from a general AI perspective (with or without intelligent agents). Specifically, the OSP cast information gathering as an optimization problem over at least one objective function reflectively measuring the goodness of information (e.g., likelihood of contamination detection by distributed sensors in a monitored space (Krause and Guestrin, 2009), minimizing variance of observed data (Krause *et al*., 2008), or optimizing navigational paths for robotic patrol (Singh *et al*., 2009)) while adhering to various cost constraints. Based on properties of the objective function (e.g., submodularity), they developed greedy solutions that find approximately optimal solutions very quickly, in spite of the fact that the general OSP is NP-Complete, and thus computationally difficult to solve.

2.3.2. Reflection for the Active Sensing POMDP

With respect to the active sensing POMDP, Araya-Lopez *et al*. (2010) have recently advocated the use of a different type of reward function ρ that reflects on the current knowledge of the agent (stored in its belief state b) in order to reflectively guide deliberative information gathering. This type of function, called **belief-based reward functions**, breaks from tradition and ignores individual states (i.e., is not based on $R(s, a)$) and instead calculates a *measure of quality* over the entire belief state. Thus, belief-based rewards reflect the quality of the agent's sensing through its current knowledge refined from observations. This type of reward function is useful as it directly

measures the immediate goal of sensing: to refine the agent's knowledge about its environment. Thus, the agent can directly optimize the quality and/or quantity of information gathered (with respect to its current beliefs) by optimizing a belief-based function.

For example, if the primary goal of sensing is to reduce the uncertainty in the agent's beliefs amongst a set of alternatives, the agent can use expected entropy in its belief state as a measure of uncertainty, then employ the negative of its entropy as its rewards to minimize uncertainty in its beliefs:

$$\rho(b,a) = -H(b) = \sum_{s \in S} b(s) \log_{|S|} b(s) \tag{2.11}$$

This example of a belief-based function is one of the most commonly proposed (e.g., Araya-Lopez et. al, 2010; Mihaylova et. al, 2002; Sabbadin, Lang, & Ravoanjanahary, 2007). Other belief-based reward functions that also reflect on agent knowledge in order to accomplish similar goals include maximizing the expected top belief (an approximation of certainty):

$$\rho(b,a) = \max_{s \in S} b(s) \tag{2.12}$$

when only the top belief is important, or maximizing expected information gain, such as through the popular Kullback-Leibler divergence measure (i.e., relative entropy) (Araya-Lopez et. al, 2010; Mihaylova et. al, 2002):

$$\rho(b,a) = E[KL(b,b^a)] = E\left[\sum_{s \in S} b(s) \log_{|S|} \frac{b(s)}{b^a(s)}\right] \tag{2.13}$$

Furthermore, **hybrid reward functions** represent a way to combine both state- and belief-based rewards in a coherent, principled manner in order to achieve action selection that is both task-oriented and reflective about information gathering. As its name implies, this type of function considers both of the other types simultaneously,

often in the form of a weighted function between the alternative reward types (Araya-Lopez et. al, 2010; Mihaylova et. al, 2002; Eck & Soh, 2012c). Hybrid reward functions are potentially useful because they simultaneously consider both the cost-aware perspective of state-based functions and the sensing benefit-aware perspective of belief-based functions to potentially produce very efficient and effective sensing. For example, an agent might use a combination of expected state-based rewards $R(s,a)$ (Eq. 2.5) and the negative entropy function (Eq. 2.11):

$$\rho(b,a) = w\sum_{s\in S} b(s)R(s,a) - (1-w)\,H(b) \qquad (2.14)$$

to simultaneously consider both the costs and immediate belief improvement benefits of sensing, along with the benefits and costs of stopping sensing to perform its task. Here, w represents a weight balancing the importance of the two types of rewards. This weight can either be fixed *a priori* or adjusted over time in response to both changing environment conditions and/or the performance of the agent.

Furthermore, other types of hybrid functions have also been proposed. For example, Sabbadin, Lang, & Ravoanjanahary (2007) proposed (as one of many reward functions considering beliefs) including costs incurred for all non-terminating sensing actions used to gather information, then rewarding the agent based on a belief-based reward function only for the final step of its policies (i.e., when the agent stops sensing). This is similar to state-based functions in that sensing actions incur costs and positive rewards are received after sensing (to guide the agent towards terminal conditions for sensing, e.g., task accomplishment). However, the final rewards depend on the value of the agent's beliefs rather than any particular state the agent believes is correct.

Table 2.4: Related Multiagent Information Sharing Research

Reference	Contributions
(Glinton, Scerri, & Sycara, 2009; 2010; 2011)	Glinton, Scerri, & Sycara defined the Large Team Information Sharing (LTIS) problem for observing static environment phenomena and: • studied emergent information flow behavior within the team when various problem parameters were changed (e.g., belief update weighting, degree network connectivity) • developed analytical models predicting and describing emergent information flow • produced a distributed algorithm (DACOR) for optimizing information flow to reach consistent, accurate beliefs through the team of agents, and • studied the effect of malicious or faulty agents injecting bad information within the networked team
(Pryymak, Rogers, & Jennings, 2012)	Pryymak, Rogers, & Jennings developed another distributed algorithm (AAT) for the LTIS problem that achieved similar good performance to DACOR without requiring any more network communication than just shared information (i.e., no coordination messages)
(An *et al*., 2011)	An *et al*. studied agent-powered distributed resource allocation for sensing networks applied to environmental weather monitoring.
(Stein, Williamson, & Jennings, 2012)	Stein, Williamson, & Jennings studied information sharing with limited communication resources and developed an algorithm controlling who an agent should communicate with, what information should be shared, and how communication resources should be divided between agents.

2.4. Multiagent Information Gathering with Limited Sensors

Next, we introduce recent related work from the multiagent systems literature describing multiagent sensing when the sensing capabilities of agents are limited (related to our Information Sharing Problem, c.f., Section 1.3). We summarize this related work in Table 2.4.

Most relevant to our own research presented in Chapter 5, Glinton, Scerri, & Sycara (2009; 2010; 2011) introduced and studied the Large Team Information Sharing (LTIS) problem. In LTIS, a very large team (e.g., consisting of more than 1000 agents) work together to form consistent, accurate beliefs about some phenomena in the environment. Only a very small number of agents (relative to the size of the team) posses

sensors that can directly observe each phenomenon of interest, whereas all other agents must rely on shared information from sensor agents to gather information about the phenomenon. Glinton, Scerri, & Sycara (2009) first studied the emergent dynamics of information flow and belief updates throughout such a team observing static phenomena based on different parameters of the network (e.g., belief update weighting representing confidence in neighbors' beliefs, degree network connectivity representing communication pathways and size of sub-teams). Afterwards, they (2010) developed analytic models formalizing the behavior of information flow in such teams, as well as a distributed solution for optimizing the team's convergence to consistent, accurate beliefs. Later, Pryymak, Rogers, & Jennings (2012) produced another distributed solution that improved upon the work of Glinton, Scerri, & Sycara by not requiring additional network traffic to reach good beliefs throughout the team of agents. Finally, Glinton, Scerri, & Sycara (2011) also studied the robustness of information flow when malicious or faulty agents inject bad information into an LTIS team of agents. For more details describing prior work on LTIS, please consult Section 5.2 later in this dissertation.

Beyond LTIS, other recent work has also considered different aspects of information sharing between cooperative agents when sensing is limited. For example, An *et al*. (2011) studied negotiation methods for developing plans allocating limited resources between agents responsible for cooperatively monitoring the environment. This research was applied to weather monitoring in a real-world radar system. Additionally, Stein, Williamson, & Jennings (2012) studied information sharing between cooperating agents consuming limited shared communications resources. In particular, they developed a distributed approach for determining (1) who amongst the team each

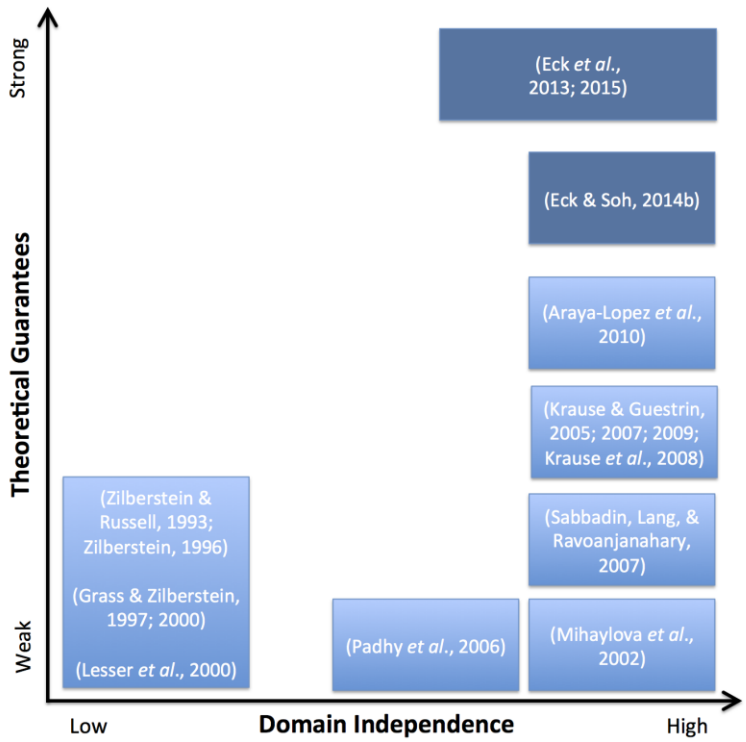

Figure 2.1: Comparison to Prior Reflective, Deliberative Information Gathering Research within the Analysis Problem

agent should communicate with, (2) what information should be transmitted by each agent to avoid overloading shared communication resources, as well as (3) how limited communication channels should be distributed across the team of agents.

2.5. Comparison of our Research to Prior Work

We conclude this related work chapter by placing our dissertation research studying reflective, deliberative agent-based information gathering within the context of the state-of-the-art in the intelligent agents and multiagent systems literature described previously in this chapter.

First, our research studying the Analysis Problem extends prior research on reflective, deliberative information gathering in the following manner, summarized in Figure 2.1. On the one hand, our PBRS for POMDPs and DHS + LSEM solutions represent *domain-independent* solutions that can applied to a wide variety of intelligent

agent applications and domains. This is an improvement over initial reflective solutions developed for deliberative information gathering (e.g., Zilberstein & Russell, 1993; Zilberstein, 1996; Grass & Zilberstein, 1997; 2000; Lesser *et al*., 2000). In particular, our DHS + LSEM solution works off the shelf to add reflection about sensing benefits to any problem using POMDPs for planning, whereas our PBRS for POMDPs solution enables both domain-independent and domain-dependent measures of action benefits and costs (including towards sensing and knowledge refinement) to be considered during reflective metareasoning to improve overall agent performance.

On the other hand, our two solutions also provide *stronger theoretical guarantees* with respect to improving agent reasoning and actuation (through reflective information gathering) than the state-of-the-art. Whereas prior research has primarily focused on theoretically understanding (1) problem complexity (e.g., Krause & Guestrin, 2007; Sabbadin, Long, and Ravoanjanahary, 2007), or (2) applicability for use within prior deliberative information gathering techniques (e.g., Araya-Lopez *et al*., 2010), our two solutions add additional guarantees that (1) an approximately optimal solution can be found in finite time (Eck & Soh, 2014b) (c.f., Section 4.4.4), (2) metareasoning can best benefit the agent when adequate sensing is most difficult (Eck *et al*., 2015) (c.f., Section 3.3.2), and (3) including metareasoning doesn't change the objective function being optimized by the agent and thus should improve the overall performance of the agent (Eck *et al*., 2015) (c.f., Section 3.3.2). This is especially important because we have previously demonstrated (Eck & Soh, 2012c, 2012d) that the aforementioned belief-based and hybrid reward functions (Eqs. 2.11-2.14) (Araya-Lopez *et al*., 2010; Mihaylova *et*

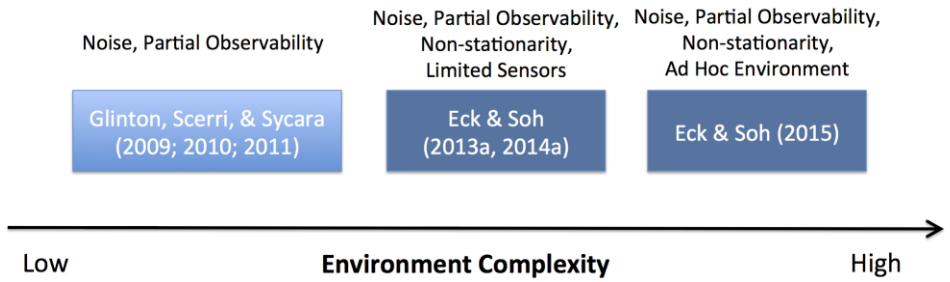

Figure 2.2: Comparison to Prior Multiagent Information Gathering Research within the Information Sharing Problem

al., 2002; Sabbadin, Long, & Ravoanjanahary, 2007) used to provide reflective metareasoning about sensing to the popular active sensing POMDP can lead to complicated (and not necessarily beneficial) relationships between reflective information gathering and overall agent performance, even in two relatively simple active sensing POMDPs (with very small state, action, and observation spaces).

Second, our Information Sharing Problem research extends prior research on *multiagent* reflective, deliberative information gathering in the following manner, summarized in Figure 2.2. First, our research on the flow of shared information in LTIS (c.f., Chapter 5) extends prior research studying this problem to consider *non-stationary* environments that change over time, and thus require more complicated sensing control to not only *reach* consistent, accurate beliefs about environment phenomena of interest to the team's reasoning, but also *maintain* such beliefs as the phenomena change over time. Additionally, our other research on the Information Sharing Problem studies how to share information in ad hoc environments, where agents have no prior knowledge of their peers' capabilities or willingness to cooperate. Thus, we study *more complicated environments*, such as those agents are likely to experience in real-world applications.

Finally, our additional research studying the Environment Impact Problem extends prior research on reflective, deliberative information gathering in the following

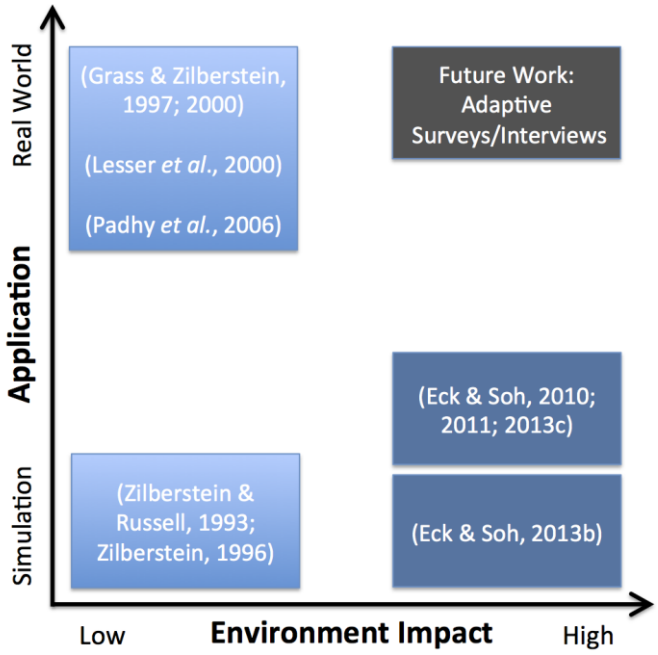

Figure 2.3: Comparison to Prior Research on Resource Usage during Information Gathering within the Environment Impact Problem

manner, summarized in Figure 2.3. In particular, although prior research has studied the use of limited resources during sensing, such as computational resources (Zilberstein & Russell, 1993; Zilberstein, 1996) or energy resources (Padhy *et al.*, 2006), little research has focused on how the use of such resources can change the state of the environment and thus impact the observations collected by the agent during information gathering. In our prior work studying the Observer Effect within the Environment Impact Problem (Eck, 2010; Eck & Soh, 2011; 2013c), we began studying such impacts on the quality or quantity of information gathered by agent sensing when using stateful resources whose behavior change as they are used by agents for sensing. However, our own prior work only studied environment impacts *in simulation*. As part of our future work (c.f., Chapter 7), we intend to study the Observer Effect in a *real-world application* of reflective, deliberative information gathering – an intelligent agent for producing adaptive surveys/interviews for collecting information from human respondents.

CHAPTER 3 POTENTIAL-BASED REWARD SHAPING FOR POMDPS

In this chapter, we present our first solution to the Analysis Problem (c.f., Section 1.3) within the context of POMDPs, a popular approach to deliberative information gathering (c.f., Section 2.2.2). Taking inspiration from the related field of reinforcement learning (RL), our solution is to shape the agent's reward function with information reflecting the quality of its sensing (e.g., knowledge refinement) to guide the agent towards actions that both best improve its knowledge (represented by belief states), as well as allow it to achieve its tasks with high reward.

However, this approach also solves a greater general problem in the POMDP literature: creating plans to achieve high, cumulative rewards with only short, finite horizons (i.e., planning steps n, Eq. 2.6). The same technique we use to imbed reflection on agent knowledge refined through sensing (potential functions from PBRS) can also be used to provide hints of where the agent might find high future rewards beyond its planning horizon, and thus achieve greater cumulative rewards over time (reflection on sensing outcomes being one such type of hint). As such, this chapter is written to address the greater finite horizon problem, and was recently published in the Journal of Autonomous Agents and Multiagent Systems (Eck *et al*., 2015). We theoretically prove several important properties and benefits of using PBRS for online POMDP planning and empirically demonstrate these results in a range of classic benchmark POMDP planning problems.

This research is joint work with our collaborators Dr. Sam Devlin and Dr. Daniel Kudenko of the University of York in the United Kingdom.

3.1. Introduction

Partially observable Markov decision processes (POMDPs) (Kaelbling, Littman, & Cassandra, 1998) have become a very popular approach to agent reasoning and planning, such as for robotics (e.g., Mihaylova *et al*., 2002; Spaan, Veiga, & Lima, 2010) and human-agent interactions (e.g., Boutilier, 2002; Doshi & Roy, 2008; Williams & Young, 2007). POMDPs explicitly model complex environment dynamics, such as partial observability of environment states revealed through actions, as well as changes to environment state resulting from actions. Using such information, agents can (1) discover the true environment state hidden by partial observability in order to reduce the uncertainty in its beliefs and make more informed decisions, and (2) plan action sequences that maximize expected rewards given its uncertain beliefs.

Reducing the time spent (i.e., the computational complexity) on planning with POMDPs has been a topic of much research in the literature (e.g., Kurniawati, Hsu, & Lee, 2008; Ong *et al*., 2010; Pineau, Gordon, & Thrun, 2003; Ross & Chaib-draa, 2007; Silver & Veness, 2010; Smith & Simmons, 2004; Somani *et al*., 2013; Spaan & Vlassis, 2005; Zhang & Chen, 2012). This is especially important for online POMDP planning (Ross *et al*., 2008), where an agent interleaves planning and execution as it operates in the environment and must therefore plan quickly due to real-time constraints. Ultimately, the agent's goal when planning is to calculate a good estimate of the cumulative, future rewards from its current situation dependent on different actions it could take in order to choose how to behave in the environment. In most problems, this requires being able to plan many steps in advance in order to form good estimations of future rewards. Unfortunately, the complexity of *optimal* planning is *exponential* in the planning horizon

(i.e., the number of steps the agent looks ahead during planning). Moreover, the complexity is also *polynomial* in the size of the state space, which is often quite large (necessary to adequately capture and reflect the nuances of real-world environments). Therefore, planning far enough in advance across all possible future situations is prohibitively expensive (due to time constraints), and thus agents are commonly restricted to forming *approximately* best plans, rather than acting optimally, which reduces their ability to maximize long-term rewards and achieve correct, goal-directed behavior.

In order to provide the most useful cumulative, future reward estimations, many of the state-of-the-art approaches to online planning *sacrifice the breadth of planning* in order to enable the agent to *plan farther in advance* for *certain* situations, thereby forming better estimations of the rewards (and thus better understanding how to act) in those situations. The success of this type of approach depends on the agent's ability to select (in advance) the correct scenarios it will indeed face. Two common such approaches to planning include (1) expanding plans *selectively* along attractive belief states (according to some heuristic function) using heuristic search (e.g., AEMS2 (Ross & Chaib-draa, 2007)), or (2) sparse random sampling of situations biased towards highly probable state/action/observation sequences and high estimated rewards using Monte Carlo search techniques (e.g., DESPOT (Somani *et al*., 2013)). So long as the heuristic chosen in heuristic search methods or the sampling performed in Monte Carlo methods expands plans along the correct situations towards high future rewards and goal accomplishment, these approaches have demonstrated an ability to form plans equally as good as the state-of-the-art offline planners where time constraints are more relaxed and

agents can afford greater breadth and depth of planning (Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013; Zhang & Chen, 2012).

However, it would be ideal for a POMDP planning algorithm to achieve accurate cumulative, future reward estimations *without* having to sacrifice the breadth of planning. Indeed, sacrificing breadth can be inherently detrimental to the agent's behavior in several ways. For example, depth-focused planning algorithms can cause an agent to fail to adequately consider scenarios it might actually encounter in the near future when executing the plan (i.e., if they are unattractive according to the chosen heuristic in heuristic search algorithms or if they are not quite as likely as other scenarios in Monte Carlo methods), and thus the agent could end up in a position where it does not know what to do in order to adequately achieve its goals. In complex, real-world applications of intelligent agents and multiagent systems, such a predicament could even pose imminent danger to the agent (e.g., a search and rescue robot exploring a damaged building in a section about to collapse) or affect the quality of the system (e.g., increased human user frustration caused by improper interactions from a mixed-initiative software agent). Additionally, in problems requiring long action sequences to achieve large rewards (e.g., highly uncertain environments requiring large quantities of information gathering), even depth-focused planning algorithms might fail to adequately plan far enough down to discover large future rewards and thus underestimate the value of the best actions, leaving it potentially confused on how best to act, or even overvalue suboptimal actions (that achieve greater intermediate rewards but lower cumulative rewards in the long run). This, too, can cause the agent to reach undesirable situations that make it difficult for the agent to achieve its goals in the long run.

Overall, it would be advantageous for an agent if it could *implicitly* estimate cumulative, future rewards without requiring time-consuming, *explicit*, depth-based calculations so that it can achieve the best of both worlds: *allowing time for full breadth of planning*—to avoid the potential pitfalls described above—and also *creating better estimations of cumulative rewards* over the long term. This should produce a planner that is both safer to use in complex environments and still achieves high rewards over time and ultimately goal achievement. In this chapter, we explore how to perform implicit future reward estimation within full breadth planning.

In particular we consider a popular technique for implicitly guiding agents towards large future rewards from the related field of reinforcement learning called **potential-based reward shaping** (**PBRS**) (Asmuth, Littman, & Zinkov, 2008; Devlin & Kudenko, 2011; 2012; Ng, Harada, & Russell, 1999) and apply this technique to online POMDP planning. In this context, PBRS uses additional information about the agent's current situation (represented by belief states in POMDPs) measured by **potential functions** reflecting the potential of earning large future rewards from any particular situation in order to shape the rewards maximized by the agent. That is, this additional information guides the agent to optimistically take actions leading to situations (i.e., belief states) likely to earn large future rewards beyond its planning horizon, thereby enjoying the benefits of deeper planning without suffering from the would-be computational costs.

Although PBRS has previously been applied to planning in less complex fully observable Markov decision processes (MDPs) (Sorg, Singh, & Lewis, 2011) and can be

seen as an extension of leaf evaluation heuristics[6] (e.g., Ross *et al*., 2008; Sorg, Singh, & Lewis, 2011) to anytime planning, this first application of PBRS to online POMDP planning provides additional insights and benefits previously unreported. Specifically, we discover and provide several novel contributions to both the PBRS and online POMDP planning literature:

1. A novel characterization of different categories of potential functions that provide different indications of which situations are favorable to the agent (beyond its available planning horizon) for earning greater quantities of cumulative, future rewards, including both domain-specific and domain-independent expertise. Previous research has not distinguished between different types of potential functions, and this categorization helps us understand what types of potential functions might be useful in different problems.
2. Two novel types of potential functions unique to POMDPs exploiting different properties of belief states: (a) the agent's knowledge about the environment represented as a probability distribution, and (b) a sufficient statistic representing the history of interactions by the agent with its environment. Such types capture and exploit information not considered previously in the use of PBRS or leaf evaluation heuristics for planning, enable agent metareasoning with POMDP planning, and prove to be very useful for earning large rewards by agents in an empirical study.

[6] Sorg, Singh, & Lewis (2011) also propose applying their optimal reward framework to MDPs, which is slightly different from PBRS in that it allows path-dependent reward modifications (as opposed to shaping only values at leaf and initial situations in PBRS, c.f., Section 3.2). However, they note that in full breadth planning (as considered in this chapter), optimal rewards are equivalent to leaf heuristics, and thus also to PBRS. Therefore, for the remainder of the chapter, we only refer to leaf evaluation heuristics, but the same discussions apply to optimal rewards, as well.

3. Several theoretical results describing the benefits of using PBRS during online POMDP planning, including (a) for any finite horizon of planning depth, PBRS can result in different plans found than the approximately best plan found without PBRS, making it possible to achieve plans closer to the actions within the (infinite horizon) optimal policy when using a potential function that is a good indicator of future rewards, (b) PBRS has the greatest ability to produce plans that are better in the long term when using the shortest horizons, making it a good choice for online planning with real-time constraints, (c) even though PBRS modifies the reward function maximized by the agent, the (infinite horizon) optimal policy under PBRS is the same as the (infinite horizon) optimal policy to the original reward function, so using PBRS still targets plans that optimize the agent's goals and task accomplishment (i.e., using PBRS is still working towards the same objective, even if it finds different, and hopefully better, policies when using finite horizon planning), and (d) so long as the potential function is convex, the shaped reward calculations remain convex and can thus be solved by a wide range of popular POMDP solvers.
4. A comprehensive experimental study investigating the empirical performance of PBRS for online POMDP planning using 20 different potential functions across multiple benchmark problems with different properties, as well as an identification of the benefits and weaknesses of PBRS when compared against state-of-the-art heuristic search and Monte Carlo planning approaches commonly used for online POMDP planning. In particular, we discover that combinations of potential functions including both (a) domain-specific information (as done

elsewhere in the PBRS literature) and (b) forms of metareasoning about agent knowledge and/or histories of agent interactions with the environment (both novel for POMDPs and proposed in this research) results in *improved full breadth planning by implicitly estimating cumulative, future rewards*, and *performs very competitively with (and often exceeding) depth-focused state-of-the-art online POMDP planning algorithms.*

Overall, these contributions demonstrate the usefulness of employing PBRS to improve online POMDP planning. PBRS enables full breadth planning (for more comprehensive planning by considering all nearby reachable situations from the current one) to achieve greater cumulative reward estimation *implicitly*, as other approaches intend to do *explicitly* at the cost of needing to sacrifice breadth of coverage due to limited time constraints on planning. These contributions also provide additional insights into the types of information measurable by potential functions that can be useful to improve agent reward accumulation, which could be used to improve the use of PBRS in other settings (beyond online POMDP planning, e.g., partially observable reinforcement learning).

The rest of this chapter is organized as follows. Section 3.2 provides important background for understanding our approach, including a discussion of POMDPs, online planning, and PBRS as originally formulated for RL. Section 3.3 introduces our approach and contains proofs for several important theoretical properties of the policies found during online POMDP planning with PBRS. Section 3.4 describes the experimental setup used to empirically evaluate the performance of online POMDP planning with PBRS on several benchmark POMDP problems, followed by the analysis of our results

and a discussion of the broader implications of this work in Section 3.5. Section 3.6 concludes with a summary of our approach and findings, as well as additional suggestions for future work that we intend to explore.

3.2. Background

3.2.1. Online POMDP Planning

Online planning is one approach to policy construction. In online planning, an agent iteratively (1) plans a policy π from its current belief state b while operating in the environment, then (2) executes that policy for a while before returning to (1) and repeating the process. By interleaving planning and execution, the agent focuses its planning efforts on beliefs it *actually* encounters in the environment, allowing it to adapt to unlikely and unexpected situations, as well as not waste valuable resources planning for many unencountered beliefs. These properties are especially beneficial in real-world applications where agents operate in real-time and cannot estimate in advance all possible encountered beliefs (e.g., robotic exploration).

Because the agent interleaves planning and execution while operating in the environment, online planning is usually restricted to limited amounts of time it can afford for planning. This requirement of quick planning requires the agent to plan for a limited number of steps ahead (i.e., limited depth) and/or a limited number of possible belief states imminently reachable from the current belief state (i.e., limited breadth).

Among online planning approaches, several different methods have been proposed that deal with time constraints during planning in different ways in order to produce the best estimates of cumulative, future rewards (c.f., Ross *et al*. (2008) for a recent survey of online planning methods). Generally, these approaches represent the

agent's policy as a tree with belief states represented by nodes, whereas actions and observations are represented by branches between belief states (where an action and observation from one belief state produces another belief state, as in Eq. 2.4). As the tree is expanded, the algorithms use the new actions and belief states added to the tree to update the estimated cumulative rewards from the agent's current belief state (using Eqs. 2.6-2.8). Thus, planning has two parts: (1) constructing the tree by expanding nodes as time permits, and (2) evaluating the value of action sequences within a tree according the agent's reward function to form the policy of actions to take. Different existing algorithms for online POMDP planning primarily differ in how they choose to expand the tree to best estimate cumulative rewards within the limited amount of time allotted for online planning.

Two of the most popular categories of online planning algorithms include heuristic search methods and Monte Carlo search methods. First, **heuristic search methods** (e.g., AEMS2 (Ross & Chaib-draa, 2007), FHHOP (Zhang & Chen, 2012)) focus planning on the *most attractive* beliefs. Iteratively, heuristic search methods choose to expand the plan from the leaf belief state in the policy tree that maximizes some heuristic function. This heuristic function measures how informative each leaf belief state is towards improving the quality of the plan. For example, state-of-the-art heuristic search algorithms (e.g., AEMS2 (Ross & Chaib-draa, 2007)) rely on heuristics measuring both (1) the error bounds on the value function V as **leaf evaluation heuristics** (i.e., additional upper and lower bounds on future rewards added to the value of a belief state), reflecting the uncertainty introduced by the belief state into the agent's overall

plan, as well as (2) whether or not the belief state is reached by actions that optimistically maximize the upper bound on future rewards.

Second, **Monte Carlo search methods** (e.g., Rollout (Bertsekas & Castanon, 1999), POMCP (Silver & Veness, 2010), DESPOT (Somani *et al*., 2013)), also called Monte Carlo Tree Search (MCTS) when used with tree-based policy representations, perform sparse random sampling of future belief states to estimate cumulative, future rewards. In particular, these methods expand plans by sampling situations that have (1) high probabilities in the state transition and observation functions to focus planning on the *most likely* sequences of agent beliefs, and (2) earn greater rewards under the current reward estimations.

Both heuristic search methods and Monte Carlo search methods commonly result in *depth-focused planning* since (1) heuristics like AEMS2 favor expanding belief states along optimistically optimal sequences of actions (determined by the upper bound on future rewards), and (2) biased sparse random sampling prefers expanding sequences of belief states that have the greatest likelihood of occurrence. As discussed in Section 3.1, this focus on depth is advantageous because it allows agents to form more accurate estimations of the cumulative, future rewards along the deep expansion paths by recalculating Eqs. 2.6-2.8 repeatedly for the parent belief states along these paths. That is, it suffers less from over- and under-estimation of future rewards on chosen action/belief sequences by explicitly searching many steps in advance. So long as the heuristic function or biased random sampling identifies the correct belief states for which to plan between the agent's current belief state and its goal, then the heuristic search or Monte Carlo search methods should work quite well in practice, as indeed shown through

several experimental studies (e.g., Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013; Zhang & Chen, 2012).

However, increasing the depth of planning along select paths in the policy tree requires the agent to *sacrifice the breadth of planning* within the tree due to limited time constraints. Specifically, heuristic search methods neglect belief states with high (but not quite maximum) heuristic value, and random sampling in Monte Carlo search methods avoids less likely but certainly possible belief state sequences. In many situations, especially in complex environments, planning for these other belief states could be very beneficial to improving the overall quality of the agent's plan and its estimation of cumulative, future rewards. That is, sacrificing breadth can also lead to suboptimal policies within the (deeper) finite horizon used for depth-focused planning due to over- or underestimation of the value of the computed policy since the agent fails to explore all possible belief state transitions within the policy tree, possibly missing unexpected high rewards that follow from actions and belief state transitions that are myopically suboptimal and not chosen for expansion. As discussed in Section 3.1, sacrificing the breadth of planning can also cause the agent to reach dangerous or undesirable situations with no forethought on what to do or how to reach a better situation in order to eventually achieve its goals.

Additionally, heuristic search methods (and some Monte Carlo search methods) generally require the agent to have computed rough policies offline before using online planning in order to calculate the upper and lower bounds on the value of actions in belief states that are used to guide planning. However, if the agent is placed in a complex environment (e.g., robotic exploration) where the agent has high uncertainty in what

situations it will face or if the size of the POMDP is very large, appropriate pre-planning might be prohibitively expensive.

In Section 3.3, we explore an approach to online POMDP planning that does not require sacrificing breadth of coverage during planning, yet improves the ultimate actions chosen from planning by enabling the agent to *implicitly* look beyond a limited planning horizon when valuing the actions and belief state transitions within the planning horizon, enabling better long-term reward maximization. Our approach is most similar to heuristic search methods for online planning in that it evaluates the quality of belief states for more than just immediate rewards. However, our approach does not limit expanding plans only along selected belief states with high heuristic value. Instead, the approach modifies the rewards considered at each belief state to bias the agent to place higher value during short, finite horizon planning on policies with greater long term cumulative rewards (even if such policies are otherwise suboptimal within the short, finite horizon). Furthermore, our approach does not require information from precomputed plans, although it can exploit such information if available. We will further describe in more detail in Section 3.3.1 the fundamental differences between our approach and those described previously in this section.

3.2.2. Potential-Based Reward Shaping

Potential-based reward shaping (PBRS) was originally proposed by Ng *et al*. (1999) as a method to provide hints on how to achieve greater long-term rewards as the agent learns the reward function in RL. PBRS addresses one important challenge within RL commonly known as the *exploration-exploitation problem*: determining how to best improve the agent's learned knowledge whilst simultaneously maximizing long-term

reward (Eq. 2.1). PBRS handles this challenge by embedding *a priori* information about the *potential* of states to provide the agent with more valuable rewards. Using this information, the agent is encouraged to choose actions that explore states of high potential in order to learn about these states and hopefully earn greater future rewards while operating in the environment.

Within PBRS, a potential function $\phi(s)$ defined over states encodes or measures such *a priori* information. For example, in a path finding application (e.g., Asmuth, Littman, & Zinkov, 2008), a good potential function might evaluate the inverse of the agent's distance from the goal location, which returns greater values for states (i.e., agent locations in the maze) closer to where the agent earns large rewards (the goal location).

In order to guide the agent during RL, PBRS *shapes* the rewards considered during action selection in Eq. 2.1 by adding an additional amount determined by the potential function. Specifically, PBRS considers the following reward:

$$r_t = R(s_t, a) + F(s_t, a, s_{t+1}) \tag{3.1}$$

$$\text{where} \quad F(s_t, a, s_{t+1}) = \gamma\Phi(s_{t+1}) - \phi(s_t) \tag{3.2}$$

Here, Eq. 3.2 represents the difference in potential future rewards due to moving from state s_t to s_{t+1}. Shaping r_t by adding this value provides additional motivation to the agent to choose actions that *increase* the potential of earning future rewards. Therefore, by maximizing this representation of r_t in Eq. 2.1, the agent targets actions that improve its learning and are more likely to lead to larger rewards. Once the rewards are learned for those high potential states, the agent can then exploit its learned knowledge to maximize long-term rewards.

Furthermore, it can be shown (see the proof for Theorem 3.4 for similar details) that when planning over an infinite horizon, the same policy optimizes rewards with and

without PBRS (Asmuth, Littman, & Zinkov, 2008; Ng *et al*., 1999). Therefore, using PBRS does not change the (infinite horizon) optimal policy and, due to targeted exploration, results in faster learning convergence to the optimal policy and higher cumulative unshaped rewards than only using the original reward function R. This equivalence property of the (infinite horizon) optimal policy is one of the primary advantages of using PBRS to guide exploration in RL (Asmuth, Littman, & Zinkov, 2008; Devlin & Kudenko, 2011; 2012; Ng *et al*., 1999).

Extending beyond RL, PBRS has also been used to improve planning in fully observable domains using a Markov decision process (MDP) (e.g., Sorg, Singh, & Lewis, 2011), which has the same mathematical framework as RL but knows the model parameters *a priori*. In the context of MDPs, PBRS uses a potential function to guide the agent to favor policies found during planning that are likely to lead to large future rewards (equivalent to the use of leaf evaluation heuristics (Sorg, Singh, & Lewis, 2011)). This prior work inspired our own extension of PBRS (which is the *first to formally consider partial observability*) to POMDPs, where guiding planning towards future rewards is especially important when working with limited planning time due to the increased complexity caused by handling *partial* observability, as motivated previously.

Of note, POMDPs can be viewed as a special case of the MDP called a (continuous state) *belief MDP* (Kaelbling, Littman, & Cassandra, 1998), where the state of the MDP represents the current belief state and the state transition function encompasses all the necessary details of belief state changes (e.g., factoring in observation probabilities). Thus, upon first glance, using PBRS for planning with

POMDPs is a relatively straightforward extension of the prior research employing this technique with MDPs. However, the novelty of the research presented here is not in the extension itself (for which we supply the necessary details), but in *realizations about the characteristics of potential functions* and the *discovery of different types of information useful to evaluate the value of plans in POMDPs for finding better approximations of the (infinite horizon) optimal policy when planning with only small, finite horizons*. In previous PBRS research, only a single type of potential function has been defined: potentials over individual states (Eqs. 3.1-3.2, c.f., Type 1 in Section 3.3.1), whereas in leaf evaluation heuristics research, another type (c.f., Type 4 in Section 3.3.1) is commonly used. However, the richness of belief states as probability distributions representing both agent knowledge about the environment, as well as histories of agent interactions with the environment, open up *additional exploitable opportunities* available when using PBRS with more complex POMDPs, rather than simpler, fully observable MDPs. In particular, we identify *two novel types of information* measurable by potential functions in POMDPs not achievable in MDP*s* or fully observable reinforcement learning, including opportunities for metareasoning through reflecting upon the quality of agent knowledge or the history of the agent's actions in order to guide improved action selection. Indeed, we rely on a feature of POMDPs that make planning more complicated in general (handling partial observability through probabilistic beliefs) and turn it instead into an advantage in designing good potential functions that improve planning. Ultimately, both the identification of the existence of different types of available potential functions, and the consideration of the types of information used in our novel potential

functions, could inspire better usage of PBRS in other settings (especially to the very complicated partially observable reinforcement learning).

3.3. Potential-Based POMDP Planning

In this section, we describe the extension of PBRS to online POMDP planning. Whereas PBRS has been considered previously for planning in fully observable MDPs (Sorg, Singh, & Lewis, 2011), this is the first consideration of PBRS for planning within POMDPs. Thus, we first briefly explain the general thought process behind the extension and the transformative steps from prior usage of PBRS with MDPs required to use PBRS with POMDPs. We next identify several different types of potential functions possible with POMDPs and introduce several novel types that exploit the nature of belief states to provide a richer set of information than considered previously with PBRS. We also prove several important results describing the impact of planning with PBRS on both (1) the policies favored during online POMDP planning, and (2) the optimality of planning.

3.3.1. Extending PBRS to Online POMDP Planning

Overview: We begin by noting that in RL (or MDPs), the agent makes decisions based on the environment *state* s. This is why the potential function $\phi(s)$ is defined over states. In POMDPs, the environment is only partially observable, and thus the agent rarely knows the true state of the environment. Instead, the agent makes decisions based on its uncertain *belief state* b, which represents the agent's probabilistic beliefs over which possible state is the correct one. Therefore, since decisions are made over belief states in POMDPs, the first fundamental step of our extension is to define potential functions over belief states: $\phi(b)$.

Here, the potential function represents *a priori* information about the potential of an agent to reach high future rewards from any particular belief state b. Shortly, we will detail different classes of information such a potential function can encode or measure, including novelties to using PBRS with POMDPs (as opposed to fully observable RL and MDPs, as previously considered).

To include $\phi(b)$ in POMDP planning, we define analogous equations to Eqs. 3.1-3.2 for POMDP rewards:

$$r_t = R(b_t, a) + F(b_t, a, b_{t+1}) \tag{3.3}$$

$$\text{where} \quad F(b_t, a, b_{t+1}) = \gamma\Phi(b_{t+1}) - \phi(b_t) \tag{3.4}$$

As in RL, the reward r_t for Eq. 2.6 is shaped by adding the difference in potential in changing belief from b_t to b_{t+1}.

Within the context of POMDPs, we now establish several different ways that the potential function can measure different classes of information based on belief states, each indicators of future rewards. In addition to considering domain-dependent information about individual states (as done previously with PBRS in both fully observable RL and MDPs), an agent can also consider information based on the nature of belief states as probability distributions representing an agent's knowledge about the environment. That is, an agent can directly reason about what it knows (or does not know) and/or the quality of its knowledge through evaluating these probability distributions as a form of reflective, deliberative metareasoning. The agent can then relate its current knowledge to its task at hand in a potential function to predict the future rewards it will earn. As we will explain below, this provides two key implications: (1) extending PBRS to POMDPs enables a *richer set of information* to be considered by potential functions during planning to result in better plans, and (2) this information can

Table 3.1: Types of Potential Functions for POMDPs

Potential Function Type	Description
Domain-Dependent Information from Expected State Potential	Expected value of domain-dependent information encoded in state-based potential functions (extended from prior uses of PBRS in RL and MDPs) . These represent potential functions of the type commonly used with PBRS.
Domain-Independent Information	Measures of the intrinsic quality or a property of a belief state (as a probability distribution over hidden states), such as certainty in the agent's beliefs. These represent a novel type of potential function useful for metareasoning about the quality of agent knowledge.
Belief Prioritization	Preferential ordering on belief states to encourage agents to reach certain belief states before others (based on domain expertise). These represent a novel type of potential function useful for metareasoning about the history of an agent's interactions with its environment.
Approximation of Optimal Value Function	Approximations of the optimal value function from a leaf belief state (and thus the optimal potential function by directly measuring future rewards) based on pre-computed policies using algorithms such as Fast Informed Bound and Blind (Hauskrecht, 2000). These represent leaf evaluation heuristics commonly used in online POMDP planning (e.g., Ross *et al*., 2008).

be *abstracted beyond the agent's particular domain* and can be reused across applications in characteristically different domains, which is in stark contrast to PBRS for *fully* observable environments where potential functions have traditionally been tailor-made for the agent's particular domain. We summarize our categorization of four proposed types of potential functions in Table 3.1.

Potential Function Type 1 (Domain-Dependent Information from Expected State Potential): First, the information encoded in a potential function might be *domain-dependent information* about environment states, similar to the usage of PBRS in fully observable RL and MDPs. In this case, an extension of the potential function to belief states would measure the *expected* potential over states (analogous to Eq. 2.5), based on the probabilities assigned to each environment state in the belief state:

$$\phi(b) = \sum_{s \in S} b(s)\phi(s) \qquad (3.5)$$

This type of potential function is a simple extension of prior potential functions to handle the uncertainty present in partially observable domains. It retains the benefits of exploiting domain-dependent expertise about individual states that have led to the success of PBRS in fully observable RL and MDPs. However, this type of potential function is limited in that each potential function must be carefully constructed for the application and domain at hand, limiting reuse across domains. It is also difficult to apply to a new domain where little domain expertise is known, or domains that are very complicated with many possible environment states (as common to many real-world applications of POMDPs, e.g., robotic exploration).

Potential Function Type 2 (Domain-Independent Information): On the other hand, by reflecting upon a belief state as a probability distribution representing the agent's current knowledge about the environment (i.e., beliefs about the likelihood that any particular environment state is the correct one), we can produce additional types of potential functions unique to POMDPs that relate additional classes of information to the potential of the agent to earn future rewards. Improving upon the first type of potential function described above, this information can be *domain-independent* and apply across multiple applications and domains with differing characteristics, allowing for generalized solutions having applicability to any domain (especially useful when domain expertise is limited or difficult to capture within especially large POMDPs, such as those with many possible hidden states).

In particular, a POMDP potential function might measure some *quality* or *property* of the probabilities in a belief state to predict future rewards. Such behavior is independent of any particular environment state (differing from traditional potential

functions) and can also be independent of the domain where the POMDP is being employed for planning. For example, in many domains and applications of POMDPs (e.g., active sensing (Boutilier, 2002; Spaan, Veiga, & Lima, 2010)), one of the primary goals of the agent is to discover the environment's hidden state before it acts on its beliefs to achieve tasks and goals. In such an application, it does not matter which particular state is the hidden one, only that the agent discovers the hidden state. Therefore, an important property of a belief state related to the ability of the agent to accomplish its goals and earn large future rewards is the *certainty* in its distribution. That is, when an agent is more certain, it is closer to discovering the true state of the environment and can soon earn large rewards for accomplishing its goal. Considering agent certainty in this manner enables the agent to self-reflect on its own beliefs and metacognitively choose actions that will best revise its knowledge, using potential functions as a form of metareasoning to improve agent behavior. Certainty in a belief state can be measured in several ways, each representing a domain-independent potential function leading the agent towards large future rewards. One method for measuring certainty is to consider the entropy in the agent's belief state, more specifically by using the negative[7] entropy in the belief state (e.g., Araya-Lopez *et al*., 2010, c.f., Eq. 2.11):

$$\phi(b) = 1.0 + \sum_{s \in S} b(s) \log_{|S|} b(s) \tag{3.6}$$

Alternatively, an agent can quickly estimate its overall certainty by considering the probability assigned to the most likely environment state in the belief state:

$$\phi(b) = \max_{s \in S} b(s) \tag{3.7}$$

[7] We consider the negative of the entropy since entropy measures uncertainty, which is the reciprocal of certainty.

As the agent's overall certainty increases, so too does the probability assigned to the most likely state, so this potential function can serve as a good proxy for overall certainty. This potential function exploits another possible property of the POMDP and belief state in order to speed up computation. That is, this function is especially advantageous in large, complicated domains where the state space in a POMDP is represented as a *factored state space* comprised of multiple state variables: $S = S_1 \times S_2 \times ... \times S_m$ (c.f., Section 3.4.1.2 for an example used in our experiments). In a factored state space, a belief state can be represented more compactly by a set of conditional probability distributions between variables. Exploiting the structure of these conditional probability distributions can sometimes be more efficient than dealing with the entire joint probability distribution, allowing the most likely state to be identified with lower computational complexity than finding the entropy of the belief state (Eq. 3.6) or some other property of a belief state that requires iterating over all possible states.

Of note, this type of potential function is very closely related to belief-based rewards proposed by Araya-Lopez *et al*. (2010), which directly reward the agent based on measurable qualities of belief states (including Eq. 3.6). However, there is both (1) a lack of theoretical understanding of the impact on agent policies from belief-based rewards, which we provide (in the next section) by including such measures as potential functions within PBRS, and (2) a lack of empirical evidence of their usefulness on POMDP benchmarks, which we provide in the context of PBRS in Section 3.5.

Potential Function Type 3 (Belief Prioritization): Additionally, since belief states represent both (1) an agent's knowledge about the current state of the environment, and (2) a sufficient statistic describing an agent's history of observations (Kaelbling, Littman,

& Cassandra, 1998), they can be used to determine *preferential orderings* on an agent's actions and beliefs, which can be encoded in a potential function. In some applications, a domain expert might have some knowledge about strategies for plans that could be used to achieve an agent's goals, but specific details about how to implement those strategies could be lacking. That is, an expert might know that to achieve its goal, the agent needs particular knowledge about particular states (e.g., that the state is either highly likely or unlikely) before it can complete its task or learn about another particular state. Or the expert might know that certain observations are beneficial, but it is unknown how to achieve those observations. In either case, a potential function can assign higher value to belief states that include certain knowledge (e.g., a particular state is highly likely or unlikely) or are only reachable after certain observations.

This is a way of encoding domain expertise about agent beliefs that *strategically* guides the agent to achieve certain beliefs before others, without necessarily requiring prior knowledge about how to *tactically* achieve those beliefs. In turn, this approach possibly speeds up an agent's knowledge acquisition so that it can accomplish tasks and goals faster, requiring less planning and achieving faster and greater reward accumulation.

For example, consider a robotic agent[8] responsible for gathering information about the quality of a set of rocks $r \in R$. The agent's goal is to determine with near perfect certainty whether each rock is good or bad before moving on to another area of interest. In this situation, a potential function could assign higher priority to belief states that reflect histories where the agent has tested every rock and determined whether each

[8] This example is based on the RockSample benchmark problem described in more detail in Section 3.4.1.2 and used in our experimental study evaluating the empirical performance of PBRS for online POMDP planning.

is good or bad in order to guide the agent to take actions that perform the necessary sensing as quickly as possible. Assuming a binary state variable for each rock (representing a good or bad state), the agent's belief state would be almost perfectly certain a rock was good if $b(r) > 0.99$ and almost perfectly certain the rock was bad if $b(r) < 0.01$. Then the potential function:

$$\phi(b) = \begin{cases} -1000 & \text{if } \{r \in R \mid 0.01 < b(r) < 0.99\} \neq \emptyset \\ 0 & \text{else} \end{cases} \quad (3.8)$$

represents a potential function that prioritizes beliefs (by penalizing beliefs representing histories where the agent has not tested and determined the state of every rock), thereby encouraging the agent to perform its sensing as soon as possible. Moreover it does so without directly explaining to the agent how to do so, and thus represents strategic (instead of tactical) advice.

Potential Function Type 4 (Approximation of Optimal Value Function): Finally, since potential functions are equivalent to leaf evaluation heuristics in planning (Sorg, Singh, & Lewis, 2011), the optimal potential function is the (domain-dependent, infinite horizon) optimal value function $V^*(b) = V(b, \pi^*)$ under the (infinite horizon) optimal policy π^*, since this function exactly measures the future rewards earned from a belief state when following the optimal policy in the agent's particular application. Thus, such a potential function contains exactly the information missing from approximate planning, overcoming the problems addressed in this chapter. However, such optimal policies and value functions are rarely computable or known in practice (or else we would not need techniques such as PBRS in the first place), so the best we can often do is to approximate these values.

Within the heuristic search online POMDP algorithm literature (e.g., Ross & Chaib-draa, 2007; Ross *et al*., 2008; Zhang & Chen, 2012), it is common to approximate $V^*(b)$ using upper and lower bounds on the value function: $\overline{V}(b)$ and $\underline{V}(b)$, respectively, with $\underline{V}(b) \leq V^*(b) \leq \overline{V}(b)$, frequently employed as leaf evaluation heuristics (e.g., Ross *et al*., 2008). These approximations are calculated using policies π_{FIB} and π_{Blind} formed offline using algorithms such as Fast Informed Bound (FIB) and Blind (Hauskrecht, 2000), such that $\overline{V}(b) = V(b, \pi_{FIB})$ and $\underline{V}(b) = V(b, \pi_{Blind})$. With these approximations, we can then define potential functions $\phi(b) = \overline{V}(b)$ and $\phi(b) = \underline{V}(b)$. The tighter the bounds (depending on the application), the better these approximations estimate the optimal value function and thus better guide the agent to optimal rewards.

By using $\overline{V}(b)$ and/or $\underline{V}(b)$ as potential functions, PBRS is able to include the key heuristic information used to guide planning in state-of-the-art heuristic functions without limiting the breadth of planning, and thus not leave the agent in possibly dangerous situations where it reaches a belief state for which it has performed minimal advance planning. Of note, this type of potential function does require offline computations, so this type has the same pre-deployment costs associated with other online POMDP planning approaches discussed in Section 3.2.1, which could be problematic in large, complex real-world problems.

Discussion: Overall, potential functions over belief states can include information (1) about individual states (Type 1, as previously considered with PBRS in RL and MDP planning), (2) about direct estimations of future rewards from a belief state (Type 4, as previously considered with leaf evaluation heuristics), and/or (3) about belief states themselves *independent* of individual states, in both domain-independent and domain-

dependent manners (Types 2 and 3). This enables a *richer set* of information to be embedded during reward shaping for guiding online POMDP planning towards greater future rewards than previously considered in the PBRS literature.

Moreover, amongst the two novel types of potential functions (Types 2 and 3) discovered in this research, reflecting on (1) agent knowledge to determine how to act (e.g., measuring the quality of knowledge about the current state of the environment as indicated by certainty measures, Eqs 3.6-3.7) or (2) the history of the agents' interactions with the environment (e.g., through priority orderings on belief states both currently experienced and soon reachable) both represent metareasoning methods for improving general reasoning in POMDPs with interesting potential applications in many domains (e.g., better information gathering in active sensing applications).

Comparing PBRS with other types of approaches to online POMDP planning, we see that shaping rewards is advantageous because the shaped amount encourages the agent to place higher value on action sequences that can potentially lead to higher future rewards, including *beyond* the planning horizon. Thus, planning with a potential function can allow the agent to estimate cumulative, future rewards (or at least maximize indicators possibly correlated to large future rewards, such as belief certainty) in order to better evaluate the long term values of taking different actions while planning only within short finite horizons without having to spend the limited time on deep planning. As a result of these time savings, the agent can instead maintain a breadth of planning to avoid the pitfalls identified in Section 3.1, such as suboptimal finite horizon planning due to not considering all belief states, and avoiding reaching dangerous or undesirable situations with no forethought on what to do or how to reach a better situation in order to eventually

achieve its goals. Moreover, implicitly estimating future, cumulative rewards can possibly achieve superior action selection than spending time explicitly building such estimates with depth-focused planning, if the agent faces a problem where very long sequences of actions are required to reach the goal from its current situation, and there is not enough time to plan for such a long sequence, even with depth-focused approaches.

Additionally, when comparing our proposed PBRS approach to other types of online POMDP planning, we note that there is a distinct difference in the way the potential function values are considered versus (1) how heuristic function values are used in heuristic search methods, or (2) how probabilities and reward estimations are used in Monte Carlo search methods. In our proposed approach, potential function values are *never* used to *control* planning – they are not used to guide which belief states are expanded in the policy tree at any point in time during planning. In heuristic search methods, on the other hand, the heuristic values calculated for each belief state do indeed determine which belief state is expanded next, in order to guide depth-focused planning, by selecting some belief states for which to plan and excluding others. Likewise, in Monte Carlo search methods, the calculated probabilities for transitions between belief states and reward estimations are used to control how the plan is expanded in a depth-focused fashion. Instead, in our approach, we propose performing a simple breadth-first search (BFS) to consider all belief states within the short, finite horizon, which does not require special control of plan expansion, in order to maintain the breadth of planning and achieve the benefits previously described.

That is, the reward shaping performed by our inclusion of potential functions does not cause some belief states to be considered or excluded during planning[9] (as controlled by heuristic functions and random sampling), but instead changes the *evaluation of the value of action sequences* by adding domain-dependent or domain-independent information about belief states reached by those action sequences in order to place greater value on policies that have the potential to achieve greater long term, cumulative rewards, even if those action sequences would not be considered optimal under the short, finite horizon used for planning with only the original reward function. In the next subsection, we provide theoretical results illustrating how the evaluation of the value of policies is changed with reward shaping, as well as the benefits of this change.

Finally, comparing PBRS to the leaf evaluation heuristics, we note that although the two approaches are functionally equivalent (Sorg, Singh, & Lewis, 2011), there are still advantages to studying and employing PBRS for online POMDP planning. First, PBRS and its mathematical framework (especially Eqs. 3.3-3.4) are the natural extension of leaf evaluation heuristics to anytime online planning algorithms. That is, such algorithms might not know in advance how long they will have to run, and instead must be capable of both (1) returning a plan at any point in time, and (2) continually running as more time is allotted to improve the quality of the plan calculated. Thus, an anytime online planning algorithm might not know in advance when it will stop. In turn, it will not know in advance which nodes will be leaves in the final policy tree, so it will not necessarily know where to apply the leaf evaluation heuristics. The difference function

[9] On the other hand, if we used potential function values to determine how to expand plans, then they would simply represent heuristic functions and the result would be a standard heuristic search algorithm. Since our potential functions are used instead for the evaluation of action values, potential functions are orthogonal to heuristic functions.

(Eq. 3.4) in PBRS incrementally considers each node to be a leaf (and is evaluated with a potential function as a leaf evaluation heuristic), then removes that additional shaped value when a node in the policy tree ceases to be a leaf (as the tree is expanded while time is still allocated for planning). Therefore, the mathematical framework for PBRS defines the calculation procedure for employing leaf evaluation heuristics in anytime online planning algorithms, and the theoretical analyses below informs us on how both PBRS and leaf evaluation heuristics would perform in anytime online planning. Second, unlike the leaf evaluation heuristics commonly used in the literature (our Type 4 potential functions), the first three potential function types proposed above do not require any precomputation before operating in the environment. Thus, an agent using PBRS can operate without having to do any work in advance, which is important when (1) the problem domain is very large and precomputations are prohibitively expensive, or (2) the agent must be quickly reconfigured to deploy to multiple environments (e.g., search and rescue robotics).

3.3.2. Impact of PBRS on Online Planning

Because incorporating PBRS into online POMDP planning involves shaping the rewards the agent wants to earn, the policies formed using shaped or unshaped rewards could be different. This provides us with a dilemma. On the one hand, due to time constraints in online planning, we want to find better policies with PBRS since any policy found is only optimal over the finite horizon used for planning, and thus only *approximately* optimal over the infinite horizon. As such, the policies found during planning can suffer from over- and under-estimation problems (which PBRS is intended to address), as described in Section 3.2.1. On the other hand, since PBRS entails

maximizing shaped rewards with the addition of the potential function, we do not want to sacrifice the ability to optimize the original reward function R over the long run (i.e., infinite horizon), which is, after all, the ultimate goal of the agent.

To better understand the relationship between the value of policies with respect to shaped (with PBRS) and unshaped (original) rewards, we evaluate these values from the theoretical perspective. We follow a similar approach taken to understand the values of policies with and without PBRS in RL (e.g., Asmuth, Littman, & Zinkov, 2008).

In the following, we develop several key results. First, Lemma 3.1 derives the difference in the valuations of an arbitrary policy both with and without reward shaping over the finite horizons used for planning. This represents the difference between how good a policy looks under one approach or the other. Next, Theorem 3.2 establishes the conditions (Eq. 3.13) for which PBRS can lead the agent to a different policy than the original reward function when performing finite horizon planning, based on the results of Lemma 3.1. In conjunction, Remark 3.3 observes the condition (small planning horizons n) when a greater number of potential functions might lead PBRS to different policies than planning without reward shaping. Afterwards, Theorem 3.4 considers the relationship between (infinite horizon) optimal policies with and without reward shaping to establish that reward shaping still causes the agent to optimize its original reward function over the infinite horizon, in spite of working on a modified objective function. Remark 3.5 then extends this result (based partly on the proof to Theorem 3.4) to observe that PBRS also performs well as the planning horizon increases, regardless of the potential function chosen. Finally, Theorem 3.6 establishes a sufficient condition for the

objective function (Eq. 2.6) with shaped rewards (Eq. 3.3) to remain convex and thus still be solvable by a wide range of POMDP solvers.

We begin by computing the difference between the values of a policy for a finite horizon n. This captures the impact of using PBRS with online planning for short horizons required due to time constraints.

> **Lemma 3.1**. *Let $S, A, \Omega, T, O, R, b_0, \gamma$ from the definition of a POMDP be given, and let $n \in \mathbb{N}$ be a fixed planning horizon, ϕ be a potential function over belief states, and π be a policy of action. Then the difference between the value with PBRS $V^{PBRS}(b_0, \pi)$ of π starting at b_0 and the value using unshaped rewards $V^{orig}(b_0, \pi)$ is given by:*
>
> $$V^{PBRS}(b_0, \pi) - V^{orig}(b_0, \pi) = \gamma^n \sum_{b_n \in \Pi(s)} P(b_n | \pi, b_0) \phi(b_n) - \phi(b_0) \quad (3.9)$$

Proof. For notational convenience, we denote the unshaped reward earned at each step $t \in \{0, 1, \ldots, n-1\}$ as R_t:

$$R_t = R(b_t, a_t) = r_t^{orig} \quad (3.10)$$

and the shaped reward earned at each step t as $R_t + F_t$:

$$R_t + F_t = R(b_t, a_t) + F(b_t, a_t, b_{t+1}) = r_t^{PBRS} \quad (3.11)$$

where b_t denotes the belief state after performing t actions and $a = \pi(b_t)$ is the action chosen according to policy π.

As an intermediate result, consider an arbitrary history $H = \{b_0, a_0, o_1, b_1, \ldots, b_n\}$ (i.e., a fixed sequence for a particular experience in the environment) consisting of (1) the actions taken by the agent according to policy π, (2) the resulting observations, and (3) the sequence of beliefs after making those observations. For fixed n, the value using unshaped rewards of any policy π according to particular history H can be computed as the cumulative reward series:

$$V^{orig}(b_0, \pi, H) = \sum_{t=0}^{n-1} \gamma^t r_t^{orig} = \sum_{t=0}^{n-1} \gamma^t R_t \quad (3.12)$$

and the value using shaped rewards of the same policy π:

$$\begin{aligned}
V^{PBRS}(b_0, \pi, H) &= \textstyle\sum_{t=0}^{n-1} \gamma^t r_t^{PBRS} \\
&= \textstyle\sum_{t=0}^{n-1} \gamma^t (R_t + F_t) \\
&= \textstyle\sum_{t=0}^{n-1} \gamma^t \big(R_t + \gamma\phi(b_{t+1}) - \phi(b_t)\big) \\
&= \textstyle\sum_{t=0}^{n-1} \gamma^t R_t + \sum_{t=0}^{n-1} \gamma^{t+1}\phi(b_{t+1}) - \sum_{t=0}^{n-1} \gamma^t \phi(b_t) \\
&= V^{orig}(b_0, \pi, H\,) + [\textstyle\sum_{t=1}^{n-1} \gamma^t \phi(b_t) + \gamma^n \phi(b_n)] - [\sum_{t=1}^{n-1} \gamma^t \phi(b_t) + \phi(b_0)] \\
&= V^{orig}(b_0, \pi\,, H) + \gamma^n \phi(b_n) - \phi(b_0)
\end{aligned}$$

Because this result holds for arbitrary history H starting at arbitrary b_0, it will hold for *any* sequence of beliefs when following policy π. Therefore, since the valuation of a policy from a belief state is the expected value over all possible histories (Eq. 2.6), we find that:

$$\begin{aligned}
V^{PBRS}(b_0, \pi) &= E[V^{PBRS}(b_0, \pi, H)] \\
&= E\big[V^{orig}(b_0, \pi, H) + \gamma^n \phi(b_n) - \phi(b_0)\big] \\
&= E\big[V^{orig}(b_0, \pi, H)\big] + \gamma^n E[\phi(b_n)] - E[\phi(b_0)] \\
&= V^{orig}(b_0, \pi) + \gamma^n \sum_{b_n \in \Pi(s)} P(b_n|\pi, b_0)\phi(b_n) - \phi(b_0)
\end{aligned}$$ ■

where $P(b_n|\pi, b_0)$ is the probability of transitioning to b_n when following policy π from initial belief b_0, considering the probabilities of the necessary state transitions and observations required to reach b_n. From this result, we can subsequently find the following theorem:

> **Theorem 3.2:** *Let $S, A, \Omega, T, O, R, b_0, \gamma$ from the definition of a POMDP be given, and let $n \in \mathbb{N}$ be a fixed (finite) planning horizon and ϕ be a potential function over belief states. Then, the policy π' optimizing V^{PBRS} will differ from the policy π optimizing V^{orig} over the fixed horizon n, provided that*

$$V^{orig}(b_0, \pi) - V^{orig}(b_0, \pi') < \gamma^n \textstyle\sum_{b_n \in \Pi(S)} \phi(b_n)[P(b_n|\pi', b_0) - P(b_n|\pi, b_0)] \quad (3.13)$$

Proof. Consider policy π that optimizes unshaped rewards V^{orig} over finite horizon n. If there is another policy π' satisfying Eq. 3.13, meaning that the difference in

the value of π and π' under the original reward function R is less than the difference in the expected (discounted) potential values along the planning horizon, then:

$$
\begin{aligned}
&V^{PBRS}(b_0,\pi') - V^{PBRS}(b_0,\pi) \\
&\quad = \left[V^{orig}(b_0,\pi') + \gamma^n \textstyle\sum_{b_n \in \Pi(S)} P(b_n|\pi', b_0)\phi(b_n) - \phi(b_0)\right] \\
&\qquad\quad -\left[V^{orig}(b_0,\pi) + \gamma^n \textstyle\sum_{b_n \in \Pi(S)} P(b_n|\pi, b_0)\phi(b_n) - \phi(b_0)\right] \\
&\quad = \left[V^{orig}(b_0,\pi') - V^{orig}(b_0,\pi)\right] + \gamma^n \textstyle\sum_{b_n \in \Pi(S)} P(b_n|\pi', b_0)\phi(b_n) \\
&\qquad\quad -\gamma^n \textstyle\sum_{b_n \in \Pi(S)} P(b_n|\pi, b_0)\phi(b_n) \\
&\quad = \left[V^{orig}(b_0,\pi') - V^{orig}(b_0,\pi)\right] \\
&\qquad\quad +\gamma^n \textstyle\sum_{b_n \in \Pi(S)} \phi(b_n)[P(b_n|\pi', b_0) - P(b_n|\pi, b_0)] \\
&\quad > \gamma^n \textstyle\sum_{b_n \in \Pi(S)} \phi(b_n)[P(b_n|\pi, b_0) - P(b_n|\pi', b_0)] \\
&\qquad\quad +\gamma^n \textstyle\sum_{b_n \in \Pi(S)} \phi(b_n)[P(b_n|\pi', b_0) - P(b_n|\pi, b_0)] \\
&\quad = 0
\end{aligned}
$$

Thus, π' achieves higher V^{PBRS} than π, so π cannot optimize V^{PBRS} over the finite horizon n. Therefore, planning with PBRS can result in a different policy using a finite horizon. Moreover, provided the potential function guides the agent towards beliefs that earn higher rewards beyond the planning horizon, PBRS could improve upon finite horizon policies that would be found without reward shaping. ■

Furthermore, the impact of the potential function on the valuation of a policy using shaped rewards depends on the size of the planning horizon n. This leads us to the following remark:

> **Remark 3.3:** *The upper bound (Eq. 3.13) on the permissible difference in the valuations of the (finite horizon) optimal policies with and without reward shaping is greater as the finite planning horizon n decreases, making it easier to find a potential function ϕ that satisfies Eq. 3.13 when the planning horizon is small.*

Recall that the discount factor is restricted such that $\gamma \in [0, 1)$. Thus, as n decreases, γ^n increases. Hence, the resulting greater upper bound on the differences between valuations permits a larger number of different policies to optimize each objective function (Eqs. 3.9, 3.13 and Lemma 3.1) over the finite horizon n, so planning

with PBRS is more able to find a different policy than planning without reward shaping *when the horizon is short*. Therefore, provided a suitable potential function, PBRS can be *most beneficial when it is most necessary* (i.e., when planning without PBRS is at greatest risk of being suboptimal (over the infinite horizon) due to short horizons and limited planning time).

Next, we prove that planning with PBRS does not sacrifice optimality over the infinite horizon with respect to the original reward function R, which ultimately the agent wants to maximize. That is, a policy is optimal (without finite horizon approximation) with PBRS if and only if it is also optimal without reward shaping using just the original rewards. Therefore, even though using shaped or unshaped rewards can find different policies for short horizons, using PBRS also optimizes the original reward function R (over the infinite horizon) and is working towards the agent's ultimate goal.

> **Theorem 3.4:** *Let $S, A, \Omega, T, O, R, b_0, \gamma$ from the definition of a POMDP be given, and let ϕ be a potential function over belief states. Then, a policy π^* is optimal (over the infinite horizon) with reward shaping using PBRS if and only if π^* is also optimal (over the infinite horizon) without reward shaping.*

Proof: Let π be any policy. From Lemma 3.1, the value of this policy with PBRS over the infinite horizon is:

$$\begin{aligned} V^{PBRS}(b_0, \pi) &= E[\textstyle\sum_{t=0}^{\infty} \gamma^t r_t^{PBRS}] = \lim_{n\to\infty} E[\sum_{t=0}^{n-1} \gamma^t r_t^{PBRS}] \\ &= \lim_{n\to\infty} \left[V^{orig}(b_0, \pi) + \gamma^n E[\phi(b_n)] - \phi(b_0)\right] \\ &= V^{orig}(b_0, \pi) - \phi(b_0) + \lim_{n\to\infty} \gamma^n E[\phi(b_n)] \\ &= V^{orig}(b_0, \pi) - \phi(b_0) \end{aligned}$$

since $\gamma \in [0,1)$ and thus $\lim_{n\to\infty} \gamma^{n+1} = 0$. Moreover, $\phi(b_0)$ is constant since initial belief state b_0 is fixed. Thus, any policy π^* that optimizes V^{PBRS} over the infinite horizon also optimizes V^{orig}, and vice-versa. Therefore, π^* is optimal over the infinite

horizon with PBRS if and only if it is also optimal over the infinite horizon for the original rewards. ■

From the perspective of finite horizon policies (which the agent is required to calculate to approximate the infinite horizon due to computational constraints), Theorem 3.4 and its proof result in the following important implication:

> **Remark 3.5:** *Planning with PBRS also results in earning greater (unshaped) reward as the planning horizon increases (or equivalently, with more planning time), even though it is optimizing a different objective function than the original reward function.*

Both the proof for Theorem 3.4 and Lemma 3.1 imply that the valuations of policies with and without reward shaping become closer and closer as the planning depth increases. Thus, the policies chosen by each method (with or without reward shaping) also become more similar since these policies maximize their respective valuations. Because approximate planning without reward shaping generally results in better policies as the planning depth increases (since more information is added to the estimation of cumulative, future rewards), this implies that the policies formed with PBRS will also improve with respect to maximizing the original reward function.

Combined with Remark 3.3, this implies that PBRS is beneficial to the agent not only when the planning horizon is small (provided a good potential function), but also as the planning horizon increases (regardless of potential function).

Finally, we derive the following theorem that is important for determining when pre-existing POMDP planning solvers are compatible with PBRS.

> **Theorem 3.6:** *Let $S, A, \Omega, T, O, R, b_0, \gamma$ from the definition of a POMDP be given, and let ϕ be a potential function over belief states. Provided that ϕ is convex, the objective function solved by the agent (Eq. 2.6) remains convex and can be solved by the traditional set of POMDP solvers.*

Proof. Assume that ϕ is indeed convex. Then, Eq. 3.3 is the linear combination of convex functions (Eq. 2.4 is also convex) (Boyd & Vandenberghe, 2004). Thus, the valuation function (Eq. 2.6) remains convex, as proven by Araya-Lopez *et al*. (their Theorem 3.1 in (2010)) (originally established outside the context of PBRS). Therefore, shaped rewards with PBRS can also be optimized by a wide range of POMDP solvers relying on convexity, not just those considered in this chapter. ■

We note here that many of the potential functions provided as examples in this chapter (e.g., Eq. 3.6 and 3.7 above) are indeed convex.

Summary. To summarize our theoretical results, we observe that Lemma 3.1 defines the difference in the evaluation of a policy both (1) with reward shaping using PBRS (V^{PBRS}) and (2) without reward shaping that considers only the original reward function R (V^{Orig}). In turn, Theorem 3.2 provides us with a necessary condition for when a policy would be evaluated as having higher value with PBRS than without. That is, this condition establishes when a different policy might be favored and returned by the planning algorithm, instead of the policy that is optimal—considering only the original reward function—for the small, finite horizon n yet possibly suboptimal over the long run. Remark 3.3 then notes that the condition of Theorem 3.2 is looser for the smallest planning horizons, making it easier for PBRS to favor a different policy that could be closer to optimal over the long run than the small, finite-horizon optimal policy. This should cause us to observe the most impactful benefits on agent performance from PBRS under the tightest time constraints on planning. Theorem 3.4 and Remark 3.5, on the other hand, explores the opposite direction and establishes that as the planning horizon increases, the favored policies found with PBRS also optimize the long term, cumulative

rewards of the agent, which is the agent's ultimate goal. This is true, even though the agent is directly optimizing a slightly different objective function. This should cause us to observe continued good performance from PBRS as planning constraints are relaxed. Finally, Theorem 3.6 establishes that POMDP planning algorithms relying on convexity in the value function to efficiently find optimal policies will also efficiently find optimal policies under PBRS.

Of note, most of these theoretical results exploit the fact that reward shaping under PBRS takes the form of the difference of potential functions (Eqs. 3.3-3.4). Without this difference and instead using arbitrary reward shaping (e.g., simply adding additional value at each node of the policy tree), the telescoping sums would disappear from the proofs. Without the telescoping sums, (1) we would not be able to bound the difference of the evaluation of a policy with and without reward shaping (Theorem 3.2), and we need this bound for Remark 3.3 describing the usefulness of PBRS with small planning horizons, which is important since we are considering time constrained, finite horizon planning that must stop before finding an optimal (infinite horizon) policy, and (2) we could not establish that as the planning horizon increases, the policy optimizing PBRS also optimizes the original reward function, which would in turn affect the ability of planning with PBRS to prefer policies that maximize long term, cumulative rewards.

3.4. Experimental Setup

To evaluate the performance of using PBRS to improve online POMDP planning, we conducted an empirical study that compares agent performance with and without PBRS (using the potential functions summarized in Table 3.2) in three benchmark

POMDP planning problems described below: (1) Tag (Pineau, Gordon, & Thrun, 2003), (2) RockSample (Smith & Simmons, 2004), and (3) AUVNavigation (Ong *et al*., 2010).

These three benchmarks were chosen for our experimental study for the following reasons. First, they are commonly used across the POMDP literature, either together (e.g., Ong *et al*., 2010; Zhang & Chen, 2012) or at least in some combination (e.g., Pineau, Gordon, & Thrun, 2003; Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013). Thus, they are relatively well understood. Second, they represent a varying range of problems: (1) Tag is a relatively small problem (i.e., a low number of states, actions, and observations) with high levels of uncertainty, but a relatively simple required behavior to solve the problem, (2) RockSample is a larger problem than Tag and one for which upper and lower bound estimates provide strong clues on how to behave, and (3) AUVNavigation is an even larger problem (especially with two orders of magnitude larger observation space than Tag or RockSample) with a very high amount of uncertainty and a difficult sequence of behavior required to solve the problem. Thus, they represent very different environments. Moreover, AUVNavigation both: (a) requires a long sequence of information gathering then movement actions to reach the ultimate goal state, and (b) contains dangerous situations that cause the agent to be unable to ever accomplish its goal, both of which were hypothesized in Section 3.1 to be problematic for depth-focused planning algorithms and could benefit from breadth-focused planning with implicit future reward estimations, as accomplished by PBRS for online POMDP planning. We limit our study to considering only three benchmarks for two reasons: (1) much of the POMDP literature considers a similar number of benchmarks (e.g., Ross *et al*., 2008; Somani *et al*., 2013; Zhang & Chen, 2012), and (2) due to the

comprehensiveness of our experimental setup for each benchmark, resulting in much time required to both (i) run the experiments for each benchmark (c.f., the start of Section 3.5) and (ii) implement and test many different potential functions on each benchmark. For comparison and easy reference, we summarize the potential functions considered in each benchmark in Table 3.2.

Table 3.2: Summary of Potential Functions Used in Each Benchmark Problem

Potential Function	Type	Tag	RockSample	AUVNavigation
MBD	Type 1: Domain-Dependent Information from Expected State Potential	Eq. 3.14		
CD			Eq. 3.15	
GD				Eq. 3.17
Entropy	Type 2: Domain-Independent Information	Eq. 3.6		
TopBelief		Eq. 3.7		
NoExit	Type 3: Belief Prioritization		Eq. 3.16	
EMBD	Types 1 + 2: Combination of Domain-Dependent & Domain-Independent Information	Eq. 3.6 + Eq. 3.14		
TBMBD		Eq. 3.7 + Eq. 3.14		
ECD			Eq. 3.6 + Eq. 3.15	
TBCD			Eq. 3.7 + Eq. 3.15	
EGD				Eq. 3.6 + Eq. 3.17
TBGD				Eq. 3.7 + Eq. 3.17
NoExitCD	Types 1 + 3: Combination of Domain-Dependent Information & Belief Prioritization		Eq. 3.15 + Eq. 3.16	
HBGD				Eq. 3.18
NoExitE	Types 2 + 3: Combination of Domain-Independent Information & Belief Prioritization		Eq. 3.6 + Eq. 3.16	
NoExitTB			Eq. 3.7 + Eq. 3.16	
NoExitECD	Types 1 + 2 + 3: Combination of Domain-Dependent & Domain-Independent Information & Belief Prioritization		Eq. 3.6 + Eq. 3.15 + Eq. 3.16	
NoExitTBCD			Eq. 3.7 + Eq. 3.15 + Eq. 3.16	
Upper	Type 4: Approximation of Optimal Value Function	$\overline{V}(b) = V(b, \pi_{FIB})$		
Lower		$\underline{V}(b) = V(b, \pi_{Blind})$		

3.4.1. Benchmark Problems

3.4.1.1. Tag

The first benchmark problem we consider is Tag (Pineau, Gordon, & Thrun, 2003), in which a robotic agent (the tagger) plays laser tag with an opponent. Both agents are randomly placed in a 2D grid consisting of 29 locations and the tagger agent's task is to find and tag the opponent, whereas the opponent tries to prolong the game by moving away from the tagger. Both agents always know their own location and the opponent knows where the tagger is at all times, but the tagger can only observe the opponent when they are in the same cell. The tagger agent earns a penalty of -1 for moving in each cardinal direction (North, South, East, and West) to find its prey, a larger penalty of -10 for trying to tag the opponent without being in the same cell, and a reward of +10 for successfully tagging the opponent, which ends the game. The tagger agent's discounted rewards are maximized by finding and tagging the opponent as fast as possible.

Altogether, Tag represents a relatively small benchmark problem, only consisting of 870 states, 5 actions (movement and tagging), and 2 observations (*True* if the tagger and opponent are in the same cell, else *False*). However, the problem is highly uncertain as the tagger can only identify the opponent's location if they are in the same cell, else it must estimate where the dynamic opponent is as it moves away from the tagger. As such, the distance of the tagger from the end of the game can be quite long and dynamically changes as both agents move through the grid. Therefore, the actual horizon for the problem can be particularly long, and time constrained planning can lead to suboptimal actions.

To improve online, short horizon planning in Tag, we consider seven potential functions representing different domain-independent and -dependent knowledge pointing the agent to future rewards beyond the planning horizon:

- **Entropy**, using a domain-*independent* measure of the certainty in the agent's belief, following Eq. 3.6
- **TopBelief**, using another domain-*independent* measure of the certainty in the agent's belief represented by Eq. 3.7, which is similar to Eq. 3.6, but (1) focuses on certainty in a single state (the most believed state), rather than across the entire belief state and (2) exploits the factored state space (fully observable tagger location vs. partially observable opponent location) to reduce computation
- **MaxBeliefDistance (MBD),** using domain-*dependent* information to assign greater potential to belief states closer to the most likely location of the opponent, thus motivating the agent to move towards the opponent and end the game as fast as possible, hopefully minimizing incurred penalties and maximizing rewards:

$$\phi(b) = \frac{1}{E[d(o,l)]+1} \tag{3.14}$$

where o is a possible opponent location, l is the agent's location, d measures Euclidian distance between o and l, and $E[d(o,l)]$ is the expected distance based on all possible opponent locations in belief state b.

- **EMBD**, which sums Entropy (Eq. 3.6) and MaxBeliefDistance (Eq. 3.14) to combine domain-independent and domain-dependent information in the same potential function
- **TBMBD**, which sums TopBelief (Eq. 3.7) and MaxBeliefDistance (Eq. 3.14) to also combine domain-independent and domain-dependent information in the same potential function
- **Upper**, which uses $\overline{V}(b)$ calculated using π_{FIB} formed using the Fast Informed Bound algorithm (Hauskrecht, 2000) as an approximation of the optimal value function, and
- **Lower**, which uses $\underline{V}(b)$ calculated using π_{Blind} formed using the Blind algorithm (Hauskrecht, 2000) as another approximation of the optimal value function

3.4.1.2. RockSample

The second benchmark problem considered in our experimental setup is RockSample (Smith & Simmons, 2004). In RockSample, an agent navigates a remote world represented by a 2D grid of size $g \times g$ to sample from k rocks. The goal of the agent is to determine which rocks are good, then sample only those rocks. Afterwards, the agent exits by moving to a special location off the grid. To accomplish its goals, the

agent can perform $k + 5$ actions: move in any of the four cardinal directions (North, South, East, West), check the quality at one of each of the k rocks, or sample the rock at its current location. To determine which actions to take, the agent considers a factored state space consisting of: (1) its fully observable current location, and (2) the hidden quality of each rock (from the set $\{Good, Bad\}$). Checking a rock returns a noisy observation about the quality of the rock (also from the set $\{Good, Bad\}$), where the observation's accuracy is greater the closer the agent is to the rock[10]. Sampling a rock changes the state of the rock to Bad (indicating it can no longer be sampled). The agent earns a reward of +10 for sampling a good rock, -10 for sampling a bad rock, and +10 for exiting the grid. All other actions earn zero reward. The agent's discounted rewards are maximized by sampling all (and only) good rocks and exiting as fast as possible.

We use the common setting $g = 7$ and $k = 8$ (e.g., Ross *et al*., 2008; Somani *et al*., 2013; Zhang & Chen, 2012) that results in a POMDP with 12,585 states, 13 actions, and 2 observations. This problem is *larger* than Tag, but *less dynamic*: the problem always ends with the agent reaching the same state (exiting the grid), and the environment does not change as the agent moves around. Thus, it presents a different set of challenges for time constrained planning, including a *broader* search tree (due to more possible actions) and *deeper* required activity to accomplish all the agent's goals (sampling as many good rocks as exist in the environment), but identifying the goal state is *less* challenging, making it easier to achieve goal directed behavior.

[10] To increase the complexity of the RockSample benchmark and make it more suitable for our experimental study by making it a little more uncertain like the other benchmark problems considered in this research, we increased the uncertainty in the observations returned when checking rocks by decreasing the half-efficiency distance of sensing from 20 to 1. This is similar to changes made in other experimental studies, including the similar FieldVisionRockSample considered in (Ross *et al*., 2008; Zhang & Chen, 2012).

To improve online planning in RockSample, we consider 13 potential functions representing different domain-independent and -dependent knowledge pointing the agent to future rewards beyond the planning horizon. Some are reused from Tag (Entropy, TopBelief, Upper, and Lower), whereas others are unique to RockSample:

- **ClosestDistance (CD),** using domain-*dependent* information to assign greater potential to belief states closer to uncertain rocks where the agent will achieve greater accuracy and thus most immediate belief improvement:

$$\phi(b) = \begin{cases} -\frac{1}{2g}\min_{r \in R}[d(r,l)+1] & \text{if } R \neq \emptyset \\ 0 & \text{if } R = \emptyset \end{cases} \quad (3.15)$$

where $R = \{r \mid 0.01 < b(r) < 0.99\}$ is the set of rocks with uncertain quality, l is the agent's location, and d measures Euclidian distance between r and l.

- **NoExit**, prioritizing beliefs reflecting more certain knowledge about rocks before exiting to avoid neglected sampling due to myopic planning (similar to Eq. 3.8 example from Section 3.3.1):

$$\phi(b) = \begin{cases} -1000 & \text{if } R \neq \emptyset \wedge l = \text{exit} \\ 0 & \text{else} \end{cases} \quad (3.16)$$

- **ECD**, which sums Entropy (Eq. 3.6) and ClosestDistance (Eq. 3.15) to combine domain-independent and domain-dependent information in the same potential function

- **TBCD**, which sums TopBelief (Eq. 3.7) and ClosestDistance (Eq. 3.15) to also combine domain-independent and domain-dependent information in the same potential function

- **NoExitE**, which sums Entropy (Eq. 3.6) and NoExit (Eq. 3.16) to combine domain-independent information and belief prioritization in the same potential function

- **NoExitTB**, which sums TopBelief (Eq. 3.7) and NoExit (Eq. 3.16) to also combine domain-independent information and belief prioritization in the same potential function

- **NoExitCD**, which sums ClosestDistance (Eq. 3.15) and NoExit (Eq. 3.16) to combine domain-dependent information and belief prioritization in the same potential function

- **NoExitECD**, which sums Entropy (Eq. 3.6), ClosestDistance (Eq. 3.15), and NoExit (Eq. 3.16) to combine domain-independent and domain-dependent information, as well as belief prioritization, in the same potential function

- **NoExitTBCD**, which sums TopBelief (Eq. 3.7), ClosestDistance (Eq. 3.15), and NoExit (Eq. 3.16) to also combine domain-independent and domain-dependent information, as well as belief prioritization, in the same potential function

3.4.1.3. AUVNavigation

The final benchmark problem considered in our experimental setup is AUVNavigation (Ong *et al*., 2010). In AUVNavigation, a robotic submarine agent is randomly placed on one side of a $20 \times 7 \times 4$ 3D underwater grid and must navigate through a set of rock obstacles to either of two known goal locations on the other side of the grid. The agent can Stay in its current position, turn Left, Right, Up, or Down to change its orientation, or it can move Forward along its orientation towards a desired location. Currents underwater also move the agent with low probability, resulting in stochastic location changes, whether or not the agent intended to move. The agent has sensors that always perfectly observe the agent's depth and orientation in the grid, but its location in the 2D plane is uncertain. Thus, navigating through the rocks to reach the goal is quite challenging. The agent can move to the surface of the water where it automatically uses a GPS sensor to perfectly determine its location, but this incurs a moderate cost of -50. Otherwise moving through the grid incurs a penalty of -1, -1.44, or -1.73, depending on its orientation (with higher cost for moving diagonally and changing depths in the grid), whereas Staying or changing orientation earns zero reward. The agent incurs a large penalty of -500 for hitting a rock and an even larger reward of +5000 for reaching a goal location, each of which result in a terminal state that ends execution. The agent's discounted rewards are maximized by reaching the goal location as fast as possible while minimizing costs incurred for spending time on the surface.

Altogether, AUVNavigation represents a *very* challenging benchmark problem compared to the other two benchmarks. Whereas the number of states and actions (13,537 and 6, respectively) in this problem is similar to RockSample, the number of

observations (144) is *much* greater, increasing the size of the POMDP and the breadth of the planning tree, and the uncertainty is also *much* greater due to the lack of full observability of the agent's location. Thus, AUVNavigation is the largest and most complex benchmark considered in our experiments. Due to this uncertainty and complexity, AUVNavigation can be viewed as containing three sub-problems in three stages: (1) determining the agent's location on the far side of the grid, (2) navigating through the many dangerous rock obstacles (requiring high certainty in the agent's location), and (3) finding a path beyond the obstacles to one of the goal locations. Furthermore, the actual horizon for this problem is *quite long* and requires more memory than an agent can afford for full breadth planning (due to exponential growth in the planning tree), requiring over 20 actions just to move the agent from its initial location to a goal location without accounting for the number of actions required to resolve its initial location uncertainty. Since a positive reward signaling a good planning path to the agent *only* occurs when it reaches the goal (after at least 20 steps), time constrained planning is very difficult in this domain since there are no intermediate positive signals to guide the agent towards the goal state. As a result, PBRS is possibly a beneficial approach for this benchmark problem since potential functions can provide such intermediate positive signals, but the potential functions need to be able to account for the *different stages* of the problem to successfully guide the agent towards its goal, which could require more complex potential functions than the other two benchmark problems.

To improve online planning in AUVNavigation, we consider eight potential functions representing different domain-independent and -dependent knowledge pointing the agent to future rewards beyond the planning horizon. Some are reused from Tag and

RockSample (Entropy, TopBelief, Upper, and Lower), whereas others are unique to AUVNavigation:

- **GoalDistance (GD),** using domain-*dependent* information to assign greater potential to belief states closer to the nearest of the two goal locations where the agent has less distance to travel (and further movement cost to incur) to reach its goal:

$$\phi(b) = \frac{1}{E[d(g,l)]+1} \tag{3.17}$$

where l is a possible agent location, g is the nearest goal location to l, d measures Euclidian distance between l and g (equal to the maximum possible distance if l is also a rock location to encourage the agent to avoid rocks), and $E[d(g,l)]$ is the expected distance based on all possible agent locations in belief state b.

- **EGD**, which sums Entropy (Eq. 3.6) and GoalDistance (Eq. 3.17) to combine domain-independent and domain-dependent information in the same potential function

- **TBCD**, which sums TopBelief (Eq. 3.7) and GoalDistance (Eq. 3.17) to also combine domain-independent and domain-dependent information in the same potential function

- **HighBeliefGoalDistance (HBGD)**, which combines prioritizing beliefs containing high certainty in a single state, reflecting more certain knowledge about the agent's current location, and the domain-dependent information GoalDistance potential function (Eq. 3.17) to help the navigate towards a goal location after resolving its own location uncertainty:

$$\phi(b) = \begin{cases} \frac{1}{E[d(g,l)]+1} & \text{if } \max_{s \in S} b(s) > 0.6 \\ 0 & \text{else} \end{cases} \tag{3.18}$$

3.5. Results

In this section, we analyze the results of our experiments using the benchmark problems and potential functions outlined in the previous section and evaluate the empirical performance of using PBRS to improve online POMDP planning.

Specifically, we evaluate performance by comparing the (infinite horizon) cumulative, discounted rewards earned by the agent while operating in each benchmark:

$$\sum_{t=0}^{\infty} \gamma^t r_t^{orig} \tag{3.19}$$

since this is the function the agent intends to optimize (even if it must rely on finite horizon approximations during planning) and is the traditional measure for evaluating POMDP planning. Please note that this measurement does *not* include the additional

rewards from any potential function in order to provide a fair comparison between approaches with and without reward shaping.

For PBRS, we performed full breadth planning using a randomized BFS expansion of the planning tree using different amounts of time τ for online planning representing different time constraints imposed on the agent's reasoning[11] (common to real-world environments): $\tau \in \{5, 10, 50, 100\}$ milliseconds for Tag and RockSample and $\tau \in \{100, 500, 1000, 5000\}$ milliseconds for the larger and more complex AUVNavigation.

Within each benchmark, we compared for each amount of allotted time τ the performance of planning (1) without reward shaping (**Original**), (2) with reward shaping using different potential functions for each benchmark problem (summarized in Table 3.2 and described above), (3) using AEMS2 (Ross & Chaib-draa, 2007), a state-of-the-art heuristic search algorithm, and (4) using ABDESPOT and ARDESPOT, two online variants of a state-of-the-art Monte Carlo tree search algorithm called DESPOT (Somani *et al*., 2013). Any *offline* planning required by the algorithms is not included in τ.

Our results were averaged over 1000 runs of each problem for each planning approach and allotted time combination (except for AUVNavigation, where we only employed 100 runs due to its higher range of τ values). To speed up computation in each benchmark, we used the state-of-the-art equivalent MOMDP[12] representation (Ong *et al*., 2010) for the POMDP model, as also done in the recent online POMDP planning

[11] We use a different range of allotted times τ for different problems due to the different sizes of the POMDPs, resulting in different exponential growth of the planning trees calculated by the agents.

[12] A mixed observability MDP (MOMDP) is a special POMDP representation that factors the state space into fully observable variables $\mathcal{X}$ and partially observable variables $\mathcal{Y}$, such that $S = \mathcal{X} \times \mathcal{Y}$, and exploits this factorization to simplify the transition and observation probability calculations to speed up computation. The resulting model is equivalent (but faster) to the canonical, unfactored POMDP representation for the same problem (Ong *et al*., 2010).

literature (e.g., Zhang & Chen, 2012). We limited each run to 200 time steps, which should be ample time for the agent to solve each problem (else the agent was acting randomly and not in a goal directed fashion, and thus would probably never accomplish its goal if left to run longer).

Because we limited planning to fixed amounts of time, all experiments per benchmark were conducted on a fixed computer to avoid introducing variance into the results due to differences between computers, instead of due to differences in the algorithms' performances that we intended to measure. Two computers were chosen for this purpose: each possessing an Intel i5 (Haswell) 3.4 GHz Quad Core processor with 8GB of RAM (limited to one thread and 3 GB of RAM per experiment run). One computer ran all of the Tag and RockSample experiments, while the other ran the lengthier AUVNavigation experiments.

In the following, we analyze performance in each of the benchmarks separately: first Tag, then RockSample, and finally AUVNavigation. Afterwards, we provide discussions generalizing our results across benchmarks to provide a more abstract identification of the strengths and weaknesses of each approach to online planning, especially focusing on using PBRS.

For each problem, we first compare the performance of full breadth planning with PBRS using the different potential functions against Original (i.e., full breadth planning without reward shaping) to explore whether or not the different types of potential functions truly provide implicit clues of what actions the agent should take to earn large cumulative, future rewards beyond the agent's planning horizon. Second, we compare the performances of each type of potential function to try to gain insights into which

Table 3.3: Results from Tag Benchmark Problem with 95% Confidence Intervals

Approach	τ 5 ms	10ms	50ms	100ms
Original	-12.84 ± 0.55	-10.09 ± 0.54	-9.13 ± 0.51	-9.68 ± 0.50
Entropy	-12.47 ± 0.56	-10.64 ± 0.54	-11.41 ± 0.54	-9.57 ± 0.51
TopBelief	-13.60 ± 0.54	-10.37 ± 0.57	-9.79 ± 0.54	-10.07 ± 0.51
MBD	-9.73 ± 0.47	-7.77 ± 0.45	-7.77 ± 0.44	-7.23 ± 0.42
EMBD	-9.33 ± 0.44	-7.98 ± 0.45	-8.89 ± 0.46	-7.08 ± 0.42
TBMBD	-9.46 ± 0.47	-7.09 ± 0.42	-7.66 ± 0.43	-7.16 ± 0.39
Upper	-8.79 ± 0.44	-7.30 ± 0.41	-7.52 ± 0.40	-6.20 ± 0.39
Lower	-13.99 ± 0.55	-10.21 ± 0.51	-10.21 ± 0.52	-12.32 ± 0.54
AEMS2	-6.40 ± 0.40	-5.65 ± 0.40	-5.75 ± 0.38	-5.78 ± 0.38
ABDESPOT	-15.54 ± 0.41	-12.16 ± 0.42	-7.36 ± 0.38	-6.57 ± 0.39
ARDESPOT	-14.94 ± 0.43	-12.36 ± 0.41	-7.03 ± 0.38	-6.61 ± 0.37

might be most advantageous to improve agent planning. Finally, we compare the performances of the best and worst potential functions (and Original) against the three depth-focused state-of-the-art online POMDP planning algorithms in order to determine how well our proposed approach compares to the best known approaches and to see what benefits we gain from maintaining full breadth planning with implicit estimations of future rewards.

3.5.1. Tag Results

3.5.1.1. Comparison of Full Breadth Planning With and Without Reward Shaping

We begin our results analysis by comparing the performance of full breadth planning with (PBRS) and without (Original) reward shaping on the Tag benchmark problem to discover the benefits of implicitly estimating future rewards without explicit calculations. We present in Table 3.3 the cumulative, discounted reward results earned by the agent on this benchmark for each solution.

From these results, we make several important observations. First, the majority of the potential functions resulted in *improved performance* across the various planning horizons when compared to breadth-first planning without reward shaping (Original): 18

of 28 (64.3%) potential function and time constraint pairs yielded higher cumulative reward in Tag. Indeed, several of the potential functions (MBD, EMBD, TBMBD, and Upper) even achieved quite significant improvements over full breadth planning with no reward shaping: improvements of 31.5%, 29.7%, 17.6%, and 36.0% in cumulative reward across the four different time constraints for planning ($\tau = 5, 10, 50, 100$ ms), respectively. Moreover, the best potential functions (MBD, EMBD, TBMBD, Upper) led to better performance with only 10 ms of planning time, compared with employing an order of magnitude more time for planning (up to 100 ms) with Original. Thus, reward shaping can yield improved performance while using even less planning time.

Overall, we conclude from these results that using PBRS to shape rewards with potential functions often *resulted in better planning and subsequent performance by the agent through considering implicit estimates of future rewards, as intended*. So, we have evidence that using potential functions is a good approach for improving the quality of plans formed during full breadth planning.

However, not every potential function achieved better performance than Original. Namely, the Entropy, TopBelief, and Lower potential functions achieved worse (or similar) performance on many of the time constraints used for planning. Thus, we have evidence that not every potential function (or indicator of future rewards) is beneficial to planning, and care must be taken when choosing an appropriate potential function for the agent's problem. In the next subsection, we will investigate further why these potential functions might have been a bad choice on Tag, and we will provide a more general discussion on this topic in Section 3.5.4.

3.5.1.2. Comparison Between Potential Function Types

Next, we try to better understand the differences between the performances resulting from each of the potential function types on the Tag benchmark problem. From the results in Table 3.3, we observe that the domain-dependent information (from expected state-based potential functions, Type 1) (MBD) generally outperformed the domain-independent information (from measures of the quality of agent knowledge, Type 2) potential functions (Entropy, TopBelief) independently. Considering the fact that Type 1 potential functions on POMDPs are a direct extension of the type of potential functions used elsewhere in the literature, we find that utilizing this extension is in fact still beneficial in POMDPs. On the other hand, *combining the two types* (Type 1 and 2 in the EMBD and TBMBD potential functions) *generally resulted in better performance than either type alone*. Therefore, we observe an *added benefit of considering different types of potential functions*, including those novel to POMDPs and proposed in this research (Type 2). In other words, the types of information provided by both form a *stronger indicator or estimator of cumulative, future rewards* the agent will earn from the belief states with higher potential under these functions.

The approximations of the optimal value function (Type 4 potential functions, commonly used in leaf evaluation heuristics), on the other hand, provided mixed results. On the one hand, the Upper bound approximation (from FIB (Hauskrecht, 2000)) outperformed Original and was the best potential function overall with the greatest performance amongst potential functions for three of the four planning times considered ($\tau = 5, 50, 100$ ms). On the other hand, the Lower bound approximation (from Blind (Hauskrecht, 2000)) was one of the worst performers of all potential functions, regardless

of the amount of planning time allotted. Thus, this particular potential function (commonly used in practice as a leaf evaluation heuristic (e.g., Ross *et al*., 2008) is possibly not as good of a choice as other types of information for guiding agent action selection, at least on the Tag benchmark.

3.5.1.3. Comparison of PBRS with Depth-Focused, State-of-the-Art Planning Algorithms

Now, we compare full-breadth planning with and without PBRS against the three state-of-the-art algorithms—AEMS2 heuristic search, as well as ABDESPOT and ARDESPOT MCTS algorithms. Our goal is to determine whether maintaining full breadth planning with implicit estimations of future rewards is beneficial in comparison to depth-focused approaches that explicitly calculate the cumulative, future rewards the agent intends to maximize. For this analysis, we plot in Figure 3.1 the performance as planning time increased for the best (Upper) and worst (Lower) potential functions, as well as Original and the state-of-the-art algorithms.

From these results, we first observe that full breadth planning (with and without reward shaping) was advantageous for the smallest amounts of planning time ($\tau = 5, 10$ ms) in comparison to the MCTS algorithms. This was due to the depth-focused MCTS algorithms not having enough time to find a path of actions to the agent's goal using biased random sampling (and thus suffered from the problems of sacrificing breadth without gaining the benefits of focusing on depth during planning). In fact, for these amounts of planning time, the MCTS algorithms had the worst overall planning performance on this benchmark (as seen in Table 3.3).

Moreover, as planning time increased, the best potential function (Upper) remained competitive with the MCTS algorithms as their performance increased (for

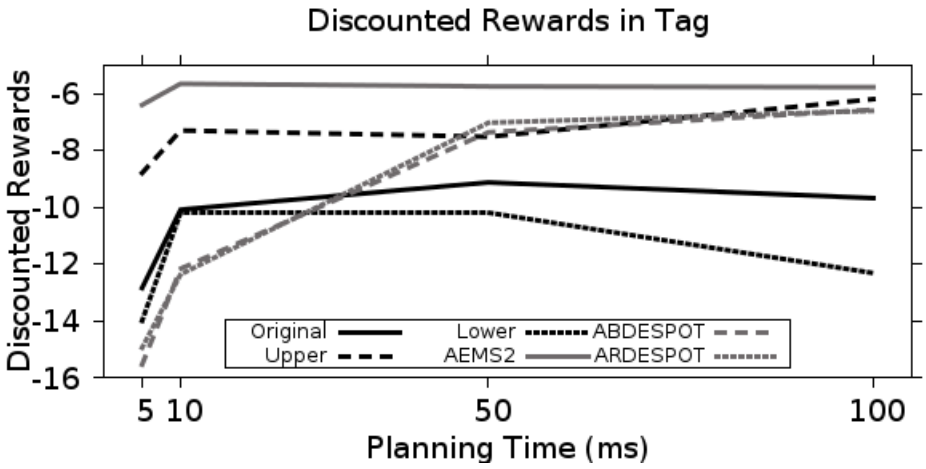

Figure 3.1: Performance of Planning Algorithms as Planning Time Increased on the Tag Benchmark Problem for Select Approaches

MCTS, due to better depth-focused planning with more planning time). These results imply that maintaining breadth-focused planning enhanced by implicit estimates of large future rewards achieved close performance to good explicit estimates of cumulative, future rewards. Therefore, implicit estimates can be as useful in at least some domains (like Tag) as explicitly calculating those rewards (under limited time constraints for planning[13]).

However, the best state-of-the-art algorithm (AEMS2 heuristic search) outperformed the best potential function (Upper). Here the PBRS performance was not quite as good, indicating for the Tag benchmark, depth-focused planning providing explicit cumulative, reward estimates was still the best approach for planning. That is, the heuristic used by AEMS2 (based on error bounds in Upper and Lower bounds in agent rewards and optimistically biased towards Upper bound rewards) indeed selected appropriate belief states to expand during planning. Therefore, implicit future reward

[13] Without time constraints, explicit calculations would always be superior because the agent could simply continue planning deeper throughout the entire planning tree. But with time constraints, the agent must of course sacrifice some breadth for depth, causing under- or over-estimations of agent rewards for some belief states, as discussed in Section 3.2.2.

Table 3.4: Results from RockSample Benchmark Problem with 95% Confidence Intervals

Approach	τ			
	5 ms	**10 ms**	**50 ms**	**100 ms**
Original	7.66 ± 0.30	9.19 ± 0.33	11.60 ± 0.35	12.47 ± 0.36
Entropy	4.35 ± 0.35	7.07 ± 0.36	10.23 ± 0.33	11.62 ± 0.35
TopBelief	8.11 ± 0.31	9.46 ± 0.33	11.68 ± 0.34	12.46 ± 0.35
CD	10.91 ± 0.33	11.45 ± 0.33	12.14 ± 0.34	12.19 ± 0.34
ECD	10.75 ± 0.49	12.02 ± 0.46	12.78 ± 0.37	13.91 ± 0.37
TBCD	10.71 ± 0.32	11.62 ± 0.33	11.98 ± 0.34	12.24 ± 0.34
NoExit	7.28± 0.30	8.41 ± 0.32	10.95 ± 0.38	11.82 ± 0.38
NoExitE	4.09 ± 0.33	6.29 ± 0.34	9.76 ± 0.34	11.16 ± 0.36
NoExitTB	7.97 ± 0.31	9.83 ± 0.35	12.74 ± 0.39	13.97 ± 0.40
NoExitCD	11.69 ± 0.36	12.16 ± 0.35	13.05 ± 0.37	13.47 ± 0.37
NoExitECD	11.16 ± 0.54	13.76 ± 0.50	14.57 ± 0.39	16.08 ± 0.40
NoExitTBCD	11.97 ± 0.36	12.73 ± 0.35	13.92 ± 0.38	14.13 ± 0.38
Upper	11.24 ± 0.36	11.16 ± 0.34	8.41 ± 0.31	16.38 ± 0.41
Lower	7.63 ± 0.09	8.15 ± 0.16	12.09 ± 0.31	14.31 ± 0.33
AEMS2	8.35 ± 0.17	14.07 ± 0.33	15.45 ± 0.35	16.41 ± 0.37
ABDESPOT	14.63 ± 0.35	14.71 ± 0.36	13.36 ± 0.39	16.13 ± 0.44
ARDESPOT	14.53 ± 0.18	14.71 ± 0.18	14.22 ± 0.20	16.50 ± 0.21

estimations are not always as good as explicit calculations, even with limited time constraints and having to sacrifice breadth to achieve such depth during planning.

3.5.2. RockSample Results

3.5.2.1. Comparison of Full Breadth Planning With and Without Reward Shaping

We continue our results analysis by comparing the performance of full breadth planning with (PBRS) and without (Original) reward shaping on the RockSample benchmark problem so that we can gain additional insights into the benefits of implicitly estimating future rewards without explicit calculations. We present in Table 3.4 the cumulative, discounted reward results earned by the agent on this benchmark for each solution.

As in the Tag benchmark problem, we again observe that many of the potential functions resulted in *improved performance* across the various time constraints on planning when compared to full breadth planning without reward shaping (Original): 34

of 52 (65.4%) potential function and time constraint pairs yielded higher reward in RockSample. Therefore, we have additional evidence that implicit estimators of cumulative, future rewards can improve full breadth planning.

Interestingly, the majority of these improved performances occurred for the three smallest amounts of time allotted for planning ($\tau = 5, 10, 50$ ms) where 27 of 39 (69.2%) potential function and time constraint pairs yielded higher cumulative reward than Original. This observation supports Remark 3.3 (c.f., Section 3.3.2) that *PBRS can be most beneficial when the amount of time allowed for planning is smallest*.

For the largest amount of planning time, on the other hand, less than half of the potential functions (ECD, NoExitTB, NoExitCD, NoExitECD, NoExitTBCD, Lower) outperformed Original. This again indicates that planning with PBRS is not beneficial with any potential function and can be less useful as time constraints are reduced (i.e., there is more time for planning and less need for implicit estimators of rewards beyond the planning horizon).

3.5.2.2. Comparison Between Potential Function Types

Comparing between potential function types, we make many of the same observations for the RockSample as we did for the Tag benchmark in Section 3.5.1.2: domain-dependent information (Type 1, CD) potential functions generally outperformed domain-independent information (Type 2, Entropy and TopBelief) individually. Indeed, the Entropy potential function yielded some of the worst performances amongst all approaches used in our experimental study. Upon further investigation, this was due to this potential function leading the agent to overly conservative behavior by sensing too frequently to reach overly high confidence values before sampling rocks, resulting in less

efficient behavior than the other approaches. However, together potential function Types 1 and 2 (especially ECD) perform better than either member type alone. Again, this demonstrates the advantages of exploiting information only available in POMDPs (Type 2 potential functions), and not in fully observable settings, as previously studied.

Furthermore, we also observe that our other proposed novel type of potential function—belief prioritization (Type 3)—also does not perform as well on its own as some of the other types, but combining Types 1, 2, and 3 yielded the best performance amongst all potential function types. In particular, planning with the NoExitECD potential function had the best performance amongst all potential functions. Thus, like Type 2, this third type of potential function (also novel to POMDPs and introduced by this research) is a beneficial form of metareasoning for the agent within a POMDP planning framework, but requires other types of information (especially domain-specific information measured in Type 1 potential functions) to best improve agent planning.

Finally, as in the Tag benchmark problems, the approximations of the optimal value function (Type 4, commonly used as leaf evaluation heuristics) provided mixed results. Whereas the Upper bound (calculated using FIB (Hauskrecht, 2000)) again generally provided improved behavior, the Lower bound potential function also led to lower performance than planning without reward shaping (Original) for the lowest time constraints on planning ($\tau = 5, 10$ ms). Thus, potential functions of the type commonly used for leaf evaluation heuristics still provided some benefit on this problem, but was less beneficial overall than other potential function types providing other indicators of which belief states yield high cumulative, future rewards.

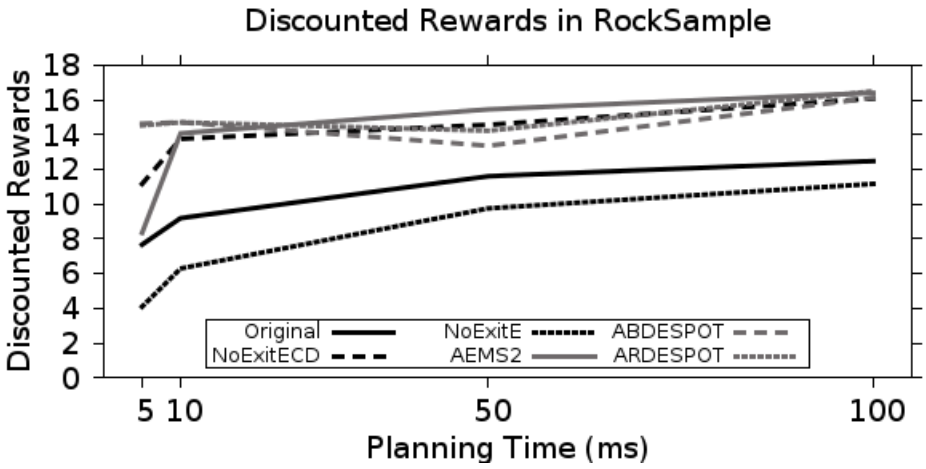

Figure 3.2: Performance of Planning Algorithms as Planning Time Increased on the RockSample Benchmark Problem for Select Approaches

3.5.2.3. Comparison of PBRS with Depth-Focused, State-of-the-Art Planning Algorithms

To better understand the relative performance of PBRS performing full breadth planning with implicit estimation of cumulative, future rewards against depth-focused state-of-the-art algorithms on the RockSample benchmark problem, we plot in Figure 3.2 the performance as planning time increased for the best (NoExitECD) and worst (NoExitE) potential functions, as well as Original and the state-of-the-art online POMDP planning algorithms.

From these results, we observe that for each planning time, full-breadth planning with the NoExitECD potential function performed favorably to the three state-of-the-art, depth-focused planning algorithms. Namely, NoExitECD outperformed the state-of-the-art heuristic search algorithm AEMS2 for the most constrained amount of planning time ($\tau = 5$ ms) and the state-of-the-art Monte Carlo search DESPOT algorithms as planning time increased ($\tau = 50$ ms), and was comparable to the state-of-the-art algorithms for the other planning times. This is a very interesting result because unlike in the Tag benchmark problem, Table 3.4 shows that in RockSample all of the depth-focused

approaches—the heuristic search algorithm (AEMS2) and the MCTS algorithms (ABDESPOT, ARDESPOT)—generally outperformed full-breadth planning (especially compared to Original), even for the lowest amounts of planning time. Thus, in this particular problem, depth-focused planning appears to generally be a better approach than full-breadth planning. However, the indicators of future rewards measured by NoExitECD (combining both a Type 1 potential function as commonly used elsewhere in the PBRS literature, as well as our novel Type 2 and 3 potential functions exploiting metareasoning about agent knowledge and histories) sometimes *led the agent to select better actions using* implicit *estimates of cumulative, future rewards instead of spending time* explicitly *calculating such rewards with depth-focused planning*. Combined with the Tag benchmark results, this is additional evidence that using the novel types of potential functions for planning is very advantageous for improving agent performance in partially observable environments.

3.5.3. AUVNavigation Results

3.5.3.1. Comparison of Full Breadth Planning With and Without Reward Shaping

Finally, we evaluate the results from the most complicated AUVNavigation benchmark, where time constrained planning is generally very difficult without some estimations of future rewards along very deep planning paths due to the long sequence of actions required to reach the goal state (which is the only state to provide positive reward to guide planning). As before, we begin our analysis of the results from this benchmark by comparing the performance of full breadth planning with (PBRS) and without (Original) reward shaping to evaluate the benefits of implicitly estimating future rewards

Table 3.5: Results from AUVNavigation Benchmark Problem with 95% Confidence Intervals

Approach	τ 100 ms	500 ms	1000 ms	5000 ms
Original	-7.41 ± 7.69	-6.63 ± 7.59	-5.19 ± 6.81	-5.02 ± 6.75
Entropy	-76.41 ± 41.76	-598.67 ± 38.01	-549.19 ± 26.55	-262.52 ± 26.80
TopBelief	-511.67 ± 49.07	-609.74 ± 22.39	-525.84 ± 25.62	-291.15 ± 66.50
GD	0.81 ± 18.00	359.86 ± 77.31	366.83 ± 82.00	480.05 ± 101.04
EGD	-218.20 ± 77.21	-35.91 ± 78.78	-40.80 ± 67.10	18.10 ± 101.97
TBGD	-671.25 ± 16.43	-602.84 ± 29.89	-505.37 ± 33.08	-580.47 ± 49.34
HBGD	63.53 ± 109.08	552.01 ± 92.95	542.61 ± 76.10	443.69 ± 96.62
Upper	-15.16 ± 9.25	-16.23 ± 9.24	163.77 ± 70.62	156.81 ± 74.42
Lower	-4.70 ± 6.59	-4.72 ± 6.59	-4.75 ± 6.59	-2.43 ± 1.77
AEMS2	-4.71 ± 6.59	-4.69 ± 6.59	-1.42 ± 0.65	-4.42 ± 6.56
ABDESPOT	305.69 ± 107.45	458.08 ± 110.50	323.94 ± 86.37	391.94 ± 80.89
ARDESPOT	32.81 ± 40.97	57.82 ± 40.19	82.30 ± 42.89	403.04 ± 80.78

without explicit calculations. We present in Table 3.5 the cumulative, discounted reward results earned by the agent on this benchmark for each solution.

In AUVNavigation, we observe far different results than in the simpler Tag and RockSample benchmarks. At first glance, PBRS often appears to have resulted in *worse* performance than planning without reward shaping (Original): 17 of 32 (53.1%) of the potential function and time allocation pairs resulted in worse performance than planning without reward shaping.

However, upon deeper investigation, these results are a consequence of an interesting quirk in the reward function optimized by the agent, rather than truly worse performance when using PBRS. In particular, recall that the agent received zero penalty for either doing nothing with the Stay action or for changing its orientation (using the Up, Down, Left, and Right actions). Otherwise, the agent received a small penalty for moving using the Forward action. Thus, for time constrained full breadth planning without PBRS, the agent rarely calculated any benefit to moving Forward and instead chose actions that yielded zero reward (and thus no cost). As a result, the agent without PBRS never reached the goal location and sat aimlessly, sometimes eventually drifting

into a rock (due to the dynamic currents underwater), resulting in a penalty of -500. Thus, the cumulative, discounted rewards earned by the agent without PBRS were close to 0 (any penalty of -500 occurred after many steps and was heavily discounted) and identical across all amounts of time allowed for planning. Therefore, planning without PBRS resulted in random, uneventful behavior (stuck in Stage 1 of the problem, c.f. Section 3.4.1.3) and *not goal-directed behavior*, as necessary (c.f., Section 3.4.1.3).

On the other hand, for the agents with potential functions using PBRS, the agent received incentive for moving Forward from its shaped rewards, thereby incurring negative costs for movement. As a result, the agent usually achieved worse cumulative, discounted rewards, but *more goal-directed behavior*. In particular, the potential functions combining domain-dependent and domain-independent information (EGD, TBGD) chose actions that successfully completed Stage 1 (uncertainty reduction) and Stage 2 (navigating through the rock obstacles) of the problem, but incurred large costs (-50 per step) by moving along the surface of the water, where the agent always updated its location with perfect accuracy. Thus, including potential functions resulted in better behavior towards goal accomplishment than full breadth planning without reward shaping (Original), due to supplying required intermediate positive signals that allowed the agent to find a plan within time constrained planning that lead the agent towards the goal state.

To better evaluate goal achievement in the challenging AUVNavigation benchmark problem, we present in Table 3.6 the proportion of the 100 runs in which the agent successfully reached a goal location. From these results, we observe that *planning with PBRS was much more successful*: 18 of 32 (56.3%) of the potential function and horizon pairs resulted in more goal achievement than planning without PBRS (Original),

Table 3.6: Proportion of AUVNavigation Runs Successfully Ending at a Goal Location with 95% Confidence Intervals

	τ			
Approach	**100 ms**	**500 ms**	**1000 ms**	**5000 ms**
Original	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Entropy	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
TopBelief	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GD	0.01 ± 0.02	0.82 ± 0.08	0.89 ± 0.06	0.88 ± 0.06
EGD	0.78 ± 0.08	0.88 ± 0.06	0.86 ± 0.07	0.87 ± 0.07
TBGD	0.01 ± 0.02	0.20 ± 0.08	0.18 ± 0.08	0.25 ± 0.09
HBGD	0.75 ± 0.09	0.87 ± 0.07	0.92 ± 0.05	0.90 ± 0.06
Upper	0.00 ± 0.00	0.00 ± 0.00	0.33 ± 0.09	0.33 ± 0.09
Lower	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
AEMS2	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
ABDESPOT	0.49 ± 0.10	0.65 ± 0.09	0.59 ± 0.10	0.76 ± 0.08
ARDESPOT	0.30 ± 0.09	0.49 ± 0.10	0.57 ± 0.10	0.75 ± 0.09

whereas PBRS never performed worse, regardless of the potential function used. Thus, we also find evidence in very complicated environments that potential functions can produce improved planning in a full breadth scenario using implicit estimations of cumulative, future rewards.

3.5.3.2. Comparison Between Potential Function Types

In particular, potential functions combining domain-dependent location information (for rock obstacle avoidance and movement towards the goal in Stages 2 and 3 using Type 1 potential function information) with either domain-independent information (for encouraging belief improvement in Stage 1 using Type 2 potential function information) (EGD, TBGD) or belief prioritization (also prioritizing belief improvement in Stage 1 using Type 3 potential function information) (HBGD) achieved much better performance than planning without PBRS. Domain-dependent location information (Type 1) also performed very favorably to planning without PBRS, although not quite as well as adding metareasoning by combining Type 1 with Type 2 or Type 3 potential functions. Overall, this level of performance is quite significant since

successful time constrained planning is generally incredibly difficult for such a complex problem!

Moreover, for each successful potential function, performance often increased as the planning horizon increased, with HBGD eventually achieving the goal in nearly all (92%) runs. Therefore, planning with PBRS was also very beneficial in AUVNavigation, and was able to guide the agent to goal achievement even with time constrained planning in a very complex domain – containing multiple stages with different objectives and long sequences of actions required to reach the goal state – so long as the potential function considered adequate information to guide the agent through the complex domain (here, combinations of information about domain-dependent location and domain-independent certainty or belief prioritization).

Interestingly, potential functions based on approximations of the optimal value function (Upper, Lower) were not as beneficial in this domain (although Upper did improve performance for the two largest amounts of planning time considered, $\tau = 1000, 5000\ ms$). This is a direct consequence of the complexity of the domain, causing the upper and lower bounds on the value function $\overline{V}(b)$ and $\underline{V}(b)$ from Fast Informed Bound and Blind (Hauskrecht, 2000) to be quite loose (ranging from over 2000 to less than 0 for most belief states), not helping agent performance (as previously observed in Tag).

3.5.3.3. Comparison of PBRS with Depth-Focused, State-of-the-Art Planning Algorithms

As a final analysis, in order to better understand the relative performance of full breadth planning with PBRS on the AUVNavigation benchmark problem against depth-focused state-of-the-art approaches, we plot in Figure 3.3 the performance as planning

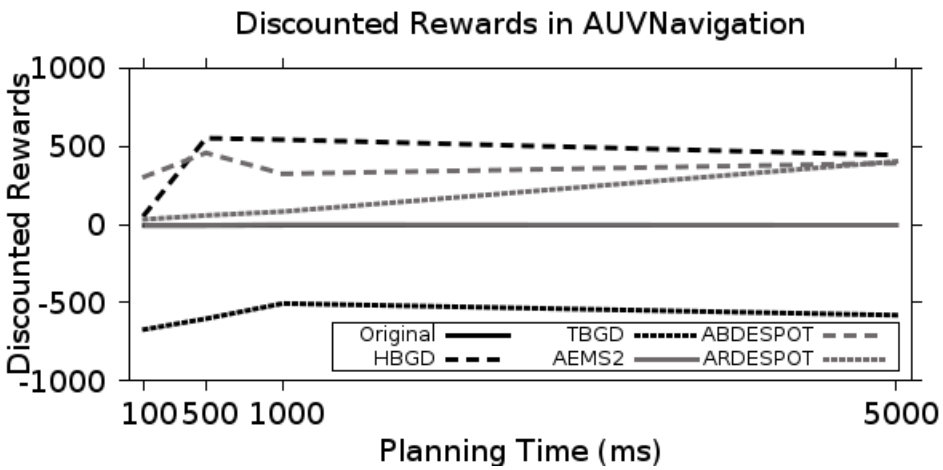

Figure 3.3: Performance of Planning Algorithms as Planning Time Increased on the AUVNavigation Benchmark Problem for Select Approaches

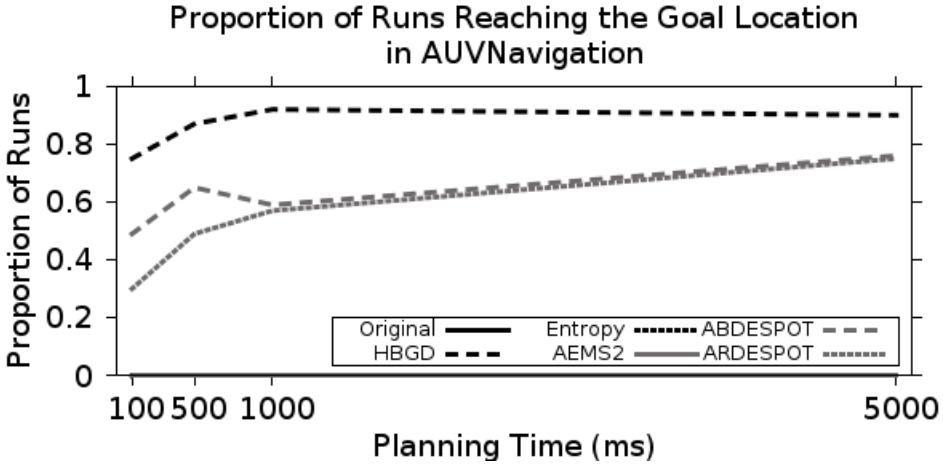

Figure 3.4: Proportion of AUVNavigation Runs Successfully Ending at a Goal Location as Planning Time Increased for Select Approaches

time increased for the best (HBGD) and worst (TBGD on rewards, Entropy on proportion of successful runs) potential functions, as well as Original and the state-of-the-art online POMDP planning algorithms. We also plot in Figure 3.4 the proportion of runs successfully ending at the goal location as a function of planning time and approach.

From these figures, we again observe very successful performance by PBRS with the best potential function: HBGD achieved the highest discounted, cumulative rewards in all but the lowest amount of time for planning ($\tau = 100$ ms) and the highest proportion of goal achievement across all planning times. This is a very interesting result as on the

one hand, AUVNavigation requires long sequences of actions to accomplish its goal, so depth-focused planning approaches like AEMS2 or the MCTS algorithms (ABDESPOT, ARDESPOT) should have an inherent advantage. However, because the required sequences are so long (more than 20 actions to find positive future rewards), even depth-focused planning could not find a path from the agent's starting belief state to the goal location under time constrained planning. Instead, such depth-focused approaches wasted time exploring down paths that earn higher *intermediate rewards* (either not incurring costs for moving forward, or moving along dangerous routes on the bottom of the grid near rocks without incurring high cost at the surface for determining the agent's true location), causing it to waste time planning down paths of overestimated value and underestimating the value of the truly best action sequences (that were either unexplored or under sampled during planning). PBRS with the HBGD, on the other hand, followed an *indicator of high future rewards beyond what depth-focused planning could achieve under such limited time constraints*, and also performed full breadth planning to minimize the risk of following a wrong path initially in the planning tree in order to avoid underestimating the value of the best action sequences, to solve this particular problem. Therefore, full breadth planning with PBRS is very beneficial over state-of-the-art approaches on the type of problem represented by the AUVNavigation benchmark: agents suffering from high uncertainty and requiring long action sequences to find positive future rewards.

Interestingly, the AEMS2 heuristic search algorithm that performed so admirably on the other two benchmark problems (generally better than MCTS and at least competitive with the best potential function using PBRS) performed very poorly on

AUVNavigation. Like full breadth planning without reward shaping (Original), the agent never accomplished the goal and generally had random, non-goal directed behavior when planning with AEMS2 for all amounts of time allocated for planning. Unlike in Tag, in this problem, the heuristic used in AEMS2 was not informative for choosing how to best expand the agent's plan and led to many bad paths and wasted planning time, making it unable to achieve the expected benefits of depth-focused planning, resulting in closer behavior to full breadth planning without implicit estimations of cumulative, future rewards (and similar overall performance to such a planner, Original). Specifically, on this benchmark, the Upper bound rewards (calculated using FIB (Hauskrecht, 2000)) guided the agent as if it had near certain knowledge of the true state of the environment (namely, its current location), but this biased the agent to explore actions maximizing agent rewards under such conditions (namely, attempting to navigate through the rocks). In turn, this led the agent away from exploring action sequences that achieved Stage 1 of the problem (determining the agent's location), and thus left the agent ultimately confused on how to act since its uncertainty was never actually resolved.

3.5.4. Discussion

Considering our results across all three benchmark problems, we now draw some general conclusions about the benefits and drawbacks of using PBRS to improve online POMDP planning. Overall, we empirically discovered from our experimental results that in general, PBRS can be very beneficial to online planning for POMDPs.

First, more often than not, the potential functions employed led to better performance than similar full breadth planning without reward shaping, demonstrating that implicit estimations of cumulative, future rewards (indicated by different types of

information) indeed can improve the quality of plans and subsequent action selection in a wide range of environments. Thus, PBRS is beneficial to consider in environments where full breadth planning might be useful and still gain some of the benefits of depth-focused planning without having to spend the computational costs to explicitly calculate cumulative, future rewards, such as environments where the agent must take care to avoiding reaching dangerous or undesirable situations with no forethought on what to do or how to reach a better situation in order to eventually achieve its goals, as discussed in Section 3.1.

Second, we also gained insights into which types of information measured by potential functions are *most* beneficial to improve agent action selection. In each of the three benchmarks, we observed that domain-dependent information (Type 1, often in the form of goal-directed movement for agents in grid-worlds like our three benchmarks), yielded better performance than either of the two novel types of potential functions proposed in this chapter exploiting properties unique to POMDPs: both domain-independent information providing metareasoning about agent knowledge (Type 2), or belief prioritization providing metareasoning about histories of agent interaction with the environment (Type 3). However, we also observed in each environment that combining these types of potential functions yielded some of the best performances of any potential function type when using these types together, allowing metareasoning from Type 2 and Type 3 to boost performance beyond that achieved by Type 1 alone. Specifically, combinations such as NoExitECD combining Type 1 + Type 2 + Type 3 in RockSample, and HBGD combining Type 1 + Type 3 in AUVNavigation produced the best performances across all potential functions (and generally across almost all considered

approaches to online planning), and EMBD and TBMBD combining Type 1 + Type 2 in Tag also performed well. However, approximations of optimal value functions (Type 4), commonly used as leaf evaluation heuristics, resulted in more mixed results. On the one hand, considering an approximation of the Upper bound on the value function (using FIB (Hauskrecht, 2000)) as a potential function led to the best results on Tag and moderately good results on RockSample and AUVNavigation. On the other hand, considering an approximation of the Lower bound on the value function (using Blind (Hauskrecht, 2000), which is also used in some online POMDP planning algorithms as a leaf evaluation heuristic, e.g., Ross *et al*., 2008), generally led to some of the worst performances and occasionally worse than full breadth planning without PBRS (Original). Overall, we conclude that metareasoning about agent knowledge (using standard measures of certainty like Entropy or TopBelief, Eqs. 3.6-3.7, Type 2) and/or about histories of agent interactions with the environment (belief prioritization, Type 3) combined with any available domain-specific information (e.g., distances to goals, whether measured in a grid space or in some other fashion as observed by Ng *et al*. originally (1999)) was generally the most beneficial type of potential functions to use for PBRS with online POMDP planning. Thus, we recommend starting with such combinations when trying to identify how to best use PBRS on a new POMDP problem. Given that standardized measures exist for Type 2, this hopefully only requires identifying relevant domain-specific information to improve planning, which is already a requirement for PBRS use in any domain, since domain-specific information is generally the only type of information previously considered in the PBRS literature.

Finally, in comparison to three depth-focused state-of-the-art online POMDP planning algorithms: the AEMS2 heuristic search algorithm (Ross & Chaib-draa, 2007) and the DESPOT MCTS algorithms (Somani *et al*., 2013), we also observed that full breadth planning using PBRS led to very favorable agent performance. On the largest and most complicated benchmark problem (AUVNavigation), the best potential function (combining Types 1 and 3 for domain-specific information and metareasoning about histories) outperformed each of the state-of-the-art algorithms for most of the allotted times for planning considered as our time constraints. On the other two benchmarks (Tag and RockSample), the best heuristic (Type 4 using approximations of the Upper bound on the value function for Tag, and combining Types 1, 2, and 3 for domain-specific information and metareasoning about agent knowledge in RockSample) also outperformed at least one of the state-of-the-art algorithms for some of the amounts of time allotted for planning, and was generally competitive on the rest. Thus, it appears overall that some combination of metareasoning (novel to POMDP applications of PBRS) and domain-specific information often provides good enough implicit estimations (or signal indicators) of cumulative, future rewards to allow the agent to save time from *not* explicitly calculating such estimations through depth-focused planning, enabling more time for full breadth planning to avoid the potential pitfalls identified in Section 3.1 from a lack of breadth in planning. Especially noteworthy is that such potential function types do not require precomputation and generally scale well with the size of the POMDP, unlike Type 4 (representing domain information also used by the state-of-the-art algorithms, as explained in the following paragraph), which can be prohibitively expensive to calculate in large POMDPs (especially those with very large state spaces).

Therefore, metareasoning with PBRS might be even more advantageous in even larger planning problems, which we intend to explore in the future (noting again that it already performed the best in our largest, most complicated problem: AUVNavigation).

Although PBRS does add some (domain-specific or domain-independent) information to the agent's planning in addition to the original reward function R, this is similar to the behavior of the state-of-the-art algorithms. Namely, state-of-the-art heuristic search algorithm AEMS2 and the state-of-the-art Monte Carlo search DESPOT algorithms each consider upper $\overline{V}$ and lower bounds $\underline{V}$ on the value function, which are either precalculated offline (e.g., using the FIB or Blind algorithms (Hauskrecht, 2000)) or are calculated directly on the agent's belief state, just like our proposed potential functions. These bounds then indirectly provide the agent with information about its domain that further inform its evaluation of policies while planning. For example, in RockSample, the bounds inform the agent about the locations of rocks, as these are the only locations where the largest positive cumulative rewards exist. Likewise, in AUVNavigation, these bounds inform the agent about the locations of obstacles and the goals as these are the only locations where the upper bound on the value function and the immediate reward are equal (since both types of locations are terminal locations). Instead, our potential function framework provides a principled, mathematical vehicle for considering additional types of information to inform policy evaluation during finite horizon planning with several established theoretical results. The goal of this research is not necessarily to produce a best new planning algorithm that is superior to all state-of-the-art algorithms, but instead: (1) to provide such a vehicle for embedding additional domain-specific or domain-independent information to further improve online planning

for POMDPs, and (2) to explore what types of such information may or may not be useful across different types of planning problems. Identifying valuable types of information could then even be used to create better heuristic search algorithms and further improve the state-of-the-art in online POMDP planning.

However, PBRS is not an approach that works with any potential function and on any problem, as it is possible for a potential function to bias policy evaluation in a bad way. Based on our results, we conclude that some forethought is certainly necessary to identify a good potential function for a particular problem. One necessary component of a good potential function appears to be domain-specific information leading the agent towards its ultimate goal (e.g., distances in grid-based worlds). In environments where such domain expertise is difficult to encode or unknown, PBRS might not be a good choice, as this type of information was generally a prerequisite for the combinations that yielded the best performance, competitive with depth-focused state-of-the-art online POMDP planning algorithms. Indeed, considering the other components (Type 2 and/or 3 metareasoning) individually generally *hurt* agent performance (compared to full breadth planning without reward shaping). In the future, we intend to explore additional types of problem domains where these types of potential functions might be more useful, which we suspect might include (1) environmental monitoring applications (e.g., sensor tracking) where the agent's sole goal is to have high belief certainty, making potential functions of Type 2 more useful alone, as well as (2) problems with multiple subtasks required to complete the agent's ultimate task, where belief state prioritization (potential function Type 3) might be more useful to identify general strategies for accomplishing subtasks individually.

Furthermore, we note that the complexity of potential functions necessary for improving planning increases with the complexity of the problem modeled by the POMDP. That is, in the challenging AUVNavigation problem, simple linear combinations of different types of potential functions were less effective in improving agent performance than in the simpler Tag and RockSample domains. Instead, we had to rely on a more complicated combination of belief prioritization (Type 3) and domain-dependent expected state-based potential (Type 1)—HBGD—in order to best guide the agent through the three subproblems represented by different stages in order to maximize goal achievement and cumulative, discounted rewards. However, even in complex AUVNavigation, simple linear combinations of potential functions still yielded significant improvements in agent performance compared to both full-breadth planning without PBRS (Original) and at least some of the state-of-the-art online planning algorithms. Furthermore, for the simpler benchmark problems (which are still reasonably complex with up to tens of thousands of states, c.f., Section 3.4.1), linear combinations of different types of simple potential functions resulted in significantly improved planning, demonstrating that even simpler potential functions can still boost planning performance.

Moreover, potential functions in complex domains might also require a bit more insight to fine-tune, as well. For example, in the AUVNavigation problem, we eventually added a coefficient of 100,000 (rather than a uniform coefficient of 1 in simpler Tag and RockSample) to the potential functions to properly guide the agent to the goal state from its initial uncertainty. Recall that the successful potential functions (EGD, TBGD, HBGD) reshaped rewards partially based on the multiplicative inverse of the agent's distance from the goal, and thus changes to these functions (Eqs. 3.17-3.18) were quite

small when the agent was highly uncertain (since it was very far from the goal). This meant that the additional signal from the potential function was easily outweighed by the costs of gathering information (namely moving with cost at most -1.73 for moving towards better location information, or -50 for surfacing to discover the agent's exact location and resolve all uncertainty). To "boost" the potential function's signal toward cumulative, future rewards, we had to multiply the signal by a large constant in order to offset the order of magnitude differences between potential differences and reward costs. In other domains with high costs for information gathering, or to otherwise complete necessary intermediate steps towards the agent's ultimate goal, large coefficients might also be necessary. Determining an appropriate coefficient can either be done through experimental investigation, or by analytically comparing the additional shaped reward (from the difference in potential values, Eqs. 3.3-3.4) against the costs associated with actions that maximize or quickly increase shaped rewards. We took a combination of both approaches to set our coefficient for AUVNavigation, although other coefficients might have also been appropriate and led to similar performance. In the future, we intend to develop a greater theoretical understanding of how such coefficients can and should be determined based on the original shape of the reward function and the signals in the potential function. Of note, the state-of-the-art Monte Carlo DESPOT algorithm also utilizes some parameter hand-tuning with respect to the problem domain, most notably the regularization parameter λ used by the ARDESPOT variant (Somani *et al*., 2013). To provide for a fair comparison, we also tuned this parameter for each of our experimental benchmarks, reusing the λ value suggested by Somani et al. in the documentation of the implementation of their algorithm[14] for the Tag and RockSample

[14] Available online at

benchmarks, and after empirically searching for an appropriate value ourselves on AUVNavigation.

3.6. Conclusions and Future Work

In conclusion, we have explored how extending potential-based reward shaping (PBRS) from reinforcement learning (RL) to online planning with POMDPs can be used to improve approximate planning and agent performance given the compuational complexity of planning and limited time constraints. In particular, our aim was to improve long term, cumulative reward estimations in full breadth planning to avoid problems with depth-focused planning identified in Section 3.1. Our approach entails defining a potential function over the agent's belief states that indicates the ability of the agent to earn future rewards. The agent's reward function is then shaped by adding value from this potential function, which leads the agent to be biased towards choosing actions during plan execution that cause the agent to reach belief states that earn larger rewards beyond the planning horizon. We categorize four types of potential functions (with examples), along with hybrid combinations: (1) domain-dependent information from expected state potential (extending directly from the prior use of PBRS with RL and MDPs), (2) domain-independent information measuring a quality or property of a belief state (e.g., certainty), (3) belief prioritization (e.g., priority ordering on belief states), and (4) approximations of the optimal value function. The second and third of these types are novel to POMDPs and offer forms of metareasoning (about agent knowledge and about histories of agent interactions with the environment, respectively) to improve POMDP planning.

---

http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.DownloadDespot

We established from a theoretical perspective that planning with PBRS (1) can, given a finite horizon, lead to different policies than planning with the original unshaped rewards, which in turn enables the agent to earn greater future rewards assuming a good potential function, (2) PBRS can most improve planning when planning horizons are shortest, and (3) even though the agent's reward function is modified, planning with PBRS still optimizes (over the infinite horizon) the agent's original reward function. Finally, we verified these results in practice using an empirical study employing three classic POMDP benchmark problems, demonstrating that under limited time constraints, an agent planning with PBRS better maximized its cumulative, unshaped rewards than planning without PBRS, especially when combining various forms of metareasoning and domain-specific information (Types 1-3). In the most difficult benchmark, we also discovered that PBRS can enable time constrained online POMDP planning to successfully reach the target goal state when such behavior is otherwise incredibly difficult without reward shaping. In particular, time limited planning requires intermediate positive signals indicating appropriate action sequences towards a goal state that are otherwise only discoverable with very deep planning identifying long sequences of actions reaching positive rewards. For complex environments where the only positive reward is earned for reaching the goal state, PBRS can provide such intermediate signals missing from the original reward function to properly guide the agent, making this form of online planning a viable approach. We also compared the performance of PBRS for online POMDP planning against three state-of-the-art online planning algorithms and discovered that PBRS using the best combination of potential functions (Types 1-3 on

two benchmarks, Type 4 on the other) performed comparable to or better than each of the state-of-the-art algorithms on all benchmarks tested.

Furthermore, whilst the focus of this chapter has been on planning, the theoretical results on how to extend PBRS to POMDPs, the novel types of potential functions, and the effect of finite horizons on PBRS are also applicable to partially observable RL.

In the future, we plan to continue this line of research in several directions. First, we intend to further study potential functions to determine what additional qualities or properties of belief states are useful indicators of future rewards in order to better determine how to choose appropriate potential functions given the properties of complex environments (and consider other forms of metareasoning that might be useful to add to other potential functions to further improve agent behavior). Second, we intend to explore the application of PBRS to other settings of planning, including (1) decentralized POMDPs, where planning complexity amongst multiple agents is even more complex than planning with a standard POMDP, and addressing multiagent planning complexity is still an open problem, and (2) offline POMDP planning, where concepts from PBRS such as the potential function could be used to better guide the selection of which belief states to plan around in order to create better plans focused on the most important belief states. Third, PBRS could be potentially included in other types of online POMDP planning algorithms (e.g., employed in Monte Carlo search methods to bias sampling towards large cumulative, future rewards), in which case both PBRS and related optimal reward functions (Sorg, Singh, & Lewis, 2011) would both be of interest to study in order to potentially further improve online POMDP planning.

CHAPTER 4 SITUATIONALLY-AWARE ONLINE HEURISTIC PLANNING FOR HIGHLY UNCERTAIN ENVIRONMENTS

In this chapter, we present a second solution for the Analysis Problem (c.f., Section 1.3), also within the context of POMDPs, a popular approach to deliberative information gathering (c.f., Section 2.2.2). In contrast to our first solution (PBRS for POMDPs, c.f., Chapter 3), this solution enables an agent to reflect upon the benefits of sensing actions (uncertainty reduction) during *planning* in order to lead the agent towards an appropriate policy for guiding action selection, instead of reflecting later during plan *execution*. In this manner, the agent will find policies that cause the agent to first perform high quality deliberative information gathering to benefit its task accomplishment. Altogether, this approach enables the agent to reflect farther into the future than our first solution in order to potentially achieve more targeted, long term benefits from improved information gathering.

This solution features reflection on the benefits of deliberative information gathering in two ways: (1) through the Long Sequence Entropy Minimization (LSEM) heuristic, which enables the agent to expand plans along paths of high quality sensing (through reduction of uncertainty in its knowledge), and (2) through the Difference-based Heuristic Selection (DHS) mechanism, which enables an agent to reflect on its most pressing needs in the context of its current plan: improving its knowledge or earning high rewards through task accomplishment. Together, these advancements in POMDP planning improve both deliberative information gathering, as well as overall agent performance.

Furthermore, as with our PBRS for POMDPs solution, this approach also solves a greater general problem in the POMDP literature: better online planning (i.e., greater cumulative reward achievement) with heuristic search algorithms, especially in highly uncertain domains that are in greatest need of reflective, deliberative information gathering. As such, this chapter is written to address the greater problem. A shorter, earlier version was accepted for publication as a full paper at the AAMAS 2014 conference (Eck & Soh, 2014b). We evaluate our solution in several benchmark POMDP problems, demonstrating that our solution yields successful policies with less planning time in highly uncertain domains and comparable performance in simpler problems.

4.1. Introduction

Intelligent agents and multiagent systems deployed to real-world applications are frequently required to make decisions about how to accomplish goals and tasks while operating in uncertain environments. One popular approach for reasoning under uncertainty is the partially observable Markov decision process (POMDP) (Kaelbling, Littman, & Cassandra, 1998), which offers several key features that enable an agent to decide how to behave even though it faces uncertainty. First, POMDPs model the *causes of uncertainty* in the complex environment's dynamics: both changes to the environment's state over time, as well as partial observability hiding the correct state from the agent. Second, POMDPs also model *the rewards earned and costs incurred* by the agent for taking different actions, enabling the agent to plan sequences of actions earning high expected cumulative rewards that accomplish its tasks and goals.

In particular, within POMDP planning, an agent faces two primary types of uncertainty: (1) uncertainty about the *current state of the environment*, and (2)

uncertainty in the *cumulative rewards* earned for different action sequences. The first form of uncertainty—which we term environment state uncertainty (ESU)—reflects how well the agent understands the current state of its environment and is addressed through a Bayesian framework for updating probabilistic beliefs about the current state of the environment. The second form of uncertainty, on the other hand—which we term cumulative reward uncertainty (CRU)—reflects the agent's understanding of the cumulative rewards it will earn and is addressed through recursively or iteratively computing the series of rewards earned for different action sequences, so long as the agent has time for planning. As the agent plans for an increasing number and depth of action sequences, its estimations of cumulative rewards become more accurate.

In this chapter, we consider the setting of online planning where agents must interleave planning and execution while operating in the environment, and thus have limited amounts of time for planning. Such an approach to planning is popular in the recent literature, as online planning enables an agent to be more *reactive* in real-world environments and *adapt* to unexpected situations. It is also more *efficient* in very large problems (with many possible states, actions, and observations) where having to plan in advance for all possible situations in offline planning can be prohibitively expensive, even though offline planning can afford more time for planning. Instead, online planning enables an agent to repeatedly plan only *locally* around its current belief and choose the best possible action in its current situation without worrying about other situations it might never encounter.

Within online POMDP planning, the state-of-the-art algorithms focus primarily on resolving the second type of uncertainty (CRU), as it is assumed that the first type

(ESU) will naturally be resolved by the Bayesian belief framework as the agent receives observations after taking each action. For instance, heuristic search algorithms such as AEMS2 (Anytime Error Minimization Search 2) (Ross & Chaib-draa, 2007) and FHHOP (Factored Hybrid Heuristic Online Planning) (Zhang and Chen, 2012) guide planning to minimize the uncertainty in the agent's estimations of cumulative future rewards for taking each action. In these algorithms, such uncertainty can be quantified through an *error bound* on future rewards, measured as the difference between upper and lower bound estimates on cumulative rewards. By minimizing this error bound, the agent tries to quickly find plans that are close to optimal by selectively targeting calculations that best improve the agent's estimations of cumulative rewards. The state-of-the-art in Monte Carlo search methods, ARDESPOT (Anytime Regularized DEterminized Sparse Partially Optimal Trees) (Somani *et al*., 2013), similarly guides random sampling of action sequences for cumulative reward calculations during online planning. In several experimental studies across a wide range of different benchmarks, these approaches have been demonstrated to be quite effective (e.g., Ross & Chaib-draa, 2007; Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013; Zhang & Chen, 2012), indeed achieving performance close to (or even exceeding) the state-of-the-art offline planning algorithms for which planning time is less constrained.

However, we will demonstrate that even state-of-the-art online POMDP planning algorithms have difficulty reducing CRU when it is also very difficult to reduce ESU, especially the heuristic search algorithms. We term such environments **highly uncertain environments**. This difficulty arises for several reasons. First, when ESU is high, the agent often requires long sequences of information gathering actions to adequately

understand the current state of the environment. Along these long sequences, the agent's beliefs about the current state of the environment will not change much after any individual action (else long sequences of information gathering would not be necessary). Given the manner in which error bounds on cumulative rewards are calculated, this also implies that the error bound will not change much from one action to the next, making it difficult to plan action sequences with low CRU until ESU is reduced. Second, the upper and lower bounds on cumulative rewards are commonly calculated using approaches (e.g., QMDP, Fast Informed Bound (FIB), Blind (Hauskrecht, 2000)) that assume full (or near full) observability of the environment state, and thus assume no (or little) ESU. As a result, actions taken to reduce ESU are suboptimal under the upper and lower bounds, and are not favored by the state-of-the-art algorithms. Overall, these challenges make it difficult for state-of-the-art heuristic search online POMDP planning algorithms to find acceptable plans within the short times allotted for planning in highly uncertain environments,

In this chapter, we propose a novel heuristic search online POMDP planning algorithm intended to address the challenge of planning in highly uncertain environments where ESU is difficult to reduce. The intuition of our solution is to enable the agent to reflect on its most pressing needs: reducing either ESU or CRU, then plan actions that address the greater need. In particular, we propose a novel heuristic called **Long Sequence Entropy Minimization** (LSEM) that considers the quality of the agent's beliefs about the current environment state in order to plan the long sequences of information gathering actions necessary to reduce ESU. Then, since the agent knows how to handle ESU, we employ situational-awareness within the agent's planning to

reflect on the agent's current uncertainty (both ESU and CRU) to determine which type it most needs to reduce in order to create a successful plan that leads the agent to both understand its environment and earn large, cumulative future rewards. With this situational-awareness, which we call **Difference-based Heuristic Selection** (DHS), the agent switches between planning with different heuristics (both our novel LSEM and state-of-the-art heuristics such as AEMS2 (Ross & Chaib-draa, 2007)) to guide its planning in order to reduce either ESU or CRU as necessary.

To evaluate our novel algorithm, we compare its performance against state-of-the-art heuristic search and Monte Carlo search online POMDP planning algorithms within several classic POMDP benchmarks. We consider both (1) highly uncertain environments that require long sequences of information gathering actions in order to demonstrate the challenges created when it is difficult to reduce ESU and the effectiveness of our approach in dealing with such challenges, and (2) more certain environments where it is easier to reduce ESU, enabling us to evaluate whether our approach is still safe to use when traditional planning algorithms are already effective. We discover that our solution: (1) successfully produces better plans in complex, highly uncertain environments when the agent was most time constrained (finding plans capable of achieving positive rewards over 200 times faster than AEMS2 and FHHOP); (2) earned some of the highest rewards even in an environment that was not highly uncertain; and (3) variants of DHS with a key property (ϵ-optimality) also achieved good performance in the highly uncertain but least complex environment where multistage planning was not necessary. Together these results demonstrate both (i) that our solution

appropriately selects heuristics to guide planning based on the agent's current need, and (ii) that our solution is safe to use in environments that are not highly uncertain.

The rest of this chapter is organizes as follows. In Section 4.2, we provide the necessary background on online POMDP planning and state-of-the-art heuristic search algorithms that are closest in design to our solution. Next, in Section 4.3, we further describe the problem addressed in this chapter: the challenge of planning in highly uncertain environments due to the influence of ESU on the quality of planning. Then, in Section 4.4, we introduce our proposed approach consisting of the LSEM heuristic and the DHS algorithm designed to balance reducing ESU and CRU to improve online POMDP planning in highly uncertain environments. In Section 4.5, we describe the experimental setup used to empirically evaluate the performance of our approach in a range of benchmark POMDP problems, followed by a discussion of the results of those experiments in Section 4.6. We conclude by summarizing our research and proposing interesting future work we intend to explore in Section 4.7.

Of note, this chapter is a significant extension of an earlier conference paper (Eck and Soh, 2014b), providing more in-depth background, problem, and methodology discussions, as well as a larger experimental setup and more theoretical and empirical results.

4.2. Background

4.2.1. *Online POMDP Planning*

In many real-world domains and applications of intelligent agents and multiagent systems, pre-planning using offline planning algorithms is infeasible for the agent. For instance, the problem might be sufficiently large in the size of the state, action, and

observation spaces that planning for all possible future belief states is prohibitively expensive (in both time and especially memory), even with copious amounts of time available for planning during offline planning. Instead, planning locally around only the belief state the agent currently holds is more efficient and effective through online POMDP planning since the latter involves frequently recalculating a plan, and thus the agent need not worry about belief states not reachable in the near future from its current belief.

Most online POMDP algorithms follow the same general search procedure to compute a policy π. In these algorithms, the agent constructs an AND-OR *policy tree* with two types of nodes[15]: OR nodes representing belief states and AND nodes representing actions. To illustrate, we provide an example tree in Figure 4.1. The tree is rooted with an OR node for the agent's current belief state b_c. From this belief state, the agent can choose to take one of several actions $a \in A$ (e.g., $A = \{a^1, a^2\}$ in our illustrative example). Thus, the node b_c has branches to corresponding AND nodes for each action $a \in A$. Since each action can produces multiple observations, each AND node has a branch for each possible observation $z \in Z$ (e.g., $Z = \{z^1, z^2\}$ in our illustrative example) leading to a new belief state OR node $b_{c+1}^{a,z}$. The tree then expands similarly along these non-root OR nodes.

Online planning itself involves three stages, summarized in Algorithm 4.1. First, the agent chooses a leaf node in the tree $b_{c+n}^* \in \mathcal{L}$ (where $\mathcal{L}$ represents the set of leaf nodes in the tree) from which to expand the tree. Second, the agent adds AND nodes for

[15] Given the close relationship between belief states and OR nodes, as well as actions and AND nodes, we reuse the same notation: b represents both a belief state and its corresponding OR node in the policy tree, and a represents both an action and its corresponding AND node in the policy tree.

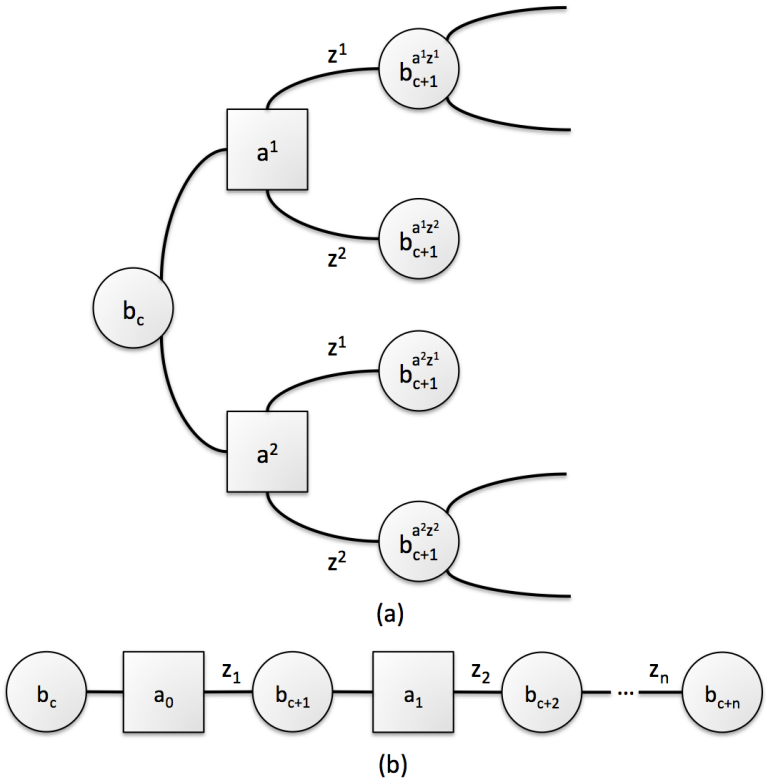

Figure 4.1: (a) Example π Tree with Two Actions and Two Observations with Depth 1, (b) Example Path with Depth n

PolicySearch(b_c, τ)
while TimeSpent()$< \tau$ **and not** DoneSearching()
1. b^*_{c+n} = ChooseLeafNode($\mathcal{L}$)
2. Expand(b^*_{c+n})
3. UpdateAncestors(b^*_{c+n})

end while
return argmax$_{a \in A}$ $\underline{Q}(b_c, a)$

Algorithm 4.1: Generic Policy Search Procedure

each action as children to the chosen leaf node, followed by OR nodes for each observation and resulting belief state from each new AND node. Finally, the agent calculates the expected rewards at the chosen OR node (that used to be a leaf before the tree expanded) and propagates this information backwards along the path from the chosen leaf node to the root of the tree using Eqs. 2.7-2.8 to update the agent's cumulative reward estimates. This additional reward information helps reduce the agent's

uncertainty about the cumulative reward expected from different actions in its current belief state.

Since each iteration of the loop in Algorithm 4.1 iteratively improves the agent's cumulative reward estimates, the algorithm can execute in an anytime fashion. Online planning generally occurs until either (1) the agent has exhausted the amount of time τ allotted for planning, or (2) some other stopping condition is met, such as the agent is certain in its estimates of cumulative rewards and further expansion will not further reduce its CRU.

To account for the fact that rewards beyond a leaf node are initially fully uncertain, online POMDP planning algorithms can improve cumulative reward estimations by adding additional *a priori* value estimates for leaf nodes. Commonly, the agent maintains upper ($\overline{Q}$ and $\overline{V}$) and lower ($\underline{Q}$ and $\underline{V}$) bounds on the discounted, cumulative rewards from each node using very simple pre-computed policies[16] estimating the cumulative rewards from a belief state. In Step 3 of the algorithm, this information is also propagated back through the tree using analogues of Eqs. 2.7-2.8.

After planning, the agent forms its policy π as a subtree of the policy tree, selecting only the actions maximizing the $\underline{Q}(b, a)$ from each belief state b, using an analogue of Eq. 2.9. The agent then executes the action within its policy for the current belief state. Afterwards, the agent can either continue to execute the formed policy for a number of future actions, or it can recalculate a new policy for its new belief state. Either is acceptable, although re-planning is commonly done each time the agent must choose

[16] Using algorithms such as Fast Informed Bound (Hauskrecht, 2000) and Blind policy (Hauskrecht, 2000) for the upper and lower bounds, respectively.

an action. For further details about online planning, please consult a recent survey paper by Ross *et al*. (2008).

4.2.2. *Heuristic Search Algorithms for Online POMDP Planning*

The key difference between different types of online POMDP planning algorithms is how the algorithm selects the leaf belief state $b^*_{c+n} \in \mathcal{L}$ to expand in Step 1. This is because Steps 2 and 3 are relatively straightforward — expansion in Step 2 generally involves the same process (adding child action AND nodes and subsequent belief state OR nodes), and updating cumulative reward estimates in Step 3 always involves computing Eqs. 2.7-2.8 (and the analogues for upper and lower bounds) along the path from the leaf node to the current belief state root b_c.

One very popular type of online POMDP planning algorithm is **heuristic search algorithms**. These algorithms use a heuristic function $h: \Pi(S) \rightarrow \mathbb{R}$ that evaluate the usefulness of expanding a leaf belief state $b^*_{c+n} \in \mathcal{L}$ with respect to improving the overall quality of the agent's estimates of cumulative rewards and thus its policy. Choosing the leaf belief state to expand in Step 1 is as simple as finding the leaf that maximizes this heuristic function:

$$b^*_{c+n} = choose(h) = \text{argmax}_{b_{c+n} \in \mathcal{L}}\, h(b_{c+n}) \tag{4.1}$$

The state-of-the-art heuristic search algorithms use heuristics designed to minimize the agent's overall uncertainty in the cumulative rewards (CRU) earned by the policy formed during planning. That is, they choose to expand the policy tree along leaf belief states that contribute the *most* uncertainty to the agent's cumulative reward estimations, since expanding the tree at these belief states provides more information about the cumulative rewards earned along the path from the leaf belief state back to the

current belief state at the root of the tree. This additional information can then help reduce the agent's CRU along that path, and ultimately in the entire tree.

Within these heuristics, the CRU from a belief state is measured using an error bound on the value function at that belief state:

$$e(b) = \overline{V}(b) - \underline{V}(b) \tag{4.2}$$

where $\overline{V}(b)$ and $\underline{V}(b)$ are the upper and lower bounds on the value function (i.e., upper and lower bounds on cumulative rewards) from the belief state. Given the definition of upper and lower bound, we know that

$$\underline{V}(b) \leq V^*(b) \leq \overline{V}(b) \tag{4.3}$$

where $V^*(b)$ is the optimal reward from a belief state. Thus, minimizing the error bound $e(b)$ causes the distance between the upper and lower bound to shrink and eventually both the upper and lower bound estimates will converge to the optimal cumulative reward under the optimal value function (by the Squeeze Theorem).

Since upper bounds can only decrease and lower bounds can only increase, choosing to expand a leaf belief state will provide information that can only decrease the error bound at the root belief state (after propagating new cumulative reward information back in Step 3 of the algorithm) and thus improves (or does not worsen) the agent's CRU. Moreover, choosing to expand the leaf belief state b_{c+n} with the greatest error bound $e(b_{c+n})$ has the greatest potential to improve the cumulative reward estimate at the root since this node is causing the greatest CRU in the tree. Thus, choosing to expand the tree along maximal error bound leaf belief states can help minimize the agent's overall CRU.

To further improve the quality of planning, the state-of-the-art heuristic search algorithms also incorporate other information into their heuristics to further refine how

the tree expands during planning so that expansions provide the most informative information to improve the cumulative reward estimations. The first such heuristic, Anytime Error Minimization Search 2 (AEMS2) (Ross & Chaib-draa, 2007), also considers the likelihood that the leaf belief state b_{c+n} is ever reached from the current belief state b_c (so that it can focus planning on the belief states the agent will most likely experience), as well as optimistically tries to follow paths where the upper bound on the cumulative rewards is maximized since these paths have the greatest potential to earn the agent large cumulative rewards, which is the goal of planning in the first place. Altogether, the AEMS2 heuristic is given by:

$$h_{AEMS2}(b_{c+n}) = e(b_{c+n}) \prod_{i=0}^{n-1} w(b_{c+i}, a_i) w(b_{c+i}, a_i, z_{i+1}) \tag{4.4}$$

where

$$w(b, a) = \begin{cases} 1 \text{ if } a \in \operatorname{argmax}_{a' \in A} \overline{Q}(b, a') \\ 0 \qquad\qquad\qquad\qquad \text{else} \end{cases} \tag{4.5}$$

favors paths maximizing the upper bound on cumulative rewards $\overline{Q}$ and

$$w(b, a, z) = \gamma P(z \mid a, b) \tag{4.6}$$

considers the probability of making observations that lead to next belief states along the path from the root of the tree to the leaf.

In practice, the AEMS2 algorithm has performed very competitively with state-of-the-art offline algorithms that do not suffer from the same time constraints on planning (e.g., Ross & Chaib-draa, 2007; Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013; Zhang & Chen, 2012). Moreover, it is also guaranteed to find an ϵ-optimal policy (i.e., a policy whose cumulative rewards fall within ϵ of the optimal cumulative rewards) in finite (albeit possibly large) time.

More recently, Zhang & Chen (2012) have proposed a complementary heuristic to work alongside h_{AEMS2} in order to further speed up planning by reducing the agent's CRU even faster. Their heuristic is included in the Fast Hybrid Heuristic Online Planning (FHHOP) algorithm and instead of optimistically following the upper bound on cumulative rewards $\overline{Q}$, it instead favors paths (1) with high *lower bound*s on cumulative rewards $\underline{Q}$ that are used in the actual policy creation stage (c.f., last line of Algorithm 4.1), and (2) considers not just maximal paths according to $\underline{Q}$, but also near-optimal paths to increase the number of leaves with non-zero value that might be selected by the heuristic during each iteration of the planning search algorithm. This heuristic in FHHOP is given by:

$$h_{FHHOP}(b_{c+n}) = e(b_{c+n}) w_{1,2}(b_{c+n}) \prod_{i=0}^{n-1} w(b_{c+i}, a_i, z_{i+1}) \tag{4.7}$$

where

$$w_{1,2}(b_{c+n}) = \max_{i \in [0,n-1]} w_2(b_{c+i}, a_i) \prod_{j=0, j \neq i}^{n-1} w_1(b_{c+j}, a_j) \tag{4.8}$$

selects near-optimal paths according to $\underline{Q}$, permitting suboptimality in one action through w_2:

$$w_1(b,a) = \begin{cases} 1 \text{ if } a \in \operatorname{argmax}_{a' \in A} \underline{Q}(b,a') \\ 0 \qquad\qquad \text{else} \end{cases} \tag{4.9}$$

$$w_2(b,a) = \begin{cases} 1 \text{ if } a \in \operatorname{argmax}_{a' \in A_S} \underline{Q}(b,a') \\ 0 \qquad\qquad \text{else} \end{cases} \tag{4.10}$$

where

$$A_S = \left\{ a \in A \setminus \operatorname*{argmax}_{a' \in A} \underline{Q}(b,a') \mid \overline{Q}(b,a) > \max_{a'' \in A} \underline{Q}(b,a'') \right\}$$

represents the second best actions (according to lower bound estimate $\underline{Q}$) that aren't guaranteed to be suboptimal (i.e., have a lower upper bound than the guaranteed lower bound of another action) and thus wouldn't be pruned by branch and bound pruning.

Comparing h_{AEMS2} and h_{FHHOP}, we note that both aim to reduce the agent's CRU by focusing on the error bound on cumulative rewards $e(b)$. However, they differ in which paths leading to leaf belief nodes that they favor for reducing such uncertainty. h_{AEMS2} optimistically favors paths (and corresponding leaf belief states) that lead to the most *possible* reward, whereas h_{FHHOP} conservatively favors paths (and corresponding leaf belief states) leading the most *guaranteed* reward. Unfortunately, due to its conservative nature, h_{FHHOP} cannot guarantee that it finds an approximately optimal policy in finite time.

To combine the best of both heuristics, the FHHOP algorithm (Zhang & Chen, 2012) actually considers both heuristics at the same time. That is, it calculates both heuristics for all leaf belief states when deciding which leaf belief state $b^*_{c+n} \in \mathcal{L}$ to expand. After calculating both, it performs a weighted comparison to bias selection to favor the heuristic that has best reduced the error bound $e(b)$ in past iterations. In this way, the algorithm gains the theoretical benefits of the h_{AEMS2} heuristic (i.e., finding an ϵ-optimal policy in finite time) by following the h_{AEMS2} heuristic often enough, yet it can possibly find high quality plans *faster* than AEMS2 by using the h_{FHHOP} heuristic when it better guides planning. Moreover, this algorithm *learns* over time which heuristic to use in order to best reduce uncertainty in cumulative reward estimations and result in the best plans for maximizing agent rewards while operating in the environment.

Of note, the *Factored* portion of the FHHOP name refers to the fact that the algorithm also exploits the state-of-the-art in POMDP representations: the Mixed Observability Markov Decision Process (MOMDP) (Ong *et al*., 2010). In a MOMDP, the state space $S = \mathcal{X} \times \mathcal{Y}$ is factored into a set of fully observable states $\mathcal{X}$ (that are always directly observed by the agent) and a set of partially observable states $\mathcal{Y}$ (that are understood through observations Z, as in the traditional POMDP representation, c.f. Section 2.2.2). Since fully observable states are not hidden, this representation speeds up several important calculations frequently performed by agents while planning, especially Eq. 2.4 since only some state variables are hidden and need to be estimated using the Bayesian belief state. Of course, there is nothing special about this representation that means that other online POMDP planning algorithms such as AEMS2 or our proposed solution cannot be used with MOMDPs, so in our experimental setup (c.f., Section 4.5), we use this representation with all algorithms.

4.3. Problem

Although the h_{AEMS2} and h_{FHHOP} heuristics are well designed to reduce agent uncertainty about *cumulative rewards* while the agent is planning a policy to control its actions, they assume that the agent's uncertainty about the *current state of the environment* will simply be resolved by whatever observations are received after taking actions. That is, the heuristics do not consider any information describing the uncertainty in the agent's beliefs about the current environment state when deciding how to expand the agent's plan, and instead rely on the Bayesian framework for belief updates (Eq. 2.4) to handle ESU.

In many kinds of environments, this is not a concern, and both the AEMS2 and FHHOP algorithms have performed quite well on a range of POMDP benchmark problems (e.g., Ross & Chaib-draa, 2007; Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013; Zhang & Chen, 2012) due to several reasons. First, in these environments, the environment state might be relatively easy to identify, e.g. due to highly accurate direct observations about the state of the environment, and thus special care is *not* needed to deal with ESU. Second, the agent might receive relatively large rewards or costs based on periodically acting on its knowledge of the environment state. Thus, planning actions to receive these easily identifiable high rewards naturally requires planning actions that first perform a small number of information gathering actions to understand the correct environment state. Finally, if the problem is sufficiently small (especially in the number of states, but also the number of actions and observations), then planning might be relatively easy in general.

Unfortunately, there are also many real-world environments and applications of intelligent agents and multiagent systems where ESU is much more difficult to reduce, which we term *highly uncertain environments*. This difficulty could be due to a number of factors. First, there might be many states of the environment that can generate the same observation. In which case, such an observation does not help us discriminate between which is the next state since many possible next states could have generated that observation. Thus belief updates in Eq. 2.4 are rather uninformative whenever the agent receives such an observation. For instance, if most states s' are equally likely to produce a recent observation z after the recent action a, then the $O(s', a, z)$ term will be equal in

Eq. 2.4 for each such next state s', resulting in minimal changes[17] to the belief state $b^{a,z}$. Second, each action and next state could generate a large number of possible observations, meaning that each new observation provides little information about the next state of the environment after the action is taken. In both scenarios, each belief update is at risk of providing minimal changes to the agent's belief state (Eq. 2.4).

As a result of these difficulties, highly uncertain environments generally require long sequences of information gathering actions in order to properly reduce ESU. This has two important implications for planning with state-of-the-art heuristic search algorithms that specialize in reducing CRU: (1) ESU will lead to similar error bound $e(b_{c+n})$ values across leaf nodes, causing the state-of-the-art heuristics to *fail to discriminate* between "good" and "bad" leaf nodes to expand during planning, and (2) paths containing the necessary long sequences of information gathering actions often fail to have maximal upper or lower bound values, causing h_{AEMS2} and h_{FHHOP} to initially *ignore these necessary action sequences* during policy tree expansion.

First, recall that the error bound $e(b)$ of a leaf belief state b is computed as the difference between the upper and lower bounds on cumulative rewards (Eq. 4.2): $\overline{V}(b) - \underline{V}(b)$. For leaf belief states b_{c+n}, both the upper and lower bounds are represented by piecewise linear convex vectors called *alpha vectors* with one alpha vector α_a per action a (Hauskrecht, 2000). The upper or lower bound is then calculated as the dot product of the belief state b_{c+n} with the alpha vector α_a giving the greatest value across the entire set of alpha vectors:

$$\overline{V}(b_{c+n}) = \max_{\alpha_a \in A_U} \alpha_a \cdot b_{c+n} \tag{4.11}$$

[17] These small changes only reflect possible state transitions from the $T(s, a, s')$ component from the prior belief b and do not consider information contained in observations.

$$\underline{V}(b_{c+n}) = \max_{\alpha_a \in A_L} \alpha_a \cdot b_{c+n} \quad (4.12)$$

where A_U is the set of alpha vectors for the upper bound, and A_L is the set of alpha vectors for the lower bound.

Since the upper and lower bounds of a leaf belief state b_{c+n} are computed as dot products with b_{c+n}, the upper and lower bound values will be very similar for belief states that are also similar. As described above, in highly uncertain environments, the agent's beliefs will not change much until it has performed a long sequence of information gathering actions. Hence, after taking any given action and receiving any given observation, the agent's next belief will be very similar to its previous belief. Thus, while expanding the policy tree, a child OR node will have a belief very similar to its parent OR node, and sibling OR nodes will also have similar beliefs. Therefore, the upper and lower bounds $\overline{V}$ and $\underline{V}$, and consequently the error bound e, will be similar across the leaves of the policy tree until the agent has gathered sufficient information to reduce its ESU. As a result, the error bound will not appropriately distinguish which belief states to expand while planning, so existing heuristics relying on the error bound will be less useful in guiding planning to reduce CRU (due to high amounts of ESU).

Second, in the algorithms used to compute the alpha vectors, such as Fast Informed Bound (FIB) or QMDP for the upper bound, and Blind for the lower bound (Hauskrecht, 2000), the algorithms assume full (or near) full observability of the environment state by transforming the original POMDP to a simpler (fully observable) MDP model. In which case, information gathering actions have little value to the agent since it has no ESU. Thus, information gathering actions (which also often incur some cost in return for information) generally have smaller upper and lower bounds than other

actions. Hence, information gathering actions will rarely maximize the upper $\overline{Q}$ and lower $\underline{Q}$ bounds on cumulative rewards from an action AND node in the policy tree. Therefore, the h_{AEMS2} and h_{FHHOP} heuristics are also biased (in the $w(b,a)$ and $w_{1,2}(b,a)$ components, Eqs. 4.5, 4.8-4.10) to *not* select leaf belief states along paths containing the necessary long sequences of information gathering actions required to reduce ESU. Thus, the agent will not discover the large cumulative rewards ultimately possible after reducing its ESU, and instead long sequences of information gathering actions will not be performed by the agent while executing its plan.

Overall, both of these problems greatly reduce the effectiveness of state-of-the-art heuristic search algorithms to create high quality plans for agents operating in highly uncertain environments, due to their inability to reduce ESU. We note that the ϵ-optimal guarantees of AEMS2 and FHHOP (c.f., Section 4.2.2) do imply that *eventually* the algorithms will produce near optimal policies, even in highly uncertain environments, but such policies could take much longer amounts of time than available during online planning. In the following section, we propose an algorithm that can find good plans faster using online POMDP planning in highly uncertain environments.

4.4. Solution Approach

In this section, we propose our solution to improve online POMDP planning in highly uncertain environments where the agent requires long sequences of information gathering actions in order to reduce uncertainty about the environment state. First, we describe the intuition for our solution: splitting planning into stages, where each stage reduces a different type of uncertainty to produce high quality plans for the agent in limited amounts of time allocated for planning. Second, we introduce a novel heuristic

for guiding planning to reduce ESU by biasing policy search towards policies favoring the necessary long sequences of information gathering. Then, we introduce a situationally-aware algorithm capable of identifying which planning stage the agent is currently in so that it knows how to guide its planning using different heuristics. Finally, we analyze the performance of our algorithm from a theoretical perspective in order to discover important properties.

4.4.1. *Planning Stages*

To develop a solution for improving online POMDP planning in highly uncertain domains, we start with a simple observation. The problem with existing heuristics—that work very well in environments with less ESU—is that they fail to plan to perform the necessary long sequences of information gathering actions needed to understand the environment. If, instead, the agent had a method for planning the needed long sequences of information gathering actions to reduce ESU, then after those actions were executed, the agent would be in a position no different from planning in environments that are not highly uncertain. At this point, existing state-of-the-art heuristics should continue to work well by planning actions that maximize the agent's rewards by reducing CRU during planning.

Based on this observation, we propose splitting planning in highly uncertain environments into two stages, depicted in Figure 4.2. In the first stage, the agent should focus on reducing ESU by planning for, then performing, the necessary long sequences of information gathering actions needed to understand the current state of the agent's environment. This enables the agent to move from an initial starting point of high uncertainty about the environment to a position where the agent has a more certain

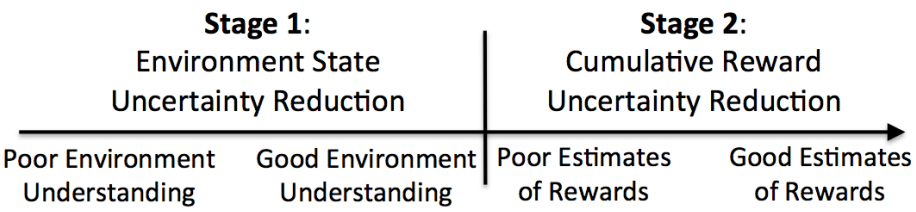

Figure 4.2: Stages of Planning in Highly Uncertain Environments

understanding of the environment. Afterwards, the agent can exploit this understanding of the environment in order to quickly reduce its CRU in order to earn the agent large rewards while operating in the environment. Splitting planning into two such stages has several advantages.

First, it enables the agent to focus most of its planning efforts towards reducing one type of uncertainty at a time, based on its most pressing need: first ESU, then CRU.

Second, by focusing on reducing ESU first, the agent will be in a position in Stage 2 where existing heuristics are quite appropriate to guide planning, allowing the agent to reuse previously reported techniques that have been demonstrated to work well in similar conditions (e.g., Ross & Chaib-draa, 2007; Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013; Zhang & Chen, 2012).

Third, by focusing on reducing ESU first, the agent can achieve beliefs close to pure certainty where the agent is close (temporarily at least) to full observability, which is the condition under which the upper bound on agent rewards $\overline{V}$ are calculated using algorithms such as FIB or QMDP (Hauskrecht, 2000). This implies that following the sequence of actions that maximize the upper bounds $\overline{V}$ and $\overline{Q}$—as favored by the $w(b,a)$ component of h_{AEMS2}—will quickly lead the agent to the sequence of actions that will also maximize its cumulative rewards. Thus, reducing ESU first can potentially improve the effectiveness of state-of-the-art heuristics like h_{AEMS2} in reducing CRU.

In order to produce an algorithm that successfully controls online POMDP planning through both stages in highly uncertain domains, our solution contains two primary novel contributions. First, we propose a novel heuristic that guides planning to expand the policy tree during the first stage in order to plan the long sequences of information gathering actions necessary for ESU reduction. The second stage, on the other hand, does not need a new heuristic as we can simply reuse the state-of-the-art heuristics such as h_{AEMS2} (or h_{FHHOP}) for CRU reduction. Instead, we also contribute a novel mechanism providing situational-awareness to identify which stage the agent is currently in, then selects the appropriate heuristic to guide planning.

4.4.2. *LSEM Heuristic*

In order to guide planning to form policies with long sequences of information gathering actions necessary to reduce ESU, we propose a novel heuristic called **Long Sequence Entropy Minimization (LSEM)**. This heuristic directly measures the ESU in an agent's belief states so that the agent can identify how confused it would be about the environment in each belief state, and then expand the policy tree in such a manner that the agent's beliefs are most certain and ESU is minimized.

In particular, because a belief state b is represented by a probability distribution, we can directly measure the uncertainty in the agent's belief using the entropy function (Araya-Lopez *et al*., 2010):

$$H(b) = -\sum_{s \in S} b(s) \log b(s) \tag{4.13}$$

which gives us a measure of ESU (in the range $[0, \log|S|]$), similar to the measure for CRU $e(b)$.

However, unlike $e(b)$, expanding the policy tree along leaf belief states $b_{c+n} \in \mathcal{L}$ with *greatest* $H(b)$ will *not* necessarily reduce the overall ESU in the policy. This critical insight stems from the fact that $H(b)$ can actually *increase* from a belief state to its children (e.g., if the agent receives an observation in evidence of a next state that is contrary to its current beliefs), whereas $e(b)$ values can only *decrease* as the policy tree is expanded (based on the definition of upper and lower bounds in Eq. 4.2). So, in order to minimize ESU, we want to select belief states with *lower* $H(b)$ values.

Since heuristic search algorithms choose leaf belief states with the *highest* heuristic values (Eq. 4.1), we consider instead the agent's *certainty* in a belief (which is the additive inverse of uncertainty):

$$C(b) = \log|S| - H(b) \tag{4.14}$$

which is maximized whenever $H(b)$ is minimized. Considering $C(b)$ in a heuristic thus guides the agent to minimize ESU.

Moreover, just as the h_{AEMS2} and h_{FHHOP} heuristics consider more than just the actual measure of CRU $e(b)$ in their calculations to more efficiently guide expansion of the policy tree, we also add additional measures to our LSEM heuristic to quickly reduce ESU. We explain the designed purpose of each additional component below. Our entire heuristic is given by:

$$h_{LSEM}(b_{c+n}) = C(b_{c+n})d(b_{c+n})\overline{V}(b_{c+n})\prod_{i=0}^{n-1} w(b_{c+i}, a_i, z_{i+1}) \tag{4.15}$$

First, the $d(b_{c+n})$ term:

$$d(b_{c+n}) = 1 + \log(n+1) \tag{4.16}$$

biases h_{LSEM} to favor expanding the tree using deeper leaf belief states to encourage the long sequences of actions necessary to gather information. Second, the $w(b, a, z)$ terms

from Eq. 4.6 favor expanding the most likely leaf belief states so that planning occurs along the situations the agent is most likely to actually encounter when it follows the formed policy. Finally, the $\overline{V}(b_{c+n})$ term encourages planning to optimistically explore policies that have the potential to earn the *greatest* future cumulative rewards to setup planning for Stage 2 after ESU is adequately reduced (instead of becoming stuck in local optima where the agent fully understands the environment state but cannot earn large future rewards). Of note, we consider $\overline{V}(b_{c+n})$ in h_{LSEM} instead of the selector $w(b,a)$ that only considers leaf belief states along paths always maximizing upper bound rewards (Eq. 4.5) as in h_{AEMS2}. This enables h_{LSEM} to tradeoff some reduction in upper bound rewards in return for less ESU, relying on planning in Stage 2 to find the best possible policy for maximizing cumulative rewards.

Within h_{LSEM}, we multiply each component for two reasons. First, it permits us to avoid having to normalize the values of the different components against one another, as we would have to do if the components were added together so that one wouldn't automatically outweigh the others. This is important because the components have vastly different ranges: for example, $C(b)$ has a range of $[0, \log S]$ in all environments whereas the range of $\overline{V}(b)$ is entirely environment-specific. Second, this practice follows in the tradition of other heuristics, such as state-of-the-art h_{AEMS2} and h_{FHHOP}.

Analyzing the structure of h_{LSEM}, we note that it has several valuable properties for guiding planning in Stage 1 of our proposed solution. First, each component has a non-negative range. Thus, the entire product is non-negative and increases for leaf belief states occurring along sequences of actions that perform long sequences of information gathering needed by the agent in Stage 1. Therefore, h_{LSEM} is maximized exactly for the

leaf belief states that best guide planning to reduce ESU. Second, $d(b_{c+n})$ has a diminishing returns property, meaning that as n increases, $d(b_{c+n})$ increases less and less. Thus, further and further increasing the depth at which the policy tree is expanded contributes less and less increase in the heuristic value. This implies that the heuristic will avoid maximizing the depth of planning at the expense of the other components. So although long sequences of information gathering are beneficial, the heuristic will still expand leaf belief states closer to the root of the tree if those leaf belief states offer more promising reductions in ESU, as desired.

4.4.3. *DHS Situational-Awareness*

Although our proposed LSEM heuristic is designed to successfully guide planning during State 1—ESU reduction—it is not as well designed to reduce CRU in Stage 2. This is because it does not directly consider CRU as measured by $e(b)$, which is orthogonal to ESU $H(b)$ (although high levels of $H(b)$ make it harder to reduce $e(b)$, c.f., Section 4.3). Therefore, planning solely with h_{LSEM} is not ideal. Instead, we propose using different heuristics for each stage of planning to best exploit the unique advantages of each heuristic and produce the best quality plans.

However, deciding which heuristic to use while planning is not a trivial problem. If we identify different heuristics as being best employed in different stages, such as h_{LSEM} in Stage 1 and h_{AEMS2} in Stage 2, then the agent must be aware of which stage it is currently in while planning so that it knows which heuristic to use to guide policy tree expansion.

Ideally, we could just add our h_{LSEM} heuristic to existing algorithms that already consider multiple heuristics to improve online POMDP planning. As briefly described in

Section 4.2.2, the FHHOP algorithm (Zhang & Chen, 2012) was the first heuristic search online POMDP algorithm to tradeoff between different heuristics during planning. In FHHOP, the algorithm learns which heuristic to use (h_{AEMS2} or h_{FHHOP}) based on their past successes in reducing CRU. Unfortunately, this approach has two key problems that prevent it from being readily adapted to accept other heuristics, such as h_{LSEM}. First, FHHOP relies on the fact that both heuristics it considers are working towards the same goal—CRU reduction by minimizing $e(b)$. Thus, their learned past successes can be directly compared—the agent can compare how well each reduced a single objective: $e(b)$. Since other heuristics such as h_{LSEM} are working towards a different goal with a different objective, it is unclear how to compare the success of h_{LSEM} in reducing ESU $H(b)$ against the success of h_{AEMS2} or h_{FHHOP} in reducing CRU $e(b)$. Second, both h_{AEMS2} and h_{FHHOP} measure very similar information about leaf belief states: (1) error bound $e(b_{c+n})$, (2) the probability of observations leading to b_{c+n}, and (3) whether or not the path from the root node b_c to b_{c+n} is optimal (or near optimal) with respect to the upper or lower bounds on cumulative rewards. Thus, the two heuristics naturally have the same ranges and do not require any kind of normalization to compare their values when choosing a heuristic for policy tree expansion. Other heuristics such as h_{LSEM} have very different ranges, and it is unclear how to normalize each heuristic to make any comparisons between their values fair and impartial. Together, these problems make it very difficult to add additional heuristics to FHHOP without modifying the way the algorithm chooses between heuristics when expanding the agent's policy tree.

Identifying Current Stage. To decide instead how to select which heuristic to use for guiding planning based on the agent's current situation (either Stage 1 or Stage 2), we

start by considering the differences between the two proposed stages for planning. We observe that in Stage 1, the key objective is to reduce the agent's uncertainty about the c*urrent state of the environment (ESU)*, measured by $H(b)$. As this type of uncertainty is reduced, the values of $H(b)$ will change from almost pure uncertainty to very low levels of uncertainty. At the same time, during Stage 1, the agent's measure of CRU $e(b)$ will not change very much, as identified as a key problem for state-of-the-art heuristics in Section 4.3. Therefore, in Stage 1, $H(b)$ will change much more than $e(b)$.

Likewise, in Stage 2, the key objective is to reduce the agent's uncertainty about its *cumulative rewards (CRU)*, measured by $e(b)$. As this type of uncertainty is reduced, the values of $e(b)$ will change from very high values (where the upper $\overline{V}(b)$ and lower $\underline{V}(b)$ bounds are far apart) to very low values (where $\overline{V}(b)$ and $\underline{V}(b)$ become closer and closer to $V^*(b)$), as discussed in Section 4.2.2 (for Eq. 4.3). At the same time, the agent will already have low amounts of ESU $H(b)$ (which was already resolved in Stage 1), so this type of uncertainty will not change much. Thus, in Stage 2, $e(b)$ will change much more than $H(b)$.

Based on these observations, we can design an algorithm for choosing an appropriate heuristic to use to guide planning through the two stages necessary in highly uncertain environments. In Stage 1, when $H(b)$ is changing as the policy tree expands, then its additive inverse $C(b)$ is also changing, so the h_{LSEM} values will be changing more than h_{AEMS2} (which relies on $e(b)$ that does not change much in Stage 1). Likewise, in Stage 2, $e(b)$ is changing as the policy tree expands, so h_{AEMS2} will be changing more than h_{LSEM} (which relies on $C(b)$ that does not change much in Stage 2). Therefore, by comparing the *change in values* of the heuristics, the agent can identify

both (1) which stage of planning it currently faces, and (2) which heuristic is most appropriate for that stage.

To calculate and then compare the changes in values for the different heuristics, we consider the following general process, summarized in Algorithm 4.2. For each $h_j \in \{h_1, h_2, \ldots, h_k\}$ (where the agent considers k heuristics), the agent calculates the heuristic value $h_j(b_{c+n})$ for all $b_{c+n} \in \mathcal{L}$, then picks the leaf belief state maximizing each heuristic:

$$b_{c+n}^j = choose(h_j) \tag{4.17}$$

where the $choose$ function is defined in Eq. 4.1. Next, the agent compares the heuristic value at the chosen leaf belief state b_{c+n}^j and its parent belief state b_{c+n-1}^j in the path back to the root of the tree b_c to compute how much the (undiscounted[18]) heuristic value changed when the parent node b_{c+n-1}^j was expanded previously to add the chosen leaf belief state b_{c+n}^j:

$$\Delta_{h_j} = \left[h_j(b_{c+n}^j)/\gamma - h_j(b_{c+n-1}^j)\right]/h_j(b_{c+n-1}^j) \tag{4.18}$$

for heuristics that increase as the agent reduces the corresponding type of uncertainty, such as h_{LSEM}, and

$$\Delta_{h_j} = \left|h_j(b_{c+n}^j)/\gamma - h_j(b_{c+n-1}^j)\right|/h_j(b_{c+n-1}^j) \tag{4.19}$$

for heuristics that monotonically decrease as the agent reduces the corresponding type of uncertainty, such as error-bound $e(b)$ based heuristics (e.g., h_{AEMS2}).

Based on Eqs. 4.18-4.19, we observe that the higher the value of Δ_{h_j}, both (1) the more the heuristic is changing, and (2) the more appropriate the heuristic is for the

[18] We divide the $h_j(b_{c+n}^j)$ term by γ in Eqs. 25-26 to remove the difference caused solely by discounting in $w(b, a, z)$, as opposed to the actual change in the heuristic values.

ChooseLeafNodeUsing DHS($\mathcal{L}$)

// find the leaf belief states maximizing each heuristic
for $j \in \{1, 2, \dots, k\}$
  $b_{c+n}^{j} \leftarrow choose(h_j)$ // Eq. 4.1
end for

// compute the change in heuristic values along the chosen leaf belief states
for $j \in \{1, 2, \dots, k\}$
  Compute Δ_{h_j} using Eq. 4.18-4.19 with b_{c+n}^{j}
end for

// choose the heuristic with maximum change weighted by the rewards upper bound
$h^* \leftarrow null$
$max\Delta \leftarrow -\infty$
for $j \in \{1, 2, \dots, k\}$
  // Eq. 4.20
  $dhs \leftarrow \Delta_{h_j} \overline{V}(b_{c+n}^{j})$
  if $dhs > max\Delta$ **then**
    $max\Delta \leftarrow dhs$
    $h^* \leftarrow h_j$
  end if
end for
return $choose(h^*)$ // Eq. 4.21

Algorithm 4.2: DHS Situationally-Aware Mechanism for Choosing the Leaf Node to Expand in Algorithm 4.1

current stage of planning. On the one hand, when $\Delta_{h_{LSEM}} > \Delta_{h_{AEMS2}}$, then the agent is in Stage 1 and h_{LSEM} is the better heuristic to use to reduce the agent's most pressing uncertainty: ESU. On the other hand, when $\Delta_{h_{AEMS2}} > \Delta_{h_{LSEM}}$, then the agent is in Stage 2 and h_{AEMS2} is the better heuristic to use to reduce the agent's most pressing uncertainty: CRU. Moreover, state-of-the-art heuristics (AEMS2, FHHOP) and our LSEM heuristic each assume that the best policies occur along paths where the heuristic values are greatest, so the fastest improving leaves (as measured by the Δ_{h_j} function, Eqs. 4.18-4.19) represent the best possible branches to expand. Since this mechanism makes decisions based on the differences in heuristic values from leaf belief states to their

parents as a measure of the rate of change in a heuristic, we call our situationally-aware heuristic selection mechanism *Difference-based Heuristic Selection* (DHS).

Transition between Stages. As a final step of our mechanism for selecting heuristics to use to guide planning, we want to smooth out the transition between the two stages of planning. That is, we want to improve planning when the agent is nearing the end of Stage 1 and starting to begin Stage 2. At this point, the change in values $\Delta_{h_{LSEM}}$ will be decreasing towards zero (as environment certainty is resolved) and $\Delta_{h_{AEMS2}}$ will be starting to increase away from zero (as CRU starts to become reduced). When this happens, both heuristics look similarly appropriate (i.e., $\Delta_{h_{LSEM}} \approx \Delta_{h_{AEMS2}}$), so it becomes difficult to properly choose one over the other. Moreover, towards the end of Stage 1 h_{LSEM} might inspire the agent to reduce ESU farther than it needs to for h_{AEMS2} to finish planning the proper sequence of actions to take that maximize cumulative rewards, which we want to avoid.

To handle this transition between planning stages, the following equation represents the final rule for selecting between heuristics in DHS:

$$h^* = \text{argmax}_{j\in\{1,2,\ldots,k\}} \Delta_{h_j} \overline{V}\left(choose\left(h_j\right)\right) \tag{4.20}$$

and the algorithm selects the following leaf belief state to expand in each iteration of the planning algorithm (Algorithm 4.1, c.f., Section 4.2.1):

$$b^*_{c+n} = choose(h^*) \tag{4.21}$$

The rationale behind Eqs. 4.20 and 4.21 is as follows. Here, we optimistically bias the heuristic selection based on the upper bound on cumulative rewards expected from the leaf belief state favored by the selected heuristic (similar to optimistic biasing in

offline algorithms such as HSVI (Smith & Simmons, 2004) or online algorithms such as AEMS2 (Ross & Chaib-draa, 2007)). Thus, when planning transitions between Stage 1 and Stage 2 and the Δ_{h_j} values for the heuristics approach one another, the agent favors planning along paths of actions that have the *potential to lead to greater cumulative future rewards*, since earning such rewards is the ultimate goal of the agent (towards which the agent strives in Stage 2 of planning).

Of note, our situationally aware DHS solution is somewhat related to another POMDP planning algorithm used in the context of multiagent I-POMDPs: bimodal switching (Sonu & Doshi, 2013). In particular, Sonu & Doshi's solution metacognitively analyzes the agent's CRU to decide how to plan: either in the simpler single agent case (to quickly reduce CRU) or in the more complicated multiagent case (to achieve even greater rewards by taking into account other agents' actions). Our DHS solution is similar in that it also metacognitively chooses how to plan to reduce uncertainty (both ESU and CRU), but we do not consider CRU directly when switching stages, nor do we consider multiagent planning. Instead, we improve single agent planning by splitting planning into stages each considering the same complexity but different objectives, rather than different complexities (single agent vs. multiagent) with the same objective.

4.4.4. *Theoretical Analysis*

Finally, now that we have described our solution consisting of both the LSEM heuristic and DHS algorithm for choosing the heuristic to guide planning, we discuss the theoretical properties of the solution.

Namely, recall from Section 4.2.2 that state-of-the-art heuristic search online POMDP planning algorithms AEMS2 (Ross & Chaib-draa, 2007) and FHHOP (Zhang &

Chen, 2012) have the beneficial property of being ϵ-optimal, meaning that they can return a policy with expected value within a desired ϵ of the value of the optimal policy using only a finite (albeit possibly large) amount of time for planning. Thus, given enough planning time, the algorithms are guaranteed to find a very good approximation of the optimal policy (we reuse notation here to term such a policy an ϵ-optimal policy), which is a desirable property of an anytime planning algorithm. We desire that our solution also have this property.

Unfortunately, the DHS approach to selecting the leaf belief state to expand as presented in Algorithm 4.2 (used as Step 1 of Algorithm 4.1) and defined by Eqs. 4.20-4.21 cannot guarantee this property in its current form. This is due to (1) the inclusion of the h_{LSEM} heuristic that is only designed to reduce ESU and will not necessarily reduce CRU to less than a desired ϵ throughout the policy tree, and (2) we cannot guarantee that DHS will not choose h_{LSEM} an infinite number of times and in turn not choose h_{AEMS2} often enough to find an ϵ-optimal policy. Therefore, we cannot guarantee that DHS is ϵ-optimal, but in practice (as we will test in the following experimental setup) it still should call h_{AEMS2} sufficiently often to properly guide planning toward good estimations of cumulative rewards.

On the other hand, we can modify Eq. 4.20 slightly to produce variants of DHS that are guaranteed to be ϵ-optimal. We propose two such variants here: (1) DHS-m, and (2) SoftMaxDHS.

First, DHS-m is a minor modification of Eq. 4.20 that deterministically forces h_{AEMS2} to be chosen often enough to guarantee that the algorithm is ϵ-optimal:

$$h^* = \begin{cases} h_{AEMS2} & \text{if } N \bmod m = 0 \\ h^* \text{ selected by Eq. 4.20} & \text{otherwise} \end{cases} \tag{4.22}$$

where N is the number of times the policy tree has been expanded and $m \in \mathbb{N}$ is any natural number. For DHS-m, we find that:

> **Theorem 4.1:** DHS-m using Eq. 4.22 is ϵ-optimal, so long as h_{AEMS2} is one of the heuristics available to the selection mechanism.

Proof: Let ϵ be given. AEMS2, which also follows Algorithm 4.1, is ϵ-optimal, so it will find an ϵ-optimal policy in a finite number of iterations $M < \infty$. Hence, choosing h_{AEMS2} within M iterations in Step 1 of the loop in Algorithm 4.1 results in a ϵ-optimal policy. DHS-m is guaranteed to choose h_{AEMS2} M times within mM iterations, simulating at worst the behavior of AEMS2 during the M iterations that h_{AEMS2} is selected. We know that $mM < \infty$ since $m \in \mathbb{N}$ and $M < \infty$, so DHS-m will also find an ϵ-optimal policy in finite time. Since ϵ was arbitrary, DHS-m is ϵ-optimal. ■

The value chosen for m in DHS-m (Eq. 4.22) has several important implications on the behavior of the algorithm. With a smaller m, the upper bound on the number of iterations (mM) required to find an ϵ-optimal policy in a smaller than using a larger m. However, a smaller m also causes h_{AEMS2} to be chosen much more often than it might be in original DHS (and thus used more often in Stage 1 of planning where it is less effective than h_{LSEM}). As a result, DHS-m might be less efficient in practice, where a greater use of h_{LSEM} in Stage 1 could speed up planning by focusing policy tree expansion around the necessary long sequences of information gathering actions needed to reduce ESU.

As a starting point (also used in our experimental setup to follow), we suggest setting $m = k$, the number of heuristics considered by the selection mechanism, in DHS-m. In the future, we intend to explore methods for adapting this parameter within the algorithm, rather than requiring a static choice in advance. For example, m could be set

proportional to $H(b_c)$, where a larger m would occur when the agent's current belief is most uncertain about the current state of the environment, allowing the algorithm to rely more often on h_{LSEM} which addresses its greatest need. Likewise, a smaller m would occur when the agent is more certain about the environment, and h_{AEMS2} (which is chosen more often when m is small) is more useful for guiding planning. Alternatively, when the agent is facing high costs for actions, a smaller m would enable the agent to focus more on improving its estimates of cumulative rewards to reduce overall costs by choosing h_{AEMS2} more often.

Second, in contrast to *deterministic* DHS-m, SoftMaxDHS represents a *stochastic* variant of DHS that relies on the values of $\Delta_{h_j}\overline{V}\left(choose\left(h_j\right)\right)$ to have a greater influence on the heuristic selected, following closer in spirit to the original DHS mechanism. In SoftMaxDHS, we replace Eq. 4.20 with:

$$h^* \sim P\left(h_j\right) = \frac{e^{\Delta_{h_j}\overline{V}\left(choose\left(h_j\right)\right)/T}}{\sum_{i\in\{1,2,\ldots,k\}} e^{\Delta_{h_i}\overline{V}\left(choose\left(h_i\right)\right)/T}} \tag{4.23}$$

Here, h^* is randomly sampled according to a probability distribution $P(h_j)$. We call this approach SoftMaxDHS because it uses the softmax function (commonly used in reinforcement learning and elsewhere in the agents literature, e.g. (Kaelbling, Littman, & Moore, 1996; Sutton & Barto, 1998)) to determine the Boltzmann (or Gibbs) probability distribution $P(h_j)$. Two key properties of this probability distribution are: (1) the probability of sampling h_j increases as $\Delta_{h_j}\overline{V}\left(choose\left(h_j\right)\right)$ increases, fitting with the original definition of DHS and Eq. 4.20, and (2) the probability of each h_j is always

greater than 0 since $\Delta_{h_j}\overline{V}\left(choose(h_j)\right)$ is always finite and thus $e^{\Delta_{h_j}\overline{V}\left(choose(h_j)\right)/\mathrm{T}} > 0$.

Given Eq. 4.23, we find that:

Theorem 4.2: SoftMaxDHS is ϵ-optimal, so long as h_{AEMS2} is one of the heuristics available for selection.

Proof: The original proof by Ross, Pineau, & Chaib-draa (2008) for the ϵ-optimality of AEMS2 contains a theorem (Theorem 2 (Ross, Pineau, & Chaib-draa , 2008)) stating that if the path from the root belief state b_c consisting only of actions with maximal $\overline{Q}$ has a non-zero probability of being expanded in Step 1 of each iteration of Algorithm 4.1, then the algorithm is ϵ-optimal. We know that h_{AEMS2} only selects such paths for expansion due to the $w(b,a)$ component. Thus, the probability of h_{AEMS2} choosing such a path for expansion is 1.0. Moreover, in SoftMaxDHS, we know that the probability of h_{AEMS2} being used to guide policy tree expansion is $P(h_{AEMS2}) > 0$. Hence, in each iteration of Algorithm 4.1, SoftMaxDHS chooses to expand the path from the root belief state b_c consisting only of actions with maximal $\overline{Q}$ with probability $P(h_j) > 0$. Since this probability is non-zero, SoftMaxDHS is ϵ-optimal. ■

Like with DHS-m, the behavior of SoftMaxDHS depends on an internal parameter T. Here, as in other softmax-based algorithms, T defines how sensitive the probability distribution $P(h_j)$ is to the values of $\Delta_{h_j}\overline{V}\left(choose(h_j)\right)$. The smaller the T, the more greedily the distribution favors the heuristic h_j with the greatest $\Delta_{h_j}\overline{V}\left(choose(h_j)\right)$. On the other hand, the larger the T, the closer the distribution approaches a uniform distribution. In practice, the best value of T depends on the

environment (and the range of values of $\overline{V}$), so this parameter would need to be fine-tuned for each environment. We perform such fine-tuning in our experiments to follow.

4.5. Experimental Setup

To evaluate the performance of our solution contributing the LSEM heuristic and DHS algorithm, we conducted an experimental study using several commonly used, well-known POMDP benchmark problems: AUVNavigation (Ong *et al*., 2010), Tag (Pineau, Gordon, & Thrun, 2003), and RockSample (Smith & Simmons, 2004). These benchmarks vary in their complexity and level of uncertainty, as described below. The goals of our study were (1) to demonstrate the problems associated with online POMDP planning in highly uncertain domains, (2) evaluate the ability of our solution (and the variants of DHS discussed in Section 4.4.4) to improve such planning by splitting planning into two stages for addressing the two main types of uncertainty facing the agent (ESU and CRU), and (3) evaluate the ability of our solution to adapt to the environment by studying how well it performs when the agent doesn't face high levels of ESU, contrary to the rationales for its design.

First, in the AUVNavigation benchmark (Ong *et al*., 2010), an autonomous underwater vehicle must navigate through a 3D grid (with size $20 \times 7 \times 4$) to move from an unknown starting location, through a maze of dangerous rocks that could destroy the vehicle, to either of two known goal locations (on the opposite side of the world from the starting location). To reach a goal location, the agent can perform six actions: Stay and not move at all, turn Up, Down, Left, or Right to change its 3D orientation, or move Forward along its current orientation. The agent might change location after every action, even if it doesn't move forward, due to dynamic underwater currents. The agent

can always fully observe its depth and orientation in the grid, but its (x, y) position is hidden from the agent unless it goes to the surface of the water, in which case it incurs a high cost (-50) in return for a perfect observation about its (x, y) location using a GPS sensor. When the agent is not on the surface, it instead receives one of four observations in every other state: Rock if it hits a rock, End if it reaches a goal location, Blind in every other location, and Terminal upon ending execution. Moving incurs increasing costs based on the number of dimensions the agent moves in according to its orientation (-1 for one dimension, -1.44 for two dimensions, -1.73 for three dimensions), hitting a rock incurs a much larger cost (-500) and ends execution on the next step, but reaching a goal location earns a very large reward (+5000) and also ends execution on the next step. The agent's goal is to reach the goal location as fast as possible while minimizing costs for moving around.

Second, in the Tag benchmark (Pineau, Gordon, & Thrun, 2003), a robotic agent moves in a 2D grid (consisting of 29 possible locations) in order to find and tag an opponent robot. The tagger can move in each cardinal direction (North, South, East, West) as well as try and Tag the opponent, which succeeds if both agents are in the same location. Movement is deterministic and incurs a cost of -1, whereas successfully tagging the opponent earns a reward of +10 and ends execution, but an unsuccessful Tag action incurs a cost of -10. The tagger always fully observes its own location, but the opponent's location is hidden from the tagger robot, and the tagger can only receive one of two observations after each action: True if both robots are in the same location, and False in all other states. The opponent, on the other hand, fully observes the tagger robot

and tries to move away from the tagger in each time step. The tagger's goal is to find and tag the opponent as fast as possible.

Finally, in the RockSample benchmark (Smith & Simmons, 2004), a robotic agent must navigate through a $g \times g$ 2D grid containing k rocks. In our study, we consider the popular setting of $g = 7$ and $k = 8$ (e.g., Ross & Chaib-draa, 2007; Ross *et al*., 2008; Silver & Veness, 2010; Somani *et al*., 2013; Zhang & Chen, 2012). The robot always fully observes its current location, but the quality of the k rocks are hidden by the environment's partial observability. The robot is tasked with identifying and sampling rocks with good quality and not sampling rocks with bad quality. To accomplish this goal, the agent can move in each cardinal direction (North, South, East, West), check the quality of each rock (using a separate Check action for each of the k rocks), or Sample the rock in the robot's current location. Execution ends whenever the robot moves off the east side of the grid. Checking the quality of a rock returns an observation about that rock from the set $Z = \{Good, Bad\}$, where the accuracy of the observation depends on the robot's distance from the rock (where farther distances d produce less accurate observations according to accuracy function $acc = 0.5 + 2^{-1-\frac{d}{20}}$). All other actions produce the same observation (Bad). The robot earns a reward for Sampling good rocks of +10, a penalty of Sampling bad rocks of -10, and a reward of +10 for moving off the grid to end execution. Each rock automatically changes state to Bad after it is sampled to prevent the robot from sampling the same rock multiple times. The robot's goal is to sample all (and only) good rocks, then exit the grid as fast as possible.

Comparing these three benchmarks, we note that they differ in their levels of uncertainty, especially ESU, making them an interesting range of environments for

evaluating our solution approach. Specifically, both AUVNavigation and Tag are highly uncertain environments, whereas RockSample has much lower levels of ESU.

In particular, the agent in AUVNavigation faces high levels of ESU because it only receives observations about its (x, y) location in a small number of states (i.e., along the surface of the water), which is necessary knowledge for planning a series of movement actions to reach a goal location. Indeed, the fact that the majority of locations (i.e., non-surface, non-rock, and non-goal locations) produce the same observation (Blind) means that most observations do *not* improve the agent's beliefs about the hidden environment state (including the agent's location), as discussed in Section 4.3. Instead, the agent must plan a lengthy sequence of information gathering actions in order to just discover the (x, y) location (e.g., by turning and moving to the surface of the water) before it can plan actions needed to reach the goal location. Moreover, this sequence of information gathering actions incurs costs for both moving and surfacing, causing such actions fail to maximize the initial upper and lower bounds on cumulative rewards. Therefore, AUVNavigation is a prime example of the highly uncertain environments studied in this research.

Similar to AUVNavigation, Tag is also highly uncertain because the tagger robot rarely knows the location of the opponent (unless they are in the same location), and most states produce the same observation, which prevents the belief updates (Eq. 2.4) from being very informative (c.f., Section 4.3). Thus, Tag might also benefit from splitting planning into two stages to enable the tagger agent to plan to reduce ESU (i.e., the location of the opponent) before reducing CRU. However, as hypothesized in Section

4.3, Tag might also not need special treatment, in spite of high levels of ESU, because the problem is relatively small (as compared below).

Unlike AUVNavigation and Tag, RockSample is *not* highly uncertain because the agent can improve its understanding of the current environment state from any state through the various Check actions. Moreover, the although the accuracy of the observations depends on the distance between the robot and a rock—meaning observations for some states are less accurate than others—the minimal possible accuracy is still pretty high for $g = 7$: $0.5 + 2^{-1-\frac{12}{20}} = 83\%$. Thus, the agent only needs very short sequences of information gathering actions in order to reduce its uncertainty about the quality of each rock, and thus its ESU.

Further comparing these three benchmarks, we note that they also differ greatly in their complexity. First, AUVNavigation is the most complex, containing 13,536 states (describing the vehicle's location, depth, and orientation), 6 actions, and most notably, 144 possible observations. Second, RockSample is moderately complex, containing 12,545 states (describing the robot's location and the quality of the 8 rocks), 13 actions, and only 2 observations. Finally, Tag is the least complex, containing only 870 states (describing the tagger and opponent's locations), 5 actions, and only 2 observations.

To evaluate the ability of our DHS solution (and variants[19]) to perform online POMDP planning in these three benchmarks, we measured success using the cumulative, discounted rewards actually earned by the agent while operating in the environment:

$$\sum_{t=0}^{\infty} \gamma^t r_t \tag{4.24}$$

[19] For DHS-m, we used the setting of $m = 2$ since we considered two heuristics during planning (one per stage). For SoftMaxDHS, we optimized the T parameter per benchmark by first searching in steps of 10, then within a step of 10, using $\tau = 5000$ for AUVNavigation and $\tau = 100$ for Tag and RockSample. This resulted in $T = 0.5, 1000, 2$ for AUVNavigation, Tag, and RockSample, respectively.

as typically used to evaluate POMDP planning, with the common setting of $\gamma = 0.95$. Using this measure, we compared the performance of each of our DHS variants (using both h_{LSEM} and h_{AEMS2} as the two heuristics used for planning) against the state-of-the-art heuristic search online POMDP planning algorithms: AEMS2 (Ross & Chaib-draa, 2007) and FHHOP (Zhang & Chen, 2012) (c.f., Section 4.2.2). For the sake of completeness, we also compared against the state-of-the-art Monte Carlo search algorithms for online POMDP planning: ABDESPOT (Anytime Basic DEterminized Sparse Partially Observable Tree) and ARDESPOT[20] (Anytime Regularized DEterminized Sparse Partially Observable Tree) (Somani *et al*., 2013), which represent the other state-of-the-art algorithms in online POMDP planning. These Monte Carlo algorithms consider very similar information as AEMS2 when guiding online planning, except they use random sampling of state transitions for action sequences both to estimate cumulative rewards and approximate the agent's belief state using a particle filter (i.e., an approximation of the belief state probability distribution using frequentist counting of randomly sampled next states, used to speed up planning in environments with large state spaces). Finally, we also considered an algorithm using only our LSEM heuristic to guide planning to gain insights into the usefulness of this heuristic alone. To ensure fair comparison, all approaches used FIB and Blind (Hauskrecht, 2000) to calculate the upper and lower bounds on leaf belief states $\overline{V}(b_{c+n})$ and $\underline{V}(b_{c+n})$.

For each benchmark, we considered a range of amounts of time allocated for planning τ in order to better understand how well each online planning algorithm handles

[20] For ARDESPOT, we reused the λ regularization parameter suggested by Somani *et al*. in their implementation for Tag ($\lambda = 0.01$) and RockSample ($\lambda = 0.1$), available at http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.DownloadDespot , and found an appropriate value through experimentation for AUVNavigation ($\lambda = 0.1$)

different time constraints in different types of environments (highly uncertain vs. less uncertain, more complex vs. less complex): $\tau = \{5, 10, 50, 100, 500, 1000\}$ ms for Tag and RockSample and $\tau = \{50, 100, 500, 1000, 5000, 10000, 15000, 20000\}$ ms for the more complex and uncertain AUVNavigation. Shorter planning times also inform us how well the agent does at the beginning of each planning step, and longer planning times inform us how well the agent's planning improves with more time allocated for planning. Since our DHS solution with LSEM heuristic was designed to speed up planning in highly uncertain environments, we expected it to produce greater rewards in less planning time in highly uncertain environments AUVNavigation and Tag. If the DHS mechanism (and its variants) indeed chooses an appropriate heuristic based on the agent's current need, we also expected it to perform well in RockSample by simply relying on h_{AEMS2} since h_{LSEM} is unnecessary.

Since we varied the amount of time allocated for planning in each benchmark, we ran all experiments on a fixed computer. This machine contained an Intel i5 (Haswell) 3.4GHz Quad Core processor (using one thread per experiment) with 8 GB of memory (3GB were allocated for planning). Each benchmark and algorithm was implemented in Java. We ran each time constraint and algorithm pair for 1,000 runs using different random seeds (with only 100 runs for the more time consuming AUVNavigation) and report 95% confidence intervals around the average cumulative rewards actually earned by the agent (Eq. 4.24). We allowed each run to execute for up to 200 chosen actions, after which we stopped execution since each problem should be solvable in far fewer steps and runs of longer than 200 steps were not goal directed. To speed up planning, we employed the state-of-the-art MOMDP (Ong *et al*., 2010) representation for each

benchmark POMDP (c.f., Section 4.2.2), with model parameters based on the POMDPX configuration files available online at the Approximate POMDP Planning Toolkit Dataset Repository[21].

4.6. Results

In this section, we present and discuss the results of our experimental study described in Section 4.5. First, we evaluate the results in each individual benchmark problem. Then, we summarize the results across all benchmarks and highlight important discoveries and conclusions.

4.6.1. *AUVNavigation Results*

We begin our results analysis by considering the most complex and highly uncertain environment—AUVNavigation—since this type of environment is exactly what our DHS solution with LSEM was designed to address. We present the results of each online POMDP planning algorithm on this benchmark in Table 4.1.

From these results, we make several important observations. First, we observe that *the state-of-the-art heuristic search algorithms AEMS2 and FHHOP indeed suffered greatly in this highly uncertain environment, unless given large amounts of time for planning* τ. That is, when the agent has less than 10 seconds to plan for each action, the agent earned very minimal rewards close to 0 due to random, non-goal directed behavior (i.e., it did not find value in spending cost for moving forward—either towards information or a goal location—and instead routinely performed random, costless actions until possibly drifting into a rock). Recall that in AUVNavigation, the agent starts from an initial unknown location and must first discover where it is to know how to find a

[21] http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.Repository

Table 4.1: Results on AUVNavigation Benchmark with 95% Confidence Intervals

AUVNavigation
$|S| = 13{,}536\ \ |\mathcal{X}| = 96\ \ |\mathcal{Y}| = 141\ \ |A| = 6\ \ |Z| = 144$

Algorithm	τ (ms)							
	50	100	500	1000	5000	10000	15000	20000
AEMS2	-2.1 ± 1.3	-4.4 ± 6.6	-4.7 ± 6.6	-4.7 ± 6.6	-1.5 ± 0.7	928.4 ± 107.6	927.7 ± 107.7	928.4 ± 107.6
FHHOP	-5.8 ± 6.8	-4.4 ± 6.6	-5.1 ± 6.6	-2.9 ± 1.9	-2.6 ± 1.8	468.2 ± 94.2	871.8 ± 108.1	928.4 ± 107.6
LSEM	248.7 ± 111.4	308.5 ± 119.3	385.8 ± 104.9	414.9 ± 83.1	427.5 ± 109.4	420.4 ± 109.5	420.4 ± 109.5	420.4 ± 109.5
DHS	273.6 ± 121.1	353.0 ± 117.6	526.0 ± 100.8	588.2 ± 103.5	501.8 ± 104.8	572.0 ± 91.9	517.7 ± 102.4	927.4 ± 107.5
DHS-m	165.3 ± 123.1	322.7 ± 126.6	445.4 ± 106.5	588.2 ± 103.5	501.8 ± 104.8	572.0 ± 91.9	927.4 ± 107.5	927.4 ± 107.5
SoftMaxDHS	268.8 ± 123.6	361.1 ± 114.7	545.0 ± 103.4	585.1 ± 108.1	652.2 ± 100.9	652.2 ± 100.9	565.3 ± 102.6	575.2 ± 98.8
ABDESPOT	595.9 ± 107.0	478.7 ± 108.0	300.8 ± 86.5	416.9 ± 90.2	1007.0 ± 83.0	969.5 ± 86.4	878.7 ± 106.3	1001.8 ± 84.6
ARDESPOT	24.7 ± 26.0	48.2 ± 29.5	321.9 ± 76.1	460.1 ± 72.0	922.5 ± 96.6	988.3 ± 86.7	961.2 ± 100.3	965.0 ± 95.7

sequence of actions moving the agent to a goal location. Thus, the agent has a high amount of ESU that needs to be reduced through long sequences of information gathering actions before it can plan actions ultimately maximizing its cumulative rewards. As discussed in Section 4.3, this results in the error bounds on cumulative rewards $e(b)$ being difficult to reduce until ESU is reduced, causing the lack of goal-directed behavior. Instead, when using state-of-the-art heuristic search algorithms, the agent had to plan for a long time in order to find policies that appropriately reduced the agent's ESU so that it could also plan a path from its initial location to a goal location and earn large cumulative rewards.

Next, we compare the performance of our LSEM heuristic alone against the state-of-the-art heuristic search algorithms. Even though LSEM is only designed to guide agent planning in Stage 1 (and does not necessarily reduce CRU in Stage 2), we observe a *significant improvement in agent behavior when planning times were most constrained*

($\tau = 50, 100, 500, 1000, 5000$ ms) *compared to AEMS2 and FHHOP*. Instead of random, non-goal directed behavior, the agent formed and executed plans that not only reduced ESU, but then also led the agent to a goal location, where it earned the only possible positive rewards. This result implies that h_{LSEM} also has some value in Stage 2 of planning in highly uncertain environments. However, the cumulative rewards earned using LSEM alone were not as high as the state-of-the-art algorithms for the least constrained planning times ($\tau \geq 10000$ ms). This imples that *although LSEM can perform somewhat admirably in Stage 2, it cannot completely reduce CRU to the point that such rewards are ultimately optimized*. Hence the need for our DHS solution.

Moving on to our DHS solution variants combining h_{LSEM} for Stage 1 of planning (ESU reduction) and h_{AEMS2} for Stage 2 of planning (CRU reduction), we observe *much better performance when planning time was most constrained* (e.g., $\tau = 50, 100, 500, 1000, 5000$ ms) *compared to AEMS2, FHHOP, and LSEM*. Particularly, we observe that all three variants (DHS, DHS-m, and SoftMaxDHS) achieved positive rewards *at least 200 times faster than the state-of-the art heuristic search algorithms* AEMS2 and FHHOP, implying that *our solution (and its variants) can successfully control planning in highly uncertain environments*. That is, our solution enabled the agent to reduce the necessary types of uncertainty at the right times in order to create plans leading the agent to reach a goal location and earn the only positive reward in the benchmark. Success with planning times as small as 50 ms is rather noteworthy since a successful run requires over 20 actions just to navigate from the initial starting location to the goal location, not counting actions to resolve ESU, which is quite deep given the complexity of this benchmark.

Furthermore, we observe that *DHS always performed better than either AEMS2 or LSEM alone for these constrained planning times ($\tau < 10000$ ms)*. This implies that splitting planning into stages, then using situational-awareness to choose appropriate heuristics for each stage, improves the agent's ability to efficiently and effectively plan policies leading to successful behavior than using either heuristic alone.

Additionally, we also observe that *the performance of DHS (and its variants) generally improved as more and more time was allocated*, as we desire out of an anytime algorithm. Moreover, both DHS and DHS-m also reached the same very high cumulative rewards (over 900) as the state-of-the-art heuristic search online POMDP planning algorithms AEMS2 and FHHOP, although our solution required a little more time to reach such high rewards (15,000 ms for DHS-m and 20,000 ms for DHS vs. 10,000 ms for AEMS2). This is somewhat expected from our theoretical results in Section 4.4.4 (especially the discussion on DHS-m), and in the future we intend to explore additional ways to further speed up the increase in reward accumulation by agents planning with DHS and LSEM.

Finally, comparing the performance of our solution against the state-of-the-art Monte Carlo search online POMDP planning algorithms ABDESPOT and ARDESPOT, we observe mixed results. First, we observe that *our solution and each of its variants (DHS, DHS-m, SoftMaxDHS) outperformed ARDESPOT for each of the most constrained planning times ($\tau = 50, 100, 500, 1000$ ms) and the simpler, non-regularized ABDESPOT for several of the same constraints ($\tau = 500, 1000$)*. Ultimately, each of these solutions achieved similar performance for the greatest amounts of planning, although the ABDESPOT and ARDESPOT approaches reached high levels sooner and

Table 4.2: Results on Tag Benchmark with 95% Confidence Intervals

Tag

$|S| = 870\ \ |\mathcal{X}| = 30\ \ |\mathcal{Y}| = 29\ \ |A| = 5\ \ |Z| = 2$

Algorithm	τ (ms)					
	5	10	50	100	500	1000
AEMS2	-5.78 ± 0.38	-5.70 ± 0.38	-5.44 ± 0.39	-5.73 ± 0.38	-5.50 ± 0.40	-5.49 ± 0.38
FHHOP	-8.17 ± 0.42	-8.26 ± 0.42	-6.53 ± 0.38	-6.46 ± 0.38	-5.95 ± 0.37	-5.90 ± 0.38
LSEM	-48.85 ± 1.60	-48.49 ± 1.62	-49.88 ± 1.61	-49.56 ± 1.58	-43.55 ± 1.29	-41.59 ± 1.21
DHS	-21.56 ± 1.27	-21.68 ± 1.26	-12.08 ± 0.75	-10.25 ± 0.65	-8.84 ± 0.48	-7.03 ± 0.40
DHS-m	-6.03 ± 0.41	-6.20 ± 0.41	-5.90 ± 0.37	-5.54 ± 0.38	-5.87 ± 0.38	-6.06 ± 0.38
SoftMaxDHS	-9.57 ± 0.60	-9.27 ± 0.58	-6.69 ± 0.40	-5.97 ± 0.40	-6.17 ± 0.37	-6.03 ± 0.38
ABDESPOT	-11.65 ± 0.43	-12.27 ± 0.40	-7.22 ± 0.37	-6.51 ± 0.38	-5.77 ± 0.38	-5.92 ± 0.38

achieved slightly greater overall rewards. Given that these Monte Carlo search algorithms were the best of the previously reported online POMDP planning algorithms and operate differently than heuristic search, our approach represents *a new heuristic search algorithm that starts to bridge the gap between Monte Carlo search algorithms and heuristic search algorithms* on such a difficult problem.

4.6.2. *Tag Results*

Next, we analyze the results of our experiments on the Tag benchmark. Recall that Tag is also a highly uncertain environment, since the agent can only observe the location of the opponent it seeks when they are in the same location. However, Tag is also much less complex than AUVNavigation—containing an order of magnitude fewer states and two orders of magnitude fewer observations. We present the results on this benchmark in Table 4.2.

From these results, we first observe that the state-of-the-art heuristic search algorithm AEMS2 performed quite well on this benchmark, achieving both (1) the best performance for most of the time constraints, and (2) quite consistent performance across all time constraints, even performing almost as well with only 5 ms of planning time compared to 1000 ms of planning time. Similarly, the other state-of-the-art heuristic

search algorithm, FHHOP, also performed quite well, especially with $\tau \geq 50$ ms planning time. Thus, we confirm our suspicion that although this benchmark is highly uncertain, it is not complex enough to warrant special solutions to handle ESU and CRU separately.

However, we still observe that our DHS-m solution variant performed almost as well as AEMS2, partially due to its bias to rely on h_{AEMS2} more often than DHS (Eq. 4.22) and partially due to its correct selection of heuristics. Similar in performance to FHHOP, SoftMaxDHS also performed very well with $\tau \geq 50$ ms planning time, in spite of no bias towards relying often on h_{AEMS2}. Thus, both of these ϵ-optimal variants still properly guided planning to performances very close to the state-of-the-art heuristic search algorithms.

On the other hand, our DHS solution *did not perform as well as its ϵ-optimal variants* DHS-m and SoftMaxDHS (although its still *greatly improved its performance with more planning time*, as desired). Looking closer at the results, we note that this is due to the LSEM heuristic actually having a problem caused by a quirk of this benchmark. In particular, the agent's Tag action not only has the ability to earn the agent a large reward (or incur a large cost), but it also identifies whether or not an opponent is in the same location. That is, if the agent performs a Tag action, it will either know with certainty that it shares a location with the opponent (since it receives a large reward and execution ends), or that the opponent cannot be in the agent's current location. As such, this action always reduces the agent's ESU (where the opponent's location is the hidden part of the environment state). Since no other actions reveal as much information about the environment state, belief states following Tag actions maximize h_{LSEM}, even though

they will earn the agent large costs if the opponent is not in the same location as the agent. As a result, the agent will often want to perform Tag actions when using h_{LSEM}, and will subsequently accumulate large costs for wrong Tag actions. Other heuristics such as h_{AEMS2}, on the other hand, will consider the possibility of these large costs and cause the agent to avoid performing Tag actions until it is likely to be in the same location as the opponent. This quirk explains why LSEM alone performed so poorly on Tag, and why DHS also suffered compared to its variants (where DHS-m is biased to perform h_{AEMS2} more often and SoftMaxDHS only stochastically chooses the heuristic considered ideal for the current expected stage of planning). On the other hand, we also observe that DHS did not perform nearly as poorly as LSEM alone (especially as planning time increased), implying that it still adjusted which heuristics were used and when in order to guide planning.

Finally, comparing against the state-of-the-art Monte Carlo search algorithms ABDESPOT and ARDESPOT, we observe that our DHS variants DHS-m and SoftMaxDHS *outperformed* the Monte Carlo search algorithms for the *smallest planning times* ($\tau \leq 50$ ms) and were close in performance for the greater planning times ($\tau \geq 100$ ms). Thus, our heuristic search solution again performed very favorably in comparison to the state-of-the-art Monte Carlo algorithms, and not just other heuristic search algorithms for online POMDP planning.

Table 4.3: Results on RockSample Benchmark with 95% Confidence Intervals

RockSample

$|S| = 12{,}545 \;\; |\mathcal{X}| = 50 \;\; |\mathcal{Y}| = 256 \;\; |A| = 13 \;\; |Z| = 2$

Algorithm	τ (ms)					
	5	10	50	100	500	1000
AEMS2	13.99 ± 0.33	14.24 ± 0.33	18.22 ± 0.39	19.02 ± 0.39	19.48 ± 0.37	20.31 ± 0.41
FHHOP	7.36 ± 0.02	7.41 ± 0.04	18.08 ± 0.38	18.91 ± 0.41	19.32 ± 0.38	20.40 ± 0.40
LSEM	7.35 ± 0.00	7.35 ± 0.00	7.35 ± 0.00	7.35 ± 0.00	7.35 ± 0.00	7.35 ± 0.00
DHS	13.47 ± 0.34	13.84 ± 0.33	18.14 ± 0.39	18.18 ± 0.39	20.16 ± 0.38	20.03 ± 0.42
DHS-m	12.98 ± 0.35	12.71 ± 0.35	18.08 ± 0.39	18.37 ± 0.41	19.19 ± 0.38	20.38 ± 0.40
SoftMaxDHS	13.72 ± 0.33	13.24 ± 0.34	18.18 ± 0.40	18.30 ± 0.39	18.85 ± 0.38	19.99 ± 0.41
ABDESPOT	18.71 ± 0.41	18.83 ± 0.41	19.61 ± 0.43	19.77 ± 0.41	19.79 ± 0.41	20.00 ± 0.41
ARDESPOT	18.72 ± 0.39	18.61 ± 0.41	19.48 ± 0.41	19.32 ± 0.40	19.74 ± 0.41	19.32 ± 0.42

4.6.3. *RockSample Results*

Finally, we analyze the results of our experiments on the RockSample benchmark. Recall that unlike AUVNavigation and Tag, this benchmark is *not* highly uncertain, and thus does not require two stages for planning (as controlled by our solution). We present the results on this benchmark in Table 4.3.

From these results, we first observe that as expected, the state-of-the-art heuristic search algorithms performed quite well. Both AEMS2 and FHHOP increased in performance with more planning time and achieved some of the highest cumulative rewards. As in our other benchmarks, we again observe that FHHOP started off a little lower than AEMS2, but eventually caught up as planning time increased. Thus, state-of-the-art heuristic search algorithms indeed properly addressed planning in this non-highly uncertain environment.

However, we also observe quite good performance from our DHS solution and its variants, in spite of the fact that planning did not require two stages since the environment was not highly uncertain. That is, not only did performance increase as planning time increased (as desired), but each of our variants (DHS, DHS-m, and SoftMaxDHS) generally outperformed state-of-the-art FHHOP for the smallest planning

times ($\tau \leq 10$ ms), and were competitive with both FHHOP and AEMS2 across all planning times. This is noteworthy since LSEM alone generally performed the worst of all solutions (since treating ESU reduction separately was not necessary in RockSample). In other words, each of our DHS solution variants *appropriately relied on the* h_{AEMS2} *heuristic throughout planning, treating nearly all of planning as if the agent were always in Stage 2* (since Stage 1 was not necessary). This is exactly the type of behavior we want to observe in environments that are not highly uncertain, implying that our solution is not only *beneficial in complex, highly uncertain environments* such as AUVNavigation, but is also *safe to use in other types of environments* as well (without suffering significantly worse performance than state-of-the-art AEMS2).

Finally, comparing our solution against the state-of-the-art Monte Carlo search algorithms, we observe that although our solution started with worse performance for the smallest planning time constraints ($\tau \leq 100$ ms), it still achieved comparable performance as planning time increased. Considering also the performance of AEMS2 and FHHOP, we note that on problems such as RockSample, Monte Carlo search algorithms appear to be the most efficient and effective at planning, as previously reported (e.g., Silver & Veness, 2010; Somani *et al*., 2013).

4.6.4. *Discussion*

Considering our results across all three benchmark problems, we now draw the following conclusions. First, *our situationally-aware DHS algorithm indeed improves planning in complex, highly uncertain environments*, as desired. In the AUVNavigation benchmark, this algorithm appropriately adapted the agent's planning based on the currently identified stage in order to select the most appropriate heuristic (novel h_{LSEM} or

h_{AEMS2}) needed to resolve the most pressing type of uncertainty: ESU or CRU. As a result, the agent achieved the greatest cumulative rewards when planning was the most constrained, and therefore also the most difficult, in comparison to the state-of-the-art heuristic search algorithms. It was also competitive with, and sometimes exceeded, the state-of-the-art Monte Carlo search algorithms that were the best previously reported algorithms on this benchmark. Therefore, our solution provides a heuristic search algorithm for online POMDP planning that bridges the performance gap between this type of planning algorithm vs. Monte Carlo search algorithms.

Moreover, our algorithm also demonstrated its ability to properly *identify the appropriate heuristic to use when the environment was* not *highly uncertain*, as in the RockSample benchmark. In RockSample, DHS and its variants appropriately relied on the h_{AEMS2} heuristic, which had similarly great overall performance on this benchmark, and chose not to use the h_{LSEM} very much, which was not needed nor successful in this environment that had easy to resolve ESU. Therefore, we also conclude that the situational-awareness of our DHS algorithm also works in environments where planning does not need to be split into stages, and is therefore safe to use in more environments than those that are highly uncertain.

Finally, the two ϵ-optimal variants of our DHS algorithm—DHS-m and SoftMax—each also performed quite well in the highly uncertain but less complex Tag benchmark, achieving cumulative rewards better than state-of-the-art FHHOP and Monte Carlo search algorithms, as well as close to AEMS2 as the amount of time allotted for planning increased. This result demonstrates that although situational-awareness and multiple stages of planning are less necessary in highly uncertain environments when the

problem isn't very complex (i.e., has small state, action, and observation spaces), our solution again can achieve good performance by relying on the appropriate heuristic at the appropriate times. However, we also discovered in our Tag experiments that our LSEM heuristic has a potential flaw: it does not consider the costs of actions in any way, and thus might try to force ESU reduction at very high costs contrary to the agent's ultimate goals. In the future, we intend to explore variants of LSEM to address this possible weakness. However, as previously described, our DHS-m and SoftMaxDHS variants were able to overcome this weakness by choosing to use the h_{AEMS2} heuristic to guide planning when appropriate.

Of note, in each of our three benchmarks, we observe different results comparing AEMS2 with FHHOP in contrast to those previously reported by Zhang & Chen (2012). Namely, Zhang & Chen reported that FHHOP routinely outperformed AEMS2, including for the times reported in our experimental results. We believe that this is due to a key difference between our experimental setup and theirs: we use a MOMDP representation with each algorithm, instead of only with FHHOP, whereas they considered this representation to be part of their FHHOP solution. Instead, a MOMDP is compatible with each state-of-the-art heuristic search algorithm, so in fairness to each, we used the same representation for all algorithms. In turn, this sped up planning for AEMS2, causing our differences in results.

4.7. Conclusions

In conclusion, in this chapter we studied the problem of online POMDP planning in highly uncertain environments, demonstrating that difficult levels of environment state uncertainty can reduce the ability of state-of-the-art heuristic search algorithms (e.g.,

AEMS2 (Ross & Chaib-draa, 2007), FHHOP (Zhang & Chen, 2012)) to reduce cumulative reward uncertainty, leading to suboptimal planning under limited time constraints. To overcome this problem, we proposed a solution based on splitting planning in such environments into two stages, each addressing a different type of uncertainty. We contributed a novel situationally-aware heuristic selection mechanism designed to identify the agent's current planning stage based on the most pressing type of uncertainty in need of reduction, then use an appropriate heuristic to guide planning based on the current stage. We also contributed a novel heuristic called LSEM that guides the agent to reduce environment state uncertainty during the first stage of planning. We analyzed the theoretical properties of our solution and developed two variants guaranteed to be ϵ-optimal, which is an important property for anytime online POMDP planning algorithms.

We conducted an experimental study comparing the performance of state-of-the-art heuristic search and Monte Carlo search online POMDP planning algorithms against our solution and its variants in three different commonly used POMDP benchmark problems. Using a range of time constraints on planning in each benchmark to understand the performance of planning in different settings, we observed several key results about our solution. First, DHS and its variants successfully produced better plans in the most complex and highly uncertain environment when the agent was most time constrained (finding plans capable of achieving positive rewards over 200 times faster than AEMS2 and FHHOP). Second, DHS and its variants earned some of the highest rewards even in an environment that was not highly uncertain, demonstrating both that (i) our solution appropriately selects heuristics to guide planning based on the agent's

current need, and (ii) that our solution is safe to use in environments that are not highly uncertain. Finally, the ϵ-optimal variants of DHS also achieved good performance in the highly uncertain but least complex environment where multistage planning was not necessary.

In the future, we intend to continue this research along several directions. First, we intend to implement our solution in actual real-world deployments of intelligent agents and multiagent systems within highly uncertain environments to further evaluate its performance. For example, POMDPs have been used to control information gathering in domains such as human-agent interactions (e.g., Boutilier, 2002; Doshi & Roy, 2008; Williams & Young, 2007) and robotics (e.g., Mihaylova *et al*., 2002; Spaan *et al*., 2010), and we suspect our multistage planning could further improve planning in such applications. Second, we intend to produce an improved version of LSEM that considers the costs of actions in order to avoid possible problems like we observed in Tag, where the agent could exchange (unnecessary) high costs for reduced environment state uncertainty. Third, we want to further study variants of DHS to hopefully produce a solution that reaches optimal levels of rewards faster to further complement its ability to find good (albeit suboptimal) rewards quickly. Finally, we want to consider additional types of heuristic functions within a heuristic selection mechanism like DHS to see if our general approach of situationally-aware multistage planning might be useful in other types of complex, challenging environments (and not just highly uncertain environments).

CHAPTER 5 INTELLIGENT INFORMATION SHARING WITH LOCALIZED, NON-STATIONARY PHENOMENA

In this chapter, we present our research on the Information Sharing Problem (c.f., Section 1.3) in the context of large teams where only a small subset of the agents can directly observe local phenomena within the environment. Previous research has demonstrated the challenges of converging to consistent, accurate beliefs throughout the team when observing such localized phenomena. However, sharing is further complicated in non-stationary environments, where changes in the observed phenomena over time require the team to collectively revise their beliefs as the phenomena change.

In this chapter, we first analytically and empirically demonstrate the difficulty inherent in sharing information and revising beliefs over time about localized, non-stationary phenomena, uncovering the inertia-based Institutional Memory Problem. Subsequently, we propose two novel solutions for addressing this problem: (1) a change detection and response algorithm, and (2) a forgetting-based solution. In both solutions, agents reflect on their own knowledge or the knowledge shared by neighbors, the deliberatively decide how to incorporate such information to improve their knowledge updates and information gathering. We test our solutions under several network structures and sequences of non-stationary phenomena to verify the efficacy of our approaches and evaluate their robustness in the presence of faulty and/or malicious agents injecting incorrect information into the team.

Please note that this chapter represents an extended version of a workshop paper presented at the 6th International Workshop on Emergent Agent Intelligence (WEIN 2014) alongside the AAMAS 2014 conference (Eck & Soh, 2014a) in May 2014.

5.1. Introduction

Real-world environments contain complex phenomena that are increasingly observed by computational devices and systems, often to enhance human knowledge and/or provide real-time support for some task. For example, sensors networks and robot teams are employed for area surveillance (e.g., Padhy *et al*., 2006; Pavon *et al*., 2007; Spaan, Veiga, & Lima, 2010), autonomous robots are used to discover victims of disasters in search and rescue applications (e.g., Calisi *et al*., 2007), and human relationships and preferences are tracked in social networking systems (e.g., Yin *et al*., 2011).

In many of these environments, the observed phenomena are very **localized**, such as detected events (e.g., fires) in a specific area, victims trapped in particular buildings, or individual user's preferences. Although there might be many sensing units within the system, only a few sensing units are capable of directly observing such local phenomena, limiting the ability of the system to gather information *en mass*. Furthermore, the phenomena are also often **non-stationary** and change *dynamically* over time. Thus, information gathering by sensing units becomes outdated and must be revised frequently to adapt with the changing phenomena.

To address these challenging phenomena properties, improve the quality of gathered information, and accurately maintain up-to-date beliefs about the observed phenomena, intelligent software and hardware agents can be employed to control sensing units. Such intelligent agents are capable of exhibiting social behavior by sharing information with one another, helping overcome the localization problem in real-world applications. Agents can also provide both goal-directed behavior to accomplish system

goals, as well as reactive behavior to adapt system performance in unexpected situations (Wooldridge, 1999). In this manner, intelligent agents can reason about the sensing performed by the system in order to optimize or improve the information gathered (e.g., Padhy *et al*., 2006; Spaan, Veiga, & Lima, 2010). Altogether, agents can improve the robustness, scalability, effectiveness, and efficiency of observational systems.

Prior research has studied both (1) information sharing between cooperative agents (e.g., Glinton, Scerri, & Sycara, 2009; 2010; 2011; Pryymak, Rogers, & Jennings, 2012), and (2) detecting and adapting to changes in non-stationary information gathered by individual agents (e.g., Widmer & Kubat, 1996). However, little work has considered these two components of agent-based sensing *in combination*. Both are vital to sensing localized, non-stationary phenomena in real-world environments, but at the same time, localization and non-stationarity together make both information sharing and change detection more challenging. Therefore, it is important to study both components of sensing together to understand their relationship to the two aforementioned phenomena properties.

In this chapter, we begin to fill this gap in the literature by considering the impact of both localization and non-stationarity in observed phenomena on information sharing and change detection within teams of sensing agents. In particular, we start with a known model for information sharing: **large team information sharing** (LTIS) (e.g., Glinton, Scerri, & Sycara, 2009; 2010; 2011; Pryymak, Rogers, & Jennings, 2012), a formalized model where many agents work together but only a small subset of the agents can directly observe any particular phenomena. This model was chosen as a starting point due its ability to handle the localization property and its growing popularity in the agent

literature. To this model, we then add non-stationarity and study the effects of these challenging properties together to develop new solutions for handling both properties simultaneously.

We contribute (1) a formalization of non-stationary phenomena within the LTIS model, alongside localization; (2) an analysis of the difficulty of non-stationarity during belief updates using information shared by the few local agents capable of observing the phenomena; (3) two distinct solutions for overcoming the challenges of non-stationarity and localization: (i) cooperative change detection and response in local neighborhoods, and (ii) individually forgetting outdated information; (4) empirical studies investigating the impact of localized, non-stationary phenomena on large teams of agents controlling sensing units, as well as the effect of using our solutions for adapting to such phenomena; and (5) a discussion of the strengths and weaknesses of our solutions and their appropriateness in different environments.

5.2. LTIS

5.2.1. LTIS Model

We first present the formalized LTIS model (Glinton, Scerri, & Sycara, 2009; 2010; 2011; Pryymak, Rogers, & Jennings, 2012) that serves as the foundation for our solutions. In LTIS, a large set of agents A (e.g., $A \geq 1000$) work together as a team to collect information about some environment phenomena. However, only a small subset $S \subset A$ (with $|S| \ll |A|$) of the agents have sensors that can directly observe a phenomenon. For simplicity, agents represent a phenomenon as a binary fact $F \in \{True, False\}$, although the model can be easily extended to a greater number of values (Pryymak, Rogers, & Jennings, 2012). Each sensor returns binary observations ob

describing the current value of the phenomenon. The sensors are imperfect and only return correct ob with accuracy probability r. For agents with sensors, these observations are used to revise the agent's belief about the correct value of F. However, since the team has limited sensors that can observe the particular phenomenon, the agents must share information to revise the other agents' beliefs. Because the team is so large, agents can only communicate with nearby neighbors. Each neighborhood is relatively very small (compared to the total number of agents), with average size d.

A common set of solution techniques have been adopted for LTIS (Glinton, Scerri, & Sycara, 2009; 2010; 2011; Pryymak, Rogers, & Jennings, 2012). First, agents only communicate summarized information representing their current belief about F, instead of forwarding each individual observation from the sensors. These summarized beliefs are called opinions (denoted by op, described below). This practice (1) reduces the amount of potentially costly communication, (2) minimizes the impact of over-counting information, since each agent could repeatedly receive the same forwarded observation from multiple neighbors, and (3) hides raw observations which could be sensitive or include private information (e.g., enemies in the surveilled area, user purchasing habits) (Glinton, Scerri, & Sycara, 2010).

Given uncertain facts, beliefs are represented by a probability distribution describing the likelihood that F is either $True$ or $False$. Agents start with an initial uncertain belief that any value is equally likely, then Bayesian updating incorporates new information o (an observation ob from a sensor, or an opinion op from a neighbor):

$$b' = \frac{cp(o) \cdot b}{cp(o) \cdot b + (1 - cp(o)) \cdot (1 - b)} \tag{5.1}$$

where b is the probability that F is $True$ (so $(1-b)$ is the probability it is $False$), b' is the updated belief, and $cp(o)$ is the conditional probability that F is $True$ given the new information. Here, cp weighs newly received information o, and its value depends on the value and source of o:

$$cp(o) = \begin{cases} r & \text{if } o = True \wedge o \text{ an observation } ob \\ 1-r & \text{if } o = False \wedge o \text{ an observation } ob \\ m_i & \text{if } o = True \wedge o \text{ an opinion } op \\ 1-m_i & \text{if } o = False \wedge o \text{ an opinion } op \end{cases} \quad (5.2)$$

For observations ob, the weight depends on sensor accuracy r, whereas for opinions op, the weight depends on m_j, the likelihood that a_j's neighbors share correct opinions.

Because beliefs are uncertain, agents only share information when they become *confident* that F is either $True$ or not from received information. In particular, a confidence threshold $\sigma > 0.5$ discretizes beliefs into confident opinions:

$$op = \begin{cases} True & \text{if } b > \sigma \\ False & \text{if } b < 1-\sigma \\ Unc & \text{else} \end{cases} \quad (5.3)$$

where Unc denotes an unconfident opinion that is never communicated but noted by the agent when evaluating its belief. We illustrate this discretization in Figure 5.1 in Section 5.3.

5.2.2. Prior LTIS Research

Prior LTIS research has primarily focused on two aspects: (1) identifying important emergent behaviors during information sharing within large teams, and (2) developing distributed algorithms to achieve desired emergent behavior.

Using branching process theory, Glinton, Scerri, & Sycara (2010) developed an analytical model predicting that different settings of the cp information weighting

parameter (specifically m_j for weighting opinions from neighbors) can result in three phases of emergent behavior: (1) unstable dynamics, where too much weight causes frequent avalanches of sharing between agents, resulting in oscillating beliefs, (2) stable dynamics, where too little weight results in infrequent belief updates and few confident beliefs, and (3) scale invariant dynamics, where the optimal amount of weight permits enough sharing to propagate beliefs throughout the team without causing oscillation. Later, they (2011) discovered that LTIS was vulnerable when incorrect information was received (either from benign error or malicious injection by an attacker) and an agent's belief was near the confidence threshold σ.

Prior research has also focused on developing distributed algorithms for controlling information sharing by adapting the weight (i.e., m_j) placed in shared opinions in order to achieve desirable properties. Glinton, Scerri, & Sycara (2010) exploited their model to produce an algorithm (DACOR) that controls avalanches within an agent's local neighborhood to globally achieve scale invariant dynamics. Later, Pryymak, Rogers, & Jennings (2012) developed an algorithm (AAT) requiring no additional communication to improve belief convergence.

In this chapter, we contribute to both avenues of research on LTIS. First, we study the emergent behavior caused by including non-stationarity in the LTIS model, through which we describe analytically the impact of this property on agent information sharing. Second, we develop novel distributed solutions for adapting information sharing and belief updates to handle non-stationarity. We also evaluate these solutions empirically using different settings of teams likely to occur in real-world applications (e.g., different network structures connecting agents, and the presence of malicious or

faulty agents as previously studied (Glinton, Scerri, & Sycara, 2011)) to demonstrate the advantages and disadvantages of each approach.

5.3. Non-Stationary Phenomena

As described previously, the LTIS model is useful for addressing the challenging localization property in observed environmental phenomena because it explicitly considers the reality that only a small subset of the agents can make direct observations. In this section, we extend the LTIS model to also include a second important property of many observed phenomena: **non-stationarity**.

Recall that non-stationarity is caused by dynamic environments that result in changes to the phenomena of interest as agents perform observations (e.g., events occurring in areas of interest, additional buildings collapsing after a disaster trapping new victims, changing human user preferences). To handle non-stationarity, agents must not only be capable of determining the *initial* value of a phenomenon (equivalent to forming beliefs about *stationary* phenomena in *static* environments as previously studied with LTIS), but agents must also be capable of properly *adapting* their beliefs over time as a phenomenon changes values.

5.3.1. Modeling Non-Stationarity in LTIS

To model non-stationary phenomena in LTIS, we extend the existing model by adding a time component to the relevant factors in order to reflect changes to the phenomena over time. This approach produces the following changes.

First, we discretize time into different intervals, represented by $t \in \mathbb{Z}^+$. One time interval represents the amount of time required for a sensor to produce an observation and for an agent to transmit an opinion to one of its neighbors. Second, we redefine a fact

from a constant F to a time-dependent sequence $F(t)$ expressing the phenomenon's changing value at each elapsed time interval. For example, a fact might be (1) periodic and switch values every Δt ticks, (2) random and switch values with differing durations, or (3) simply switch values once. Third, observations ob and opinions op are time-stamped with the time t when they were observed or shared. Finally, to reflect changing fact values over time in the agents' beliefs, probabilistic beliefs are also extended to time-dependent sequences $b(t)$. Of note: since an agent can receive one or more opinions from its neighbors and also an observation from a sensor in the same time interval t, a chain of several belief updates b' can occur for $b(t)$ \$. Thus, the agent might need to incorporate multiple updates from different sources in the same time interval.

5.3.2. Analyzing the Effect of Non-Stationarity

Forming consistent, accurate beliefs about *non-stationary* phenomena is a much more challenging problem than observing *stationary* phenomena because of the amount of information required to correctly revise agents' beliefs after a phenomenon change. To illustrate (without loss of generality), consider a simple phenomenon $F_1(t)$ that is initially $True$, then changes to $False$ at $t = 1001$. Observing this phenomenon results in updates to an agent's beliefs over time illustrated in Figure 5.1 as (a) a continuous probability $0 \leq b \leq 1$, and (b) a discrete opinion $op \in \{False, Unc, True\}$ (Eq. 5.3).

Here, the agent begins with pure uncertainty $b(0) = 0.5$ and must update its belief to $b(t) \geq \sigma$ (recall $\sigma > 0.5$) to achieve a correct opinion of $True$. This requires a belief change of only $\Delta b_1 = \sigma - 0.5$, denoted by (*) in Figure 5.1.

After the non-stationary phenomenon changes values, the agent must receive a sequence of new information to revise its beliefs from $b(t) \geq \sigma - 0.5$ to a later

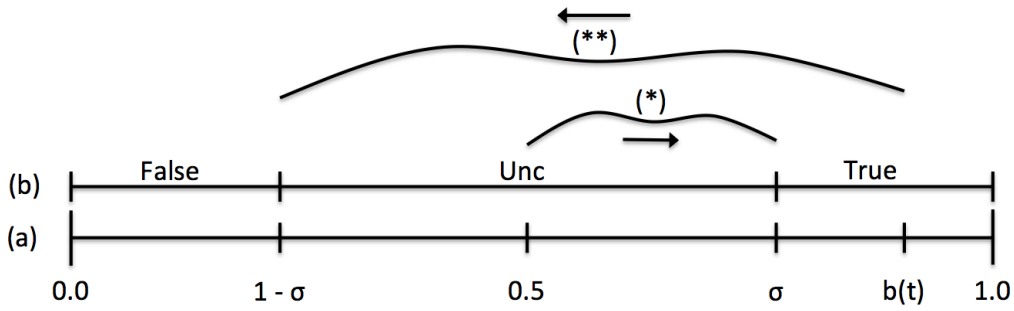

Figure 5.1: Agent Belief Updates
Note: (*) distance to reach initial *True* belief, (**) distance to reach later *False* belief

$b(t') \leq 1 - \sigma < 0.5$. This requires a belief change of $\Delta b_2 \geq 2(\sigma - 0.5)$, denoted by (**). Since $2\Delta b_1 = 2(\sigma - 0.5) \leq \Delta b_2$, we find that properly revising beliefs for non-stationary phenomena requires *at least* twice as much belief change as observing stationary phenomena, and subsequently, twice as much observed and shared information. This requirement holds for *any* change in a phenomenon value, not just in the example used here.

Unfortunately, choosing a weight placed in shared information cannot overcome this problem, as used previously to control the flow of information through the team to achieve consistent, accurate beliefs (Glinton, Scerri, & Sycara, 2010; Pryymak, Rogers, & Jennings, 2012). Instead, the above problem arises *regardless* of the weight selected. That is, given the belief update rule (Eq. 5.1) and any chosen value for $cp(o)$, two updates with opposing information simply cancel each other out. This is the underlying reason why an agent needs twice as much information to revise its belief (than it takes to arrive at an initial confident belief), as described in the previous paragraph. This result implies that controlling information sharing by selecting a weight for new information (namely m_j for shared opinions op) as studied previously for LTIS does not address the challenges posed by non-stationarity. Instead, a different type of solution for guiding agent information sharing and belief updates is necessary. We propose two such

solutions in Sections 5.4 and 5.5 that exploit different ways of closing the gap between (*) and (**) (from Figure 5.1) in order to speed up belief convergence after a change in the non-stationary phenomenon.

Furthermore, we note that the distances (*) and (**) (in Figure 5.1) also result in agents being *less likely to share opinions* from each belief update after the phenomenon has changed values than they would with stationary phenomenon. Here, the team suffers from an inertia problem, which we call the:

> **Institutional Memory Problem**: too much information needs to be received by an agent to cause the agent to also share new opinions, resulting in the team becoming stuck with outdated beliefs that do not change even when new information is observed.

Specifically, recall that agents only share information with neighbors when they cross a confidence threshold $b' \geq \sigma$ or $b' \leq 1 - \sigma$. Since more updates are required to reach a threshold after a phenomenon value change, each individual belief update is less likely to result in sharing a new opinion. Therefore, agents actually share *fewer* opinions with one another. Unfortunately, this is *opposite* of what the agents need in order to adapt to the non-stationary phenomenon since they actually need *more* updates to reach a new accurate belief, causing agents to fail to adapt and either become stuck with (1) outdated beliefs or (2) uncertainty.

The Institutional Memory Problem should not to be confused with the stable dynamics emergent behavior discovered by Glinton, Scerri, & Sycara (2010). In their work studying stationary environments, insufficient information is exchanged due to too little weight placed on new information, resulting in uncertain beliefs. In our work, an inability to overcome previous confident beliefs limits information exchange. To demonstrate that our problem is not caused by the weight chosen for incorporating new

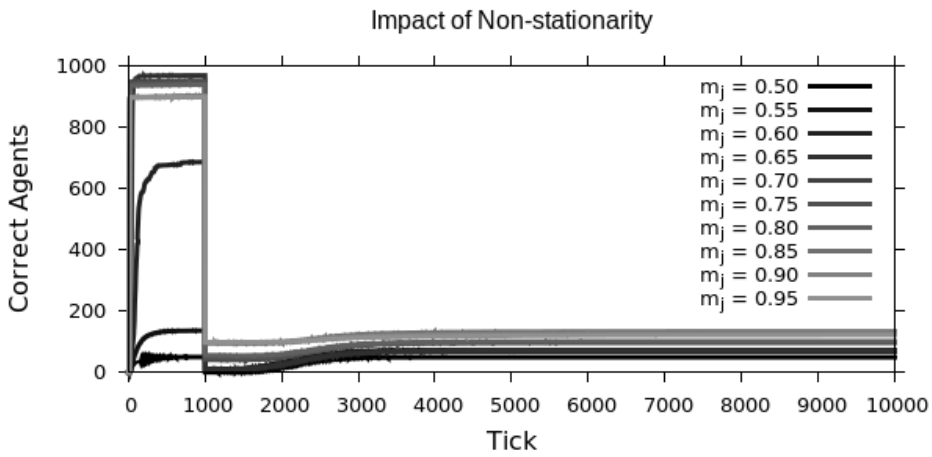

Figure 5.2: Impact of Non-Stationarity

information, Figure 5.2 presents the results of an empirical study using a team of agents observing the aforementioned simple phenomenon $F_1(t)$ (using the Random Network parameters given in Section 5.6).

Here, we measure agent performance as the average number of agents (out of $|A| = 1000$) achieving accurate beliefs over time while the non-stationary phenomenon changed values. We varied the weight for new information from neighbors and confirm that than no ideal weight exists for *non-stationary* phenomena, as opposed to the existence of an ideal weight for *stationary* phenomena (Glinton, Scerri, & Sycara, 2010). Instead, although the team could converge to consistent, accurate beliefs for the *initial* value of the non-stationary phenomenon (identical to stationary phenomena), a *much smaller* number of agents correctly revised their beliefs *over time*. Indeed, the majority of agents was unable to overcome inertia and simply retained the initial phenomenon value in their beliefs. As expected, this occurred regardless of the weight for shared information. Since appropriately choosing a weight for new information is thus not a viable solution for handling non-stationarity (as previously studied for stationarity), we instead require a new type of solution.

Overall, we make the following observations about the relationship between the two properties. First, localization magnifies the impacts of non-stationarity by limiting the flow of information *into* the team by restricting observations about changing phenomenon values necessary to update beliefs over time. Second, non-stationarity magnifies the impacts of localization by limiting the flow of information *within* the team by restricting shared opinions also necessary to update beliefs over time. Therefore, these two challenging properties unfortunately work together *adversely*.

5.4. Change Detection and Response

Similar to prior algorithms for LTIS, our first solution relies on cooperative agents making simple yet effective local decisions within neighborhoods to achieve desired emergent behavior (i.e., properly adapting agent beliefs over time to non-stationary phenomena). Here, we develop an approach for explicitly detecting and responding to non-stationarity.

Strategy. Our strategy is to convert the problem of handling non-stationarity to one closer to forming beliefs about (simpler) stationary phenomena. We start with the insight that if the team were able to detect when a phenomenon changes values, then the agents could treat a new value *independent* of the previous value – that is, as a separate stationary phenomenon and a separate instance of the original stationary LTIS problem. In which case, each agent would need less information to revise its beliefs after a phenomenon change, having instead only to change beliefs from pure uncertainty to a new confident belief (Δb_1), as opposed to moving from one confident belief to its opposite ($\Delta b_2 \geq \Delta b_1$). In turn, this behavior would mitigate the Institutional Memory

Problem by reducing inertia and subsequently increase the team's convergence to consistent, accurate beliefs.

To detect changes to a non-stationary phenomenon, we actually *exploit* the cause of the inertia property of the Institutional Memory Problem identified in the previous section. Specifically, considering how much information is needed to revise an agent's belief (i.e., $\Delta b_2 \geq 2\Delta b_1$, illustrated by (**) in Figure 5.1) causing the inertia, we note that any particular neighbor is very unlikely to share a new opinion that conflicts with the most recent opinion that it previously shared without an actual change in the phenomenon. For instance, in our prior example (Figure 5.1), sharing a new $False$ opinion (after previously sharing $True$) indicates to an agent's neighbors that it received much new information reflecting a phenomenon change. In which case, the new opinion is highly likely to be accurate since the likelihood of receiving such a large chain of information that is instead incorrect would be small. Therefore, changed opinions by neighbors provide more information than just new opinions, but also indicators signaling that the phenomenon indeed likely changed values, which other agents can exploit to overcome their inertia.

After detecting a phenomenon change by receiving a newly conflicting opinion from a neighbor, an agent responds as follows (detailed in Algorithm 5.1). First, the agent receiving a newly conflicting opinion resets its own belief to pure uncertainty ($b(t) = 0.5$), starting a new, fresh belief about the phenomenon under observation. Thus, this agent is now closer to a new correct opinion than any formerly confident belief about the previous value of the phenomenon, without having had to receive as much

```
detectAndRespond(op)
if op.value ≠ lastOpinion(op.sender).value then
    rand ~ U(0, 1)
    if rand ≤ σ then
        b ← 0.5
    end
    sendDetectedChangeAlert()
end
updateBelief(op)
```

Algorithm 5.1: Change Detection and Response (CD & R) Algorithm

information as its neighbor. Next, the receiving agent broadcasts its detection (i.e., sendDetectedChangeAlert()) to its other neighbors that are farther away from sensors and thus less likely to have already detected a change as information propagates, encouraging them to also reset their beliefs. Afterwards, it updates its belief using the information in the shared opinion (Eq. 5.1).

This reaction behavior simultaneously (1) puts agents in a position to quickly revise their beliefs after a detected change by moving away from previously confident beliefs before a belief update, and (2) spreads the detection of phenomenon changes locally within the team to speed up convergence to accurately revised beliefs without requiring all agents to receive a large chain of information to revise their beliefs.

Addressing Concerns. However, we must be careful to avoid incorrectly detecting phenomenon changes, or else the agents' beliefs could oscillate (similar to unstable team dynamics (Glinton, Scerri, & Sycara, 2010)). That is, if a neighbor shares an incorrect new opinion conflicting with past opinions, then a false change would be detected and agents would unnecessarily reset their beliefs and move away from correctly confident beliefs.

Our solution mitigates this concern in three targeted ways. First, agents only reset their beliefs with likelihood σ, reflecting the *same uncertainty* the sharing neighbor has in

its opinion (Eq. 5.3). Second, our solution only *locally* reacts within two[22] network hops from the agent that initially changed opinions, minimizing the impact of false detection on the entire team. Recall that the team's average connectivity d is assumed to be rather small (relative to the size of the team), so these are very local behaviors. Finally, even if an agent incorrectly resets its beliefs, it *only* changes its opinion to Unc and does not fully adopt the neighbor's incorrect information. Thus, the agent's belief is just as close to the correct belief as it is to the neighbor's shared incorrect belief, and the agent can re-converge to the correct belief with new information just as easily as it would converge to the incorrect belief that triggered the reset in the first place.

5.5. Forgetting Outdated Beliefs

Our second solution also relies on agents to exhibit local behaviors to adapt their beliefs over time to non-stationary phenomena. However, unlike our first solution, it is even more localized since each agent adapts *independently* of its neighbors, lessening the reliance of agents on one another. Specifically, we develop a solution employing belief decay to enable agents to forget outdated beliefs and independently and quickly adapt to changes to non-stationary phenomena.

The goals behind this solution design are that it should (1) produce *faster adaptation* to non-stationary phenomena since agents do not need to wait for conflicting opinions from neighbors to begin adaptation, and (2) be *more robust* in environments with potentially faulty or malicious agents (Glinton, Scerri, & Sycara, 2011) since it doesn't rely on neighbors for change detection.

[22] Detection is only propagated to the neighbors of the detecting agent, which is itself a neighbor of the changed agent

Strategy. This solution is based on the natural assumption that if an agent has not received information for a while, its beliefs are less likely to reflect the current value of non-stationary phenomena since each phenomenon's value changes over time. Thus, the agent's beliefs should become less confident the longer time has elapsed since the agent last received new information and updated its beliefs. Then, the agent would be more likely to (1) reach a confidence threshold opposing its most recent opinion after a belief update in order to form a new correct belief, and (2) propagate new opinions throughout the team, enabling other agents to also correctly revise their beliefs and avoid inertia and the Institutional Memory Problem.

To appropriately adapt agent uncertainty over time, we propose a solution based on belief decay, where each agent *forgets* older beliefs the longer time passes between belief updates. Belief decay has been previously used to describe the behavior of human knowledge and memory in the cognitive science literature (e.g., Murdock, 1993), as well as for related problems in artificial intelligence, such as situational awareness (e.g., Hoogendoorn, van Lambalgen, & Treur, 2011) and information foraging with fewer agents that each directly observe the environment (e.g., Reitter & Lebiere, 2012). However, while this approach has been used in other domains, this research is the first application of belief decay to information gathering problems with localized phenomena such as LTIS, so its benefits are unclear *a priori*. We expect that such an approach is especially strategic for LTIS because each agent (1) adjusts its beliefs independent of its neighbors, reducing the agent's reliance on its neighbors to adapt to changes, and (2) can control the rate of decay, useful for adapting to various frequencies of change in non-stationary phenomenon.

For this solution, we propose adding the following rule to each belief update when an agent receives new information before incorporating the new information using Eq. 5.1:

$$b'(t) = 0.5 + (b(t) - 0.5)\lambda^{\delta} \quad (5.4)$$

where δ represents the amount of time elapsed since the agent's last belief update, and $\lambda \in (0,1)$ is a parameter controlling how quickly the agent's belief decays over time: smaller λ causes faster decay, whereas larger λ causes slower changing beliefs. Thus, by choosing an appropriate λ, an agent can adjust how quickly it forgets old information and reacts to phenomenon changes (unlike our first solution).

Using Eq. 5.4, an agent's belief always decays towards pure uncertainty ($b = 0.5$), and the amount of decay is proportional to the amount of time since its last belief update. Thus, the agent moves towards the best position to form a new belief after a phenomenon value change, and it requires less evidence of change (avoiding inertia) the longer it has been since an update when it is more likely that the phenomenon indeed changed values. Afterwards, performing updates with Eq. 5.1 incorporates new information into the time-adjusted belief, allowing the agent to potentially cross a confidence threshold so that it can share a new opinion.

Another way of looking at Eq. 5.4 is *time-dependent information weighting*. That is, Eq. 5.4 weights older information (already incorporated in the agent's belief) down towards uncertainty before incorporating new information (Eq. 5.1), and the amount to down-weight is proportional to the amount of time since the older information was received.

Addressing Concerns. However, we want to ensure that belief decay does not cause agents to become uncertain if the phenomenon has not actually changed for a while, which would lead the team to fail to maintain accurate beliefs.

To mitigate this concern and avoid unnecessary mass uncertainty, we propose only decaying beliefs when *new information is received* instead of *every tick*. Recall that most agents infrequently receive information: only when new information is available, meaning only when there is actual evidence that the phenomenon might have changed. Delaying belief decay until receipt of new information *allows the agent to (cautiously) retain its prior beliefs when it has no evidence causing it to believe the phenomenon has changed*. Decaying every tick (even with a smaller decay rate) would instead constantly push agents towards uncertainty, even if the phenomenon has not changed values, as illustrated in Figure 5.3. Thus, agents would spend more time with uncertain beliefs, making it difficult for agents to *maintain* confident beliefs, similar to the stable dynamics problem observed by Glinton, Scerri, & Sycara (2010) where too little weight in new information causes the team to remain uncertain over time.

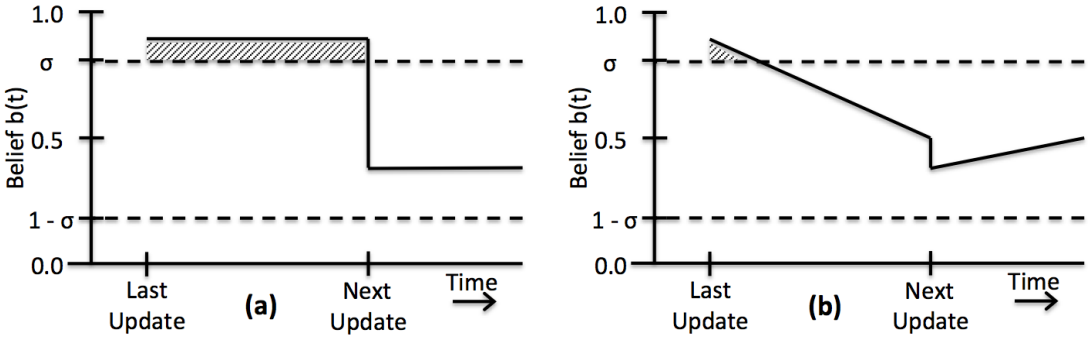

Figure 5.3: Example of Performing Belief Decay (a) Only Upon Receipt of Information vs. (b) Every Tick
Note: Shaded area above σ line indicates accumulated time with a confident belief, which is much greater for (a) than (b)

5.6. Experimental Setup

To better understand how our solutions address the challenges posed by localized, non-stationary phenomena in multiagent systems, we conducted an empirical study to evaluate the performance of our solutions in different scenarios modeling those found in different real-world applications of multiagent sensing. Our goals were to (1) determine whether our algorithms improve the ability of the team to converge to consistent, accurate beliefs about localized, non-stationary phenomena, and (2) evaluate the robustness of our algorithms in the presence of malicious and/or faulty agents that share incorrect information. Within each goal, we also consider how the network structure of the team (dependent on the application and domain) impacts performance.

First, we consider two different types of phenomenon value sequences, representing different types of phenomena: (1) a **periodic sequence** that is initially $True$, then alternates for 10 total values of equal length ($\Delta t = 1000$), and (2) **random sequences** that alternate values 10 times with random lengths (chosen uniformly). The first type of sequence represents equally challenging phenomena values to observe (since each are the same duration), whereas the second type represents less regular phenomena of greater difficulty more likely to be present in real-world applications. In either case, each sequence has a total length of 10,000 simulation ticks.

Second, we also consider the presence of faulty and/or malicious agents that share incorrect opinions every time they cross the σ or $1-\sigma$ threshold and reach a new confident opinion. We vary the number of faulty and/or malicious agents in order to evaluate the robustness of our solutions (which has been demonstrated to be a concern even for stationary phenomena (Glinton, Scerri, & Sycara, 2011)). We also intentionally

choose the agents with the **highest connectivity** to be faulty and/or malicious, which represents a worst case scenario since these agents have most influence over their peers.

Finally, we vary the network structure of the team of agents according to different types of networks present in real-world applications of multiagent sensing, including: (1) **Random networks** (RN), where connections between agents are randomly determined, such as in ad hoc sensor networks, (2) **Small world networks** (SWN), where agents are clustered in large, important subgroups, such as surveillance applications, and (3) **Scale-free networks** (SFN), where connectivity follows a power-law distribution (i.e., a few agents are connected to many neighbors, whereas many agents have small connectivity), such as social networks or the Internet.

To create these networks, we use the Erdos-Renyi (Erdos & Renyi, 1960), Watts-Strogatz (rewire $p=0.5$) (Watts & Strogatz, 1998), and Barabasi-Albert preferential attachment (Barabasi & Albert, 1999) models, respectively. For each network, we use the standard setting from prior studies (e.g., Glinton, Scerri, & Sycara, 2010; Pryymak, Rogers, & Jennings, 2012): the number of agents $|A| = 1000$, the number of sensors $|S| = 0.05|A| = 50$, sensor accuracy $r = 0.55$, average neighborhood size $d = 8$, and confidence threshold $\sigma = 0.8$. Unless specified, we default to the optimal weight for shared opinions given the other parameters: $m_j = 0.63\ \forall a_j \in A$ (Glinton, Scerri, & Sycara, 2010).

To evaluate our solutions, we use two measures of agent performance. First, we consider the average number of phenomena values about which the team collectively forms correct beliefs, represented by N_{800}. That is, following tradition (e.g., Glinton, Scerri, & Sycara, 2010), we consider a team's belief correct if 80% of the agents

$(0.8|A| = 800)$ form a correct, confident belief at the same time before the phenomenon changes values again. This measures how well the team *as a whole* accomplishes its goal. Second, we also consider the average number of agents holding each of the three types of discrete beliefs: correct (C) and incorrect (I) confident beliefs and unconfident beliefs (U). This measure further illuminates how the *individual* beliefs held by agents change over time as they adapt to changing phenomenon values.

With these measures, we compare our two solutions–(1) change detection and response, and (2) forgetting outdated beliefs–for handling localized, non-stationary phenomena. As a baseline, we also compare against agents that know *a priori* the ideal weight for new information, finding which is the goal of prior algorithms for stationary phenomena (e.g., DACOR (Glinton, Scerri, & Sycara, 2010) and AAT (Pryymak, Rogers, & Jennings, 2012)). Thus, our baseline results represent an upper-bound on prior algorithm performance.

5.7. Results

Performance. We first evaluate the general performance of our two solutions for sharing information about localized, non-stationary phenomena within multiagent systems. We present the results of this analysis in Table 5.1, which reports the measures of team performance (N_{800}, C, I, U) for each of the three network types and two types of sequences of non-stationary phenomena. Please note that these results represent the *best performance* of each algorithm type: using the ideal λ value for our forgetting-based solution (found by varying $\lambda \in [0.9,1.0)$ in 0.01 increments) and the ideal m_j value for the baseline and change detection and response solutions (found by varying $m_j \in$

Table 5.1: Comparison of Solutions with Different Phenomenon and Networks with 95% Confidence Intervals

		Periodic Sequence			*Random Sequence*		
	Algorithm	**RN**	**SWN**	**SFN**	**RN**	**SWN**	**SFN**
N_{800}	**Baseline**	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	4.97 ± 0.06	5 ± 0.00	4.99 ± 0.02
	Change Detection	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	7.62 ± 0.28	7.60 ± 0.27	7.40 ± 0.29
	Forgetting	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	9.38 ± 0.13	9.24 ± 0.15	9.44 ± 0.13
C	**Baseline**	479.39 ± 0.91	485.62 ± 0.91	492.37 ± 0.92	498.86 ± 0.91	495.67 ± 0.91	496.54 ± 0.91
	Change Detection	537.23 ± 0.88	546.07 ± 0.91	540.20 ± 0.84	564.16 ± 0.87	576.86 ± 0.90	558.25 ± 0.84
	Forgetting	731.76 ± 0.62	755.59 ± 0.66	642.86 ± 0.56	737.91 ± 0.61	767.63 ± 0.64	652.62 ± 0.54
I	**Baseline**	481.03 ± 0.91	487.11 ± 0.91	492.64 ± 0.92	479.61 ± 0.91	476.31 ± 0.91	486.01 ± 0.91
	Change Detection	337.67 ± 0.85	356.23 ± 0.88	314.44 ± 0.82	324.47 ± 0.85	339.74 ± 0.88	309.61 ± 0.83
	Forgetting	89.65 ± 0.47	94.66 ± 0.49	97.80 ± 0.50	85.60 ± 0.47	85.20 ± 0.47	87.38 ± 0.48
U	**Baseline**	39.58 ± 0.12	27.27 ± 0.13	14.98 ± 0.08	21.52 ± 0.09	28.03 ± 0.13	17.45 ± 0.09
	Change Detection	125.10 ± 0.27	97.70 ± 0.27	145.37 ± 0.23	111.37 ± 0.22	83.40 ± 0.22	132.14 ± 0.20
	Forgetting	178.59 ± 0.33	149.76 ± 0.36	259.34 ± 0.27	176.49 ± 0.32	147.17 ± 0.35	260.00 ± 0.27
m_j	**Baseline**	0.66	0.62	0.68	0.68	0.63	0.68
	Change Detection	0.67	0.67	0.67	0.67	0.67	0.67
λ	**Forgetting**	0.97	0.97	0.95	0.97	0.97	0.95

$[0.5, 1.0)$ first in 0.05 increments, then in 0.01 increments around the ideal value). These ideal settings are also provided in Table 5.1.

From Table 5.1, we first observe that in all network and phenomena types, both of our solutions significantly outperformed the baseline approach in terms of the number of phenomena values for which the team formed correct beliefs (N_{800}). This is because, due to the Institutional Memory Problem (c.f., Section 5.3.2-5.3.3), agents using the baseline approach only quickly converged to the first value of the phenomenon ($True$), then

maintained that belief regardless of new information received. As a result, the teams using the baseline approach only formed correct beliefs about half of the phenomenon values (since half were equal to the initial value). On the other hand, both of our solutions successfully adapted their beliefs over time after the phenomenon changed values, enabling the teams to achieve many more correct collective beliefs (as evidenced by higher N_{800} values, close to the maximum = 10), as well as superior numbers of individually correct (C) and incorrect (I) agents. Therefore, both of our solutions are improvements over the previously successful LTIS approaches when considered in environments with non-stationary phenomena.

Comparing our two solutions with one another, we observe that for the periodic sequence–the one with equally lengthy amounts of time for each phenomenon value–both of our solutions were equally successful in forming correct beliefs as a team (N_{800}) for all 10 phenomenon values. However, for the random sequences that contained several phenomenon values with shorter durations, the forgetting-based solution significantly outperformed the change detection and response algorithm. We suspect this is due to the agents' ability to adapt to changes independently by time-decaying beliefs without having to wait for a neighbor to signal a change. That is, it appears that the ideal forgetting rate allowed the agents to move towards uncertainty *faster* after a phenomenon changed values, indicated by a greater average number of unconfident agents (U), thereby overcoming inertia faster. The forgetting-based solution also typically achieved a much greater number of agents with correct beliefs (C), indicating that not only did the teams using the forgetting solution hold *more correct beliefs collectively as a team* (N_{800}), but also *more individual agents were also correct*.

However, the performance of the forgetting-based solution was highly dependent on the particular λ value used. In particular, we observed a sharp decline in performance when λ was below its optimal value, quickly falling to N_{800} values near 0 (caused by almost only unconfident agents) with decreases in λ of only 0.04. Thus, although our forgetting solution outperformed our change detection and response solution, it requires more fine-tuning (both λ and the weight to place in new information m_j, which was simply set to the theoretical best 0.63 (Glinton, Scerri, & Sycara, 2010) in these experiments). Therefore, the forgetting solution would require more consideration if deployed to real-world applications, whereas the change detection and response solution requires less foresight. In the future, we intend to further investigate predictive models to determine how to automatically set λ.

Comparing across network types (RN, SWN, and SFN) in Table 5.1, we observe that the network type did not generally impact the performance of any of the approaches for either of the phenomenon types. Thus, our solutions behave equally well in a wide range of settings. Of note: the optimal time decay parameter λ for our forgetting-based solution was slightly lower for SFN, so a small additional amount of fine tuning could be necessary based on network structure.

Robustness against Faulty/Malicious Agents. Next, we compare our solutions' performance in the presence of malicious and/or faulty agents propagating incorrect information, making it more difficult for the team to converge to correct beliefs. Figures 5.4-5.5 present the number of phenomenon values to which the teams correctly converged (N_{800}) for the periodic and random phenomenon sequences, respectively.

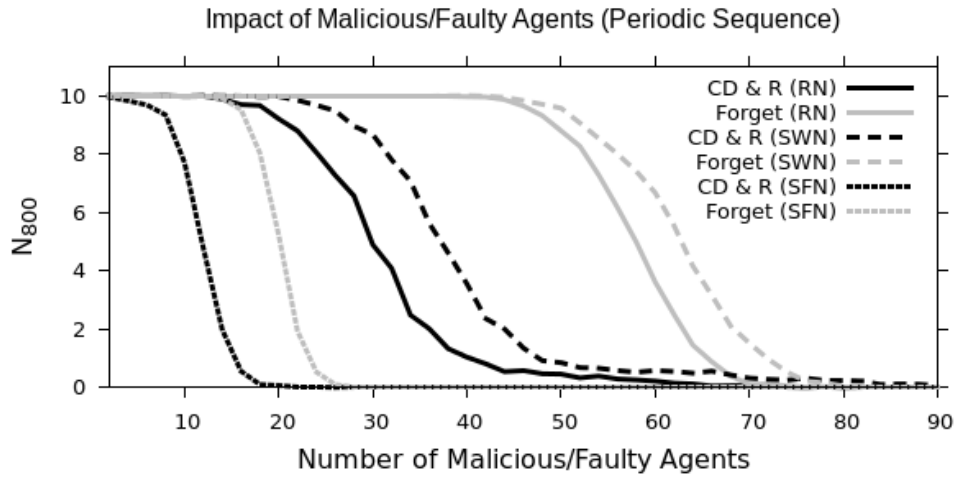

Figure 5.4: Impact of Malicious/Faulty Agents under Periodic Sequences of Phenomenon Values

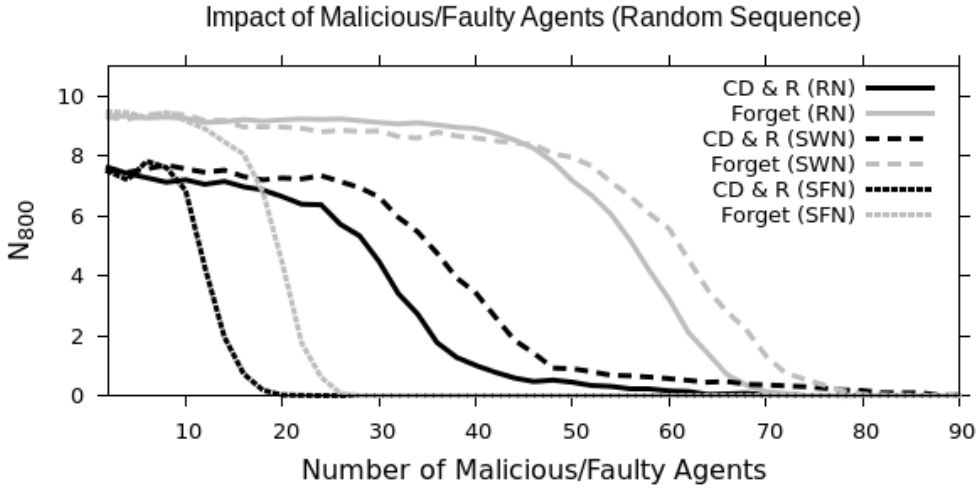

Figure 5.5: Impact of Malicious/Faulty Agents under Random Sequences of Phenomenon Values

As expected, the change detection and response algorithm is indeed more susceptible to bad information exchanged by malicious and/or faulty agents. Unexpectedly, though, the forgetting solution was actually very robust against bad agents and information. Specifically, correct convergence still occurred for many phenomena values ($N_{800} > 8$) in the RN and SWN networks as the number of bad agents approached 50. This is significant because 50 is also the number of agents with sensors inputting new information into the system. Therefore, even as the amount of bad information

approached the amount of freshly observed information, the forgetting-based solution maintained high performance. In the future, we intend to explore how robustness is related to the amount of newly sensed information input by sensors.

In contrast to our earlier results (Table 5.1) with no malicious or faulty agents, network structure *did* impact team performance once bad agents were included. In particular, in the SFN case, team performance quickly declined as the number of malicious/faulty agents increased. Recall that in our experiments, bad agents were deliberately chosen to be the most connected agents that exhibit the greatest influence on the team. In SFN, these agents have greater connectivity than in the RN and SWN, increasing the influence of such malicious/faulty agents and thus degrading team performance. In the future, we intend to study how to improve robustness in the presence of such super-connected agents.

Also unexpectedly, agent performance was *not* monotonically decreasing as the number of faulty and/or malicious agents increased, especially for random phenomena. Instead, it appears that *small numbers of agents sharing incorrect information are actually beneficial to overcoming inertia* in the Institutional Memory Problem. That is, occasionally receiving incorrect information seems to cause agents to fail to reach *overly* confident opinions, yielding less confident beliefs and thus less inertia for forming new beliefs after a phenomenon changes. This lower inertia caused by a few bad agents enabled the team to converge to team-wide correct beliefs more often for the shorter duration phenomenon values in the random sequences, especially with the change detection solution that suffered more than forgetting.

5.8. Conclusions

In conclusion, we addressed information sharing in multiagent systems observing localized, non-stationary phenomena common to many real-world applications of multiagent systems and emerging computational systems where complex environments are increasingly under observation. We first analytically predicted the impact of adding non-stationarity to an existing model for information sharing of localized phenomena called LTIS. We discovered the Institutional Memory Problem caused by inertia in the agents' beliefs, then developed two novel distributed solutions: (1) a change detection and response algorithm for improving information sharing in local neighborhoods, and (2) a forgetting-based solution for independent adaptation by individual agents. Using an empirical study considering different types of phenomena value sequences and network structures, as well as varying numbers of malicious and/or faulty agents, we evaluated the advantages and disadvantages of both types of solutions. We discovered that our change detection and response algorithm yielded improved off-the-shelf performance over prior algorithms for stationary phenomena, whereas our forgetting-based solution achieved even greater performance and robustness to bad information accidentally or intentionally injected into the system by bad agents. However, our forgetting-based solution requires additional parameter tuning (in the λ belief decay rate) to the specific application.

In the future, we intend to advance our research by (1) developing analytical models describing agent beliefs under non-stationarity and localization, extending the prior models of Glinton, Scerri, & Sycara (2010), (2) using these models to develop an approach to automatically tune the λ parameter for our forgetting- based solution, and (3) evaluate our approach in real-world deployments of multiagent information sharing.

CHAPTER 6 AD HOC INFORMATION GATHERING

In this chapter, we present additional research on the Information Sharing Problem, this time focusing on developing a solution for enabling agents to adapt their usage of different sources of information in an important subproblem: ad hoc information gathering. Namely, agents operating in complex (e.g., dynamic, uncertain, partially observable) environments must gather information from various sources to inform their incomplete knowledge. Two popular types of sources include: (1) directly sensing the environment using the agent's sensors, and (2) sharing information between networked agents occupying the same environment. We address agent reasoning for appropriately selecting between such types of sources to update agent knowledge over time. In particular, we consider ad hoc environments where agents cannot collaborate in advance to predetermine joint solutions for when to share vs. when to sense. Instead, we propose a solution where agents individually learn the benefits of relying on each type of source to maximize knowledge improvement. We empirically evaluate our learning-based solution in different environment configurations to demonstrate its advantages over other strategies. This chapter was accepted for publication as a full paper for the AAMAS 2015 conference (Eck & Soh, 2015) and will be presented in May 2015.

6.1. Introduction

One of the most fundamental responsibilities of intelligent agents is *understanding their complex (e.g., dynamic, uncertain, partially observable) environments*, which guides agent reasoning, actuation, and goal accomplishment. Often, agents lack complete knowledge of their environment *a priori* and must update their

understanding over time. These updates are informed by incorporating information gathered whilst operating in the environment. Two popular types of sources of information include (1) an agent independently *sensing* its environment, gathering direct observations as a result of the agent's actions and sensors, and (2) receiving *shared* information from other agents operating in the same environment (either cooperatively for the sake of the system or for individual profit by self-interested agents).

Depending on the application, these two types of sources might have different benefits (e.g., types of information provided, information quantity and quality) and costs (e.g., resource and time expenses). Sensing can be performed on demand, gathering information as soon as the agent needs, and the agent can do so in a timely fashion without taking away from other agents' activities. Information sharing, on the other hand, can propagate information through the entire system potentially faster and with less cost (not waiting for each agent to individually sense the same information). However, relying on sharing also means waiting for another agent to possess the desired information, and sharing takes time and resources away from other agent activities that could instead further the sharing agent's individual goals.

Because of these differences, agents in applications where both sources coexist face an interesting question: *when should I use sensing to update my understanding vs. when should I request information from other agents and rely on shared information?* Answering this question leads to a challenging tradeoff between using the two types of information sources that when properly balanced could lead to improved agent behavior and goal accomplishment (e.g., through lower cumulative cost and higher quality knowledge).

Traditionally, agents in a shared environment would pre-coordinate when they might be willing and able to share information so that each agent could plan appropriately to know when to sense vs. when to rely on shared information. However, in many applications, this pre-coordination might not be possible. Specifically, in **ad hoc environments** where pre-coordination is impossible and agents might not know the behaviors or capabilities of their peers in advance (Stone et al., 2010), agents cannot determine *a priori* the value of relying on shared information against the value of sensing alone. This is especially true in many types of ad hoc environments that are also *open environments*, where agents can join and leave the environment over time. Agent openness is especially problematic to information sharing because the availability of shared information changes over time and knowledge about the environment disappears with departing agents (who knew more than newly joining agents). Thus, determining when to sense vs. when to rely on shared information is especially difficult in ad hoc environments. In this chapter, we study how agents should balance the sensing vs. sharing tradeoff in ad hoc environments, henceforth referred to as the ad hoc information gathering (AHIG) problem.

In order to solve the AHIG, we propose a learning-based solution where agents individually learn over time how different types of information gathering actions (independently sensing vs. requesting shared information) improve their knowledge about the environment. Through learning, agents can find good information gathering strategies without relying on pre-coordination in ad hoc environments, instead treating other agents as part of the environment affecting the quality of their information gathering. Moreover, learning enables each agent to adapt its behavior as it interacts with

different agents, which is valuable in open environments where agents join and leave over time. Thus, through learning, agents can individually adapt their behavior to maximize their own knowledge improvement by learning the benefits of using different types of information sources without requiring coordination between agents.

However, because agents are operating in complex environments with incomplete information, learning is generally a computationally complex problem: learning in partially observable environments is much harder than learning in fully observable environments. To simplify the agents' learning process, we show how the agents' general problem of *understanding the current state of the complex environment* can be transformed to a simpler problem of *improving agent knowledge over time*, in a transformation we term the **Knowledge State MDP** exploiting full observability of current measures of agent knowledge as intermediate states for guiding agent decision making. As a result, an agent can learn faster how to gather information in the environment to best refine knowledge. Moreover, this transformation is potentially useful in more general information gathering problems (beyond the AHIG).

To demonstrate the effectiveness of our transformation and learning-based solution, we empirically evaluate using different experimental environment configurations how well agents learn to select between different information sources over time to improve their knowledge. We discover that our solution outperforms baseline approaches maximizing either sensed or shared information, and does so by appropriately selecting between different information sources at different times to best refine agent knowledge. Furthermore, our results indicate that learning about how to gather

information is most beneficial when information is most scarce (and careful information gathering is most necessary).

6.2. Problem

The AHIG problem occurs whenever a set of agents observe the same environment and can share information but cannot coordinate in advance to determine when agents might share or what quality of information they might provide. This includes real world examples such as (1) intelligent ad hoc sensor networks, where agents are deployed on wireless sensors that are randomly dropped to monitor an open space, (2) robotic search and rescue operations, where different organizations might bring their own robots to explore the same disaster area, and (3) ad hoc traffic information networks, where intelligent agents on cars communicate with a road infrastructure system as they navigate through town to report and understand traffic conditions.

6.2.1. AHIG Formulation

We formalize the AHIG problem as follows. A set of agents $Ag = \{i\}$ exist in a shared environment and are connected by a bidirectional communication network. Because communication costs grow as the network becomes larger, each agent's local neighborhood $N(i)$ is relatively small compared to the size of the entire network. Occasionally, due to openness, some agents will leave the network and others will join. Thus, we represent the current set of agents at time t with Ag_t, and likewise for an agent i's neighborhood $N_t(i)$.

Also in the shared environment are a finite set of phenomena $P = \{j\}$ that represent objects, entities, or properties of the environment that the agents need to understand. Each phenomenon j can take states from a finite set $PS_j = \{ps\}$, and the current state of each phenomenon in the dynamic environment changes with probability cp each time step. In AHIG, the agents are tasked with always understanding the current state of each phenomenon, which requires forming correct knowledge about each phenomenon over time that is refined through gathering information.

To gather information about a particular phenomenon, agents can perform different actions that use different types of sources for information. In particular, each agent can (1) *sense* each phenomenon directly using its sensors, or the agent can (2) *request* that its neighbors $N_t(i)$ share their beliefs about a phenomenon. We assume that the agent's sensors are noisy and imperfect, returning correct observations about the sensed phenomenon's current state with accuracy acc (and an incorrect observation with probability $1 - acc$). Agents can also perform a third type of action: (3) agents can respond to requests from neighbors with a *share* action communicating the agent's uncertain current knowledge about the state of the phenomenon in question.

The goal of each agent is to form accurate knowledge about each phenomenon, representing good knowledge about the current state of the environment, while minimizing costs incurred in sensing. Agents are awarded a reward for each time point during which they have relatively certain knowledge about a phenomenon, whereas sensing actions and requests for information incur costs to the agent. To encourage self-interested agents to collaborate, the agents are also awarded a small reward for sharing information with their neighbors, but only when requested (to avoid unnecessarily

consuming the communication resources) and when they are confident about the current state of the requested phenomenon (to avoid sharing unfruitful information). Otherwise, agents receive a penalty for sharing information.

To illustrate, consider a search and rescue (S & R) robotics example, where robot agents Ag_t explore a damaged building after a natural disaster. Here, the phenomena P represent different locations where victims might be trapped, and the phenomenon states PS_j indicate whether victims exist at location j. A robot i can either directly observe the environment with a noisy camera sensor (that consumes limited energy), or the agent can communicate with nearby robots $N_t(i)$ using line-of-sight communications. The goal of each robot is to determine with certainty whether victims exist in each location so that they can be rescued by human first responders, all-the-while minimizing energy and time costs.

Of final note: how agents represent their knowledge about the phenomena in the environment, as well as how they choose actions to refine their knowledge are not specified in the general AHIG formalization. Different domains, applications, and solutions might require different approaches to these features (knowledge and decision making) that are internal to the agent. Indeed, in real-world ad hoc environments, different agents produced by different developers might even use different approaches to these features in the same environment. However, agents must have some shared language that is consistent between agents for communicating shared information. In this chapter, we choose the knowledge representation and decision making process as part of our solution, described in Section 6.4.

6.2.2. Related Work

The AHIG problem is closely related to several other problems in the multiagent systems literature. First, the Large Team Information Sharing (LTIS) problem (e.g., Glinton et al., 2010; 2011; Pryymak et al., 2012, c.f. Section 5.2) also considers a team of agents working together to observe at least one phenomenon in the environment, where agents both sense the current state of the phenomenon individually, as well as share information through the team's network. Prior research on LTIS has focused primarily on producing analytic models for the flow of information through the team of agents (Glinton et al., 2010; 2011), as well as developing distributed solutions for adapting information flow to achieve accurate, consistent, shared beliefs (Glinton et al., 2010; Pryymak et al., 2012). However, LTIS differs from the AHIG in several key ways. First, in LTIS, the team of agents is constant over time (i.e., there is no agent openness), and agents follow a pre-coordinated strategy of when to share information. Second and most importantly, in LTIS agents do not choose between sensing, requesting, or sharing information. Instead, agents with sensors (which might not be all agents in the team) always receive observations from their sensors at every time point. Additionally, agents never request information; instead, they automatically share information with their neighbors whenever (and only when) they reach new highly certain knowledge about a phenomenon. Thus, LTIS does not consider the tradeoff between relying on different types of information as in the AHIG.

Another closely related problem studied in the multiagent systems literature is trust and reputation systems (e.g., Sabater & Sierra, 2002; Sensoy et al., 2013; Teacy et al., 2006). In such systems, agents can also request and share information with one

another to provide additional information to refine agent knowledge over time. The primary focus in trust and reputation systems is to determine how to incorporate such shared information: should the sharing agent be highly trusted and should their information heavily influence the receiving agent's knowledge, or should an agent be cautious when receiving information from another agent with which it has limited experience interacting? Like LTIS, this research does not focus on balancing information from other agents with the agent's own sensing, and thus does not solve the AHIG problem, but it is complementary in that reasoning about the trustworthiness and reputation of neighboring agents as information sources could be used to improve an AHIG agent's decision making process (which we intend to pursue as future work).

Finally, previous research in ad hoc environments has focused on problems such as how to lead teams of agents without communication (Agmon et al., 2014; Genter et. al, 2013), as well as how to learn to interact with a single Markovian agent (Chakraborty & Stone, 2013). Since information sharing inherently requires communication, our research differs from the former (although in our work, agents still cannot pre-coordinate how they will interact, under the broad definition of ad hoc environments (Stone et. al, 2010)). Similar to the latter, we also use reinforcement learning to determine how to interact with other agents, although our approach considers an agent working with multiple other agents in the environment.

6.3. POMDP Formulation

In order to solve the AHIG and gather the necessary information to understand the environment, each agent faces a sequential decision making problem of planning a sequence of actions to perform that refine its incomplete knowledge while minimizing

costs incurred for gathering such information. In most partially observable environments (which includes AHIG since sensing phenomena returns noisy, imperfect observations), sequential decision making problems are generally solved by some variant of partially observable Markov decision processes (POMDPs) (Kaelbling et al., 1998). This is especially true of applications of single agent control of environment monitoring (e.g., (Araya-Lopez et al., 2010; Boutilier, 2002; Doshi and Roy, 2008; Eck & Soh, 2013c; Spaan et al., 2010), similar to our S & R robot example), which we extend in this chapter to multiagent information gathering in ad hoc environments.

To setup our solution, in the following we next provide a description of both how the AHIG problem could be cast as a POMDP and the problems with this formulation. Then, in Section 6.4, we will introduce our Knowledge State MDP transformation of the POMDP for sequential decision making for information gathering problems.

6.3.1. AHIG as a POMDP

Since the AHIG is a sequential decision making problem in a partially observable environment (i.e., phenomenon states are partially observable), casting the AHIG as a POMDP is a natural starting point for a potential solution. In particular, we consider the POMDP formulation for the AHIG $\langle S, A, T, Z, O, R, \gamma, b_0 \rangle$ summarized in Table 6.1.

In this POMDP, the state space S contains variables representing different information about situations faced by the agent: partially observable PS_j represent the different states each phenomenon can take (e.g., the presence of victims in different locations in our S&R example), and fully observable S_{Req} and S_{Rec} represent counts per phenomenon of how long it has been since the agent last requested that its neighbors share information or received a neighbor's request, respectively. These count variables

Table 6.1: POMDP Formulation of AHIG Problem

POMDP Variable	Values	AHIG Description
State Variables $\boldsymbol{S}$	$S_{Req} \times S_{Rec} \mathrm{X}_{j \in P} PS_j$ $S_{Req} = \{0, \ldots, k\}^{\lvert P \rvert}$ $S_{Rec} = \{0, \ldots, k\}^{\lvert P \rvert}$	Counts of the number of time steps since the agent last requested (S_{Req}) or received a request for information (S_{Rec}), up to a maximum count k, and the partially observable phenomenon states (PS_j)
Actions $\boldsymbol{A}$	$\bigcup_{j \in P} \{Sense_j, Request_j, Share_j\}$	Actions (1) sensing a particular phenomenon j, (2) requesting information from neighbors about phenomenon j, and (3) sharing information to neighbors about phenomenon j (for each $j \in P$)
Observations $\boldsymbol{Z}$	$\{null\} \cup PS_j$	Observations about (1) the phenomenon state of a particular phenomenon, or (2) receiving no observation at all.
Transition Function $\boldsymbol{T}$	$[0, 1]$	Likelihood of (1) the request counts changing (deterministically) and (2) the phenomenon states changing (stochastically) after each action
Observation Function $\boldsymbol{O}$	$[0, 1]$	Likelihood of the agent receiving observations about partially observable phenomenon states from its actions
Reward Function $\boldsymbol{R}$	$\mathbb{R}$	The rewards received for taking different actions based on the current state of the environment and the agent's knowledge.
Discount Factor $\boldsymbol{\gamma}$	$(0, 1)$	A discount factor to use for weighting future, uncertain rewards
Initial Belief State $\boldsymbol{b_0}$	$U(0, 1)$	The probability the agent ascribes to each phenomenon state being the correct initial state of each phenomenon (a uniform distribution).

are useful for tracking (1) whether the agent recently requested information, so that it doesn't request too frequently and disrupt other agents, and (2) whether a neighbor requested information so that the agent knows if it is appropriate to share its own knowledge. Given this S, the belief state b represents the agent's uncertain knowledge about each phenomenon's hidden state. This knowledge is refined using information Z collected from actions A using Eq. 2.4. Beliefs start with pure uncertainty (a uniform distribution over phenomenon states, e.g., a location is equally likely to contain a victim or not).

Since the environment is dynamic, the transition function T encodes the probability that phenomena change states at each time point (to a new state with probability cp, else phenomenon states stay the same with probability $1 - cp$, c.f., Section 6.2.1) (e.g., whether a previously safe location collapses and traps new victims, or trapped victims are rescued). The fully observable states transition deterministically

each time step: the count for each phenomenon j in S_{Req} is incremented by one (up to k) unless the agent requests new information about j, and the count for each phenomenon j in S_{Rec} is incremented by one (up to k) unless the agent shares information (in which case it reverts to k to indicate no request from a neighbor is pending).

The observation function O, on the other hand, encodes the probability that the agent receives information about a particular phenomenon depending on the action taken. For $Sense_j$ actions, O encodes that the agent observes the correct state with probability acc (the agents' sensor's accuracy, c.f., Section 6.2.1) and a wrong state with probability $1 - acc$ (e.g., whether or not the robot's camera correctly identifies a victim in a room). Other actions return a null observation since they do not directly gather information about the state of any phenomenon in the environment.

The reward function R encodes (1) the rewards for having high certainty beliefs or sharing information when requested, and (2) the costs for information gathering actions or penalties for sharing unrequested or uncertain information as described in Section 6.2.1. Maximizing cumulative rewards leads the agent to highly certain knowledge (for which it receives a reward) while minimizing costs used to refine its knowledge.

6.3.2. Problems with POMDP Formulation

However, a few problems exist in this solution formulation. First, the observation set Z only considers observations from the $Sense_j$ actions and does not handle shared information from neighbors, which would occur some delayed amount of time after a $Request_j$ action. Although Z could be modified to include additional variables for received information, this limits the types of shared information neighbors can provide to

discrete quantifications of the neighbor's beliefs (e.g., the locations most likely to contain victims), which loses information about the neighbor's uncertainty (e.g., the probabilities of victims in each location). Otherwise, the observation space would be continuous (and thus very difficult to work with) if neighbors shared their full belief states.

Second, even if Z were extended to include shared information, there is no way for the observation function in a single agent POMDP to encode a probability that a neighbor shares a phenomenon state from its own beliefs (in response to a request) without some pre-coordination and agreement between agents. That is, agents must understand the likelihoods that a neighbor both (1) shares a particular piece of information (dependent on the neighbor's beliefs that change over time) and (2) any information at all (e.g., a robot might be busy and unwilling to share information at the current time). Without this information, an agent cannot calculate the overall probability that it receives any particular information from a neighbor at any point in time, necessary for updating its beliefs with Eq. 2.4 with shared information, nor plan what information it might receive over time. Therefore, a single agent POMDP formulation of the AHIG will not directly work in ad hoc environments.

Of note, traditional multiagent variants of POMDPs (e.g., DEC-POMDPs, Distributed POMDPs, and I-POMDPs (Bernstein et al., 2002; Gmytrasiewicz & Doshi, 2006; Nair et al., 2005)) provide some methods for handling both of the aforementioned problems; however, these types of POMDPs require pre-coordination so are inappropriate for ad hoc environments and do not scale well with the number of agents.

To resolve these problems inherent in a POMDP-based AHIG model, we need to add some method to incorporate shared information (which is inherently multiagent in

nature) outside of the (single agent) POMDP framework's belief updates. Then, the agent should still make decisions based on its current knowledge, but it also needs a way to plan how its beliefs will change to form an action policy.

6.4. Knowledge State MDP

In this section, we first describe how we propose to incorporate shared information from other agents, building on the aforementioned POMDP formulation. Then, we describe a *transformation* of the POMDP into a MDP that looks at solving the AHIG from a metareasoning perspective, decoupled from how the agent refines its knowledge when it receives new information. Finally, we introduce a *learning process* for the MDP that enables an agent to learn how to choose actions to take to refine its knowledge in ad hoc environments without requiring pre-coordination about how and when other agents will share information.

6.4.1. Incorporating Shared Information

For agent knowledge about phenomenon states, we consider probability distributions over all possible phenomenon states very similar to belief states described in Section 6.3.1. We reuse notation with $b_t(j, ps)$ the probability that the agent believes phenomenon $j \in P$ is currently $ps \in PS_j$. For $Sense_j$ actions, beliefs update from b to b' after receiving observation z about phenomenon j using Bayes' rule:

$$b'(j, ps) = {}^{prob}/_{\eta}\left[(1 - cp)b(j, ps) + \sum_{\substack{ps' \in PS_j \\ ps' \neq ps}} \left(\frac{cp}{|PS_j| - 1}\right) b(j, ps')\right] \qquad (6.1)$$

where $prob = acc$ when $z = ps$, else $p = 1 - acc$. This is equivalent to the belief updates performed with Eq. 2.4 using the POMDP formulation described in Section 2.2.1.

With respect to shared information, we assume[23] that agents share the full information about their beliefs: the probabilities ascribed to each phenomenon state for the particular phenomenon for which a neighbor sent a request. Then, the corresponding belief update for shared information b_{Sh} is:

$$b'(j,ps) = \frac{b(j,ps)\cdot[w\cdot b_{Shared}(j,ps)+(1-w)\cdot(1-b_{Shared}(j,ps))]}{\sum_{ps'\in PS_j} b(j,ps')\cdot[w\cdot b_{Shared}(j,ps')+(1-w)\cdot(1-b_{Shared}(j,ps'))]} \qquad (6.2)$$

where constant weight[24] w dampens shared information so that uncertain shared beliefs do not cause agents to become certain too quickly from little gathered information.

Using these two rules, agents can incorporate information from both from (1) directly observing a phenomenon with its sensors, and (2) its neighbors sharing their knowledge.

6.4.2. Knowledge State MDP Transformation

At the core of AHIG, the agent's behavior does not necessarily depend on which *particular* phenomenon state is currently correct for each phenomenon, but instead the problem is really about how the agent should choose actions to improve its *knowledge* (noting that actions to improve knowledge could be equivalent for each actual phenomenon state). After all, the agents' goal is to form highly certain knowledge about each phenomenon using the information available in the environment. For instance, in our S&R example, a robot will base its information gathering on *how certain its knowledge is* about a location (looking to resolve its uncertainty so that it knows where all victims are as quickly as possible), which is *internal* to the agent and independent of

[23] Other types of information might instead be shared, based on the domain, which we leave to consider as future work.

[24] Such weights are common in the information sharing literature (e.g., Glinton et al., 2010; Pryymak et al., 2012) and could be learned as in trust and reputation systems to further refine our solution, which we intend to explore in the future. Please see (Glinton et. al, 2010) for a more elaborate discussion of the impact of weight w.

whether or not an *external* unknown location actually contains victims. The robot isn't necessarily responsible for using the refined knowledge for a separate task (that is done by human first responders), but the goal of the agent in the AHIG is to develop high quality knowledge that could subsequently be used for other purposes, depending on the application.

Given this insight, we transform the above POMDP into what we call the **Knowledge State MDP**—an alternative formulation of the problem directly enabling an agent to make decisions of how to gather information based on considering the current state of its *knowledge*, as opposed to the state of the *environment* (including the states of phenomena under observation). This provides a ***metareasoning solution*** enabling the agent to choose how to gather information based on reflecting about the quality of its knowledge without worrying about the domain-specific contents of that knowledge. As a result, the agent's decision making (at a metareasoning level) is decoupled from its knowledge refinement (at a standard reasoning level), as desired.

The Knowledge State MDP can be mathematically described as a MDP $\langle S_{Req} \times S_{Rec} \times K, A, T, R \rangle$, summarized in Table 6.2. Here, the partially observable part of the state space is replaced with the different knowledge states K of the agent's knowledge (which are fully observable when reflecting on the agent's knowledge) as it gathers information to understand its environment. K is combined with the S_{Req} and S_{Rec} state variables representing counts of time since requests were sent or received, described in Section 6.3.1.

Recall that in the AHIG, the primary concern of the agent is to form highly certain beliefs, so the state of agent knowledge should reflect how much certainty exists in the

Table 6.2: Knowledge State MDP Formulation

MDP Variable	Values	AHIG Description
State Variables S	$S_{Req} \times S_{Rec} \times \mathrm{K}$ $S_{Req} = \{0, \dots, k\}^{\lvert P \rvert}$ $S_{Rec} = \{0, \dots, k\}^{\lvert P \rvert}$ K: $H(b, j)$ discretized into $\lvert K \rvert$ bins	Counts of the number of time steps since the agent last requested information (S_{Req}) or received a request for information (S_{Rec}), up to a maximum count k, and the agents current certainty (K) in the current state of each phenomenon j
Actions A	$\bigcup_{j \in P} \{Sense_j, Request_j, Share_j\}$	Actions (1) sensing a particular phenomenon j, (2) requesting information from neighbors about phenomenon j, and (3) sharing information to neighbors about phenomenon j (for each $j \in P$)
Transition Function T	$T_{S_{Rec}, S_{Req}} \cdot T_K \in [0, 1]$	Likelihood of state changes, as the product of the likelihood of request state variable transitions and knowledge state transitions T_K.
Knowledge State Transition Function T_K	[0, 1]	Likelihood of knowledge state changes (i.e., changes in certainty) after taking each action
Reward Function R	$\mathbb{R}$	Rewards received for taking different actions based on the agent's knowledge.
Discount Factor γ	$(0, 1)$	A discount factor to use for weighting future, uncertain rewards

agent's knowledge. Then, the agent can take actions that improve its certainty and result in better knowledge states (closer to full certainty). Given that the knowledge representation b described in Section 6.4.1 is a probability distribution over possible phenomenon states for each phenomenon, an appropriate measure of certainty in each phenomenon j's state (independent of application) is the entropy $H(b, j) \in [0,1]$ of the probability distribution representing its knowledge (Araya-Lopez et al., 2010):

$$H(b, j) = 1 + \frac{1}{\log |PS_j|} \sum_{ps \in PS_j} b(j, ps) \log b(j, ps) \tag{6.3}$$

To create a set of finite knowledge states K using $H(b, j)$ so that the MDP is a discrete state MDP, and thus is much more tractable, we suggest discretizing the certainty values into equal sized bins so that there exist a desired number of states $|K|$. Note that a larger $|K|$ creates a finer grained separation between different knowledge states, potentially enabling better planning, whereas a smaller $|K|$ make the MDP faster to solve (and has implications on the learning process described in Section 6.4.3).

Given the rewards in the AHIG described in Section 6.2.1, it is important to note that the same reward encoding works for the Knowledge State MDP as well: knowledge states identifying high certainty earn a reward, and action-based costs, rewards, and penalties stay the same.

6.4.3. Learning Knowledge State Dynamics

Now, within the Knowledge State MDP, the key to guiding appropriate action selection is the dynamics of how knowledge states change based on each action $a \in A$. That is, how actions lead the agent to improve its certainty over time. This information is encoded in the knowledge state transition function T_K. Unfortunately, due to a lack of pre-coordination to determine how and when agents will share information, this function is undetermined initially. However, whereas this was a problem in our suggested POMDP-based solution in Section 6.3.1, the transformation into an MDP makes it feasible to perform model-based reinforcement learning[25] (MB-RL) (Kaelbling, Littman, & Moore, 1996) to learn this transition function through interactions with the environment and other agents (and adjust it over time as agent openness causes the environment to change), instead of having to rely on pre-coordination.

In general, any MB-RL algorithm should be sufficient to learn the knowledge state transition function T_K. For our experimental setup in this chapter, we use a learning approach for the transition function similar to recent variants (Hernandez et al., 2014; Szita & Szepesvari, 2010) of one of the most popular MB-RL algorithms: R-max (Brafman & Tennenholtz, 2002). In particular, this algorithm uses frequentist counting

[25]Although MB-RL algorithms also exist for POMDPs (e.g., Ross et al., 2007), such algorithms have high complexity and are not generally applicable in practice for POMDPs of moderate to large state spaces (which grows quickly with phenomena P and their states PS_j for Section 6.3.1's POMDP).

by maintaining a table counting the number of times $n(s_t, a, s_{t+1})$ that the agent observes a transition from state s_t to s_{t+1} after taking action a, then the algorithm updates the transition table to:

$$T(s_t, a, s_{t+1}) = \frac{n(s_t, a, s_{t+1})}{n(s_t, a)} \quad (6.4)$$

whenever the total count of observed transitions for a state-action pair $n(s_t, a) = \sum_{s_{t+1} \in S} n(s_t, a, s_{t+1})$ equals a parameter m, after which the learning counts for the state-action pair are reset to 0. A smaller m enables *faster updates* to the transition function, whereas a larger m ensures *more precise updates* (by relying on more observed transitions before updating). Of note, smaller $|K|$ are also beneficial here, causing the same knowledge state to be encountered more frequently, and thus more frequent learning updates.

Considering the Knowledge State MDP, learning T_K amounts to learning exactly how the certainty in the agent's knowledge changes based on (1) each information gathering action, and (2) how long it has been since the agent requested information (since this alerts the agent both how timely neighbors respond, as well as whether or not they respond at all). Understanding such changes to agent knowledge is exactly the information the agent needs to determine which information gathering actions to perform in order to reach highly certain knowledge and achieve its primary goal--actions that are more likely to lead to high certainty knowledge states from the current knowledge state are actions that most improve the agent's knowledge, as desired.

This learning process only requires feedback from the agent's knowledge updates (using sensed or shared information) to observe exactly which knowledge state (i.e., certainty) transitions occur after taking each action. Thus, the agent can learn over time

how its knowledge changes when it senses, as well as when it requests shared information (including how long such information takes to arrive), without having to know in advance when or how other agents will choose to share information. Therefore, this learning process bypasses the problems of other solutions in ad hoc environments without requiring pre-coordination to understand the behaviors of neighboring agents and their impact on knowledge refinement. Moreover, the agent also adapts its understanding of knowledge state transition changes over time, which is important for open environments where information sharing can become more or less prevalent over time, in which case a smaller m might be useful for more frequent learning and faster adaptation to the changing environment.

By planning with the reward function R, the agent plans to reach certainty as fast as possible (by maximizing rewards for certain knowledge) while also minimizing costs required for gathering information, making the agent both effective and efficient at its task. Thus, our Knowledge State MDP coupled with MB-RL is an appropriate solution for the AHIG.

It is important to note that this Knowledge State MDP transformation is closely related to a similar metareasoning framework in the literature: the Observer Effect POMDP (Eck & Soh, 2013c), which combines fully observable knowledge states with partially observable environment states to guide agents to perform actions that refine knowledge over time. Our solution here differs in that (1) it learns the transitions in knowledge over time, as opposed to the domain-specific value of information, and (2) extends this type of approach to a multiagent setting where learning enables the agent to reason about the affects of other agents on its own knowledge.

6.5. Experimental Setup

To better understand our approach and investigate its performance in different AHIG settings, we conducted experiments empirically evaluating how well our Knowledge State MDP and MB-RL process guide agent information gathering using different information sources, including information sharing, without requiring pre-coordination. In particular, we considered a range of network configurations that might reflect different types of environments and applications.

That is, we varied the average neighborhood size $N_t(i)$, where larger neighborhoods made shared information more prevalent, whereas smaller neighborhoods represent more communication-constrained environments (e.g., our S&R robot example where only a few robots might be within line-of-sight of one another). The networks were randomly generated using an Erdos-Renyi model (Erdos and Renyi, 1960). Since the environment was ad hoc, agents knew nothing about their neighbors in advance. Moreover, we made the environment open, where a predetermined percentage (10%) of the agents left periodically (every 100 time steps) and new agents joined. This agent openness also reduced the availability of information over time, making information sharing more or less valuable at different points in time. Within a neighborhood (and throughout the set of agents), agents differed in their capabilities: different agents had different sensing accuracies, making them better or worse at quickly gathering good information from the environment to share with their neighbors upon request. This follows in the tradition of other ad hoc environments (e.g., Chakraborty & Stone, 2013; Stone et al., 2010), where agents must work with agents with different capabilities than themselves.

The different opponents in our experiments included: (1) **KSMDP+MB-RL**: our Knowledge State MDP solution with MB-RL, using the UCT algorithm (Kocsis & Szepesvari, 2006) to plan each time step using the learned MDP, (2) **KSMDP**: our Knowledge State MDP solution without MB-RL (also using UCT for planning, but only using the initial, uninformed T_K function where knowledge states only transition to the closest states), and two baselines: (3) **AlwaysSense**: where agents *maximized sensing* for information gathering and did not plan for information sharing since pre-coordination was not possible (which serves as a lower bound on acceptable agent performance), and (4) **RequestThenSense**: where agents requested information about each phenomenon every k steps to *maximize information sharing*, then either sensing the rest of the time to further inform agent knowledge or sharing if the agent had certain knowledge to help its neighbors.

We evaluated agent performance using three measures averaged per time step: (1) average belief certainty across all agents, (2) average proportion of agents with correct, highly certain knowledge, and (3) average total rewards earned by all agents. Each agent earned rewards: +10 whenever its b was sufficiently certain (i.e., $H(b) \geq 0.8$), -1 for every $Sense_j$ action, -1 for each $Request_j$ action (or -5 if $S^j_{Req} < k$), and +1 for each $Share_j$ action (whenever $S^j_{Rec} < k$, else -5). The other parameters were set: $|Ag| = 100$ (which is too large for multiagent POMDP solutions as a baseline), average $N_t(i) \in \{2,4,6,8,10\}$, $|P| = 1$, $|PS_j| = 3$, $cp = 1\%$, $acc \sim (0.5,0.8)$, $\gamma = 0.99$, $k = 6$, $|K| = 100$. Each configuration repeated 50 times for 1000 time steps.

6.6. Results

We begin our results analysis by considering the agent's average belief certainty, presented in Figure 6.1. From these results, we first observe that our Knowledge State MDP solution with and without MB-RL (respectively KSMDP+MB-RL, KSMDP) achieved *higher amounts of belief certainty* than either of the baselines. This implies that, instead of trying to maximize either type of information gathering, our KSMDP formulation enabled agents to *appropriately select between information gathering actions using different sources to best refine their knowledge*, as opposed to either (1) requesting shared information as often as possible (RequestThenSense), or (2) independently relying only on sensed information (AlwaysSense).

Comparing across average neighborhood sizes, we observe that as neighborhood size increased and information became more available through sharing (due to each agent being connected to more potential information sources), the average certainty of the agents increased. Most notably, *certainty increased fastest for our KSMDP solutions*, implying that they became better at controlling information gathering as information became more readily available (although they also achieved the best performances when the neighborhoods were smallest and information was most limited).

Further comparing between the two variants of our solution, we note that although adding MB-RL did not improve belief certainty very much, it did so at a 0.05 statistically significant level for the smaller average neighborhood sizes (2-6). This was when information was least available (due to fewer neighbors as sources) and thus more care was necessary during information gathering. Therefore, *adding MB-RL to our Knowledge State MDP was most beneficial when information gathering was most challenging*.

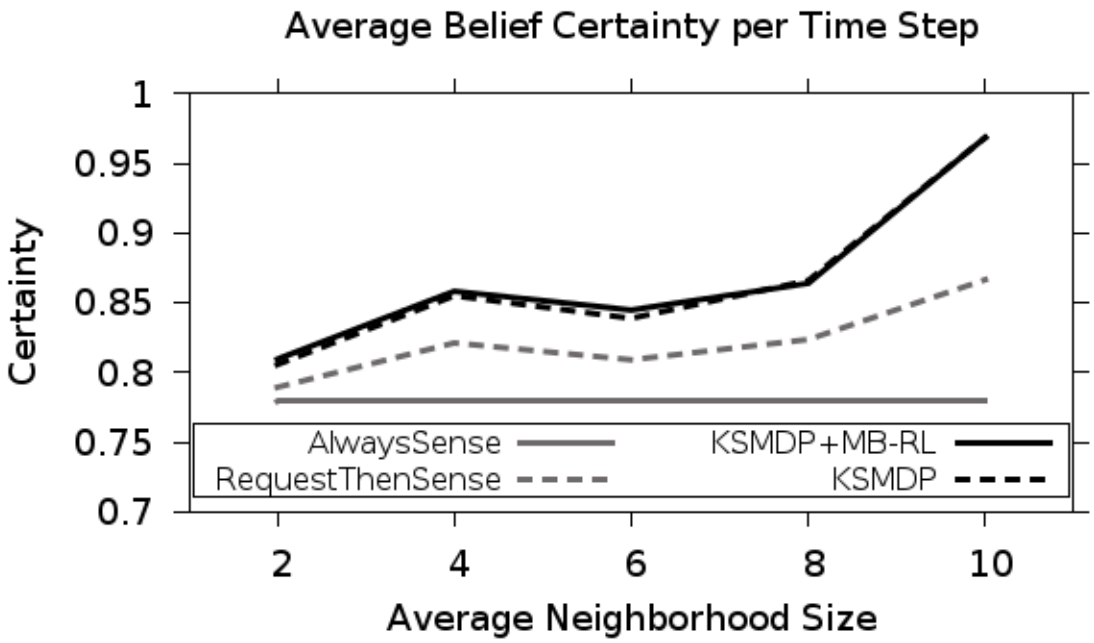

Figure 6.1: Average Belief Certainty
Note: in all figures, 95% CIs are too small to display

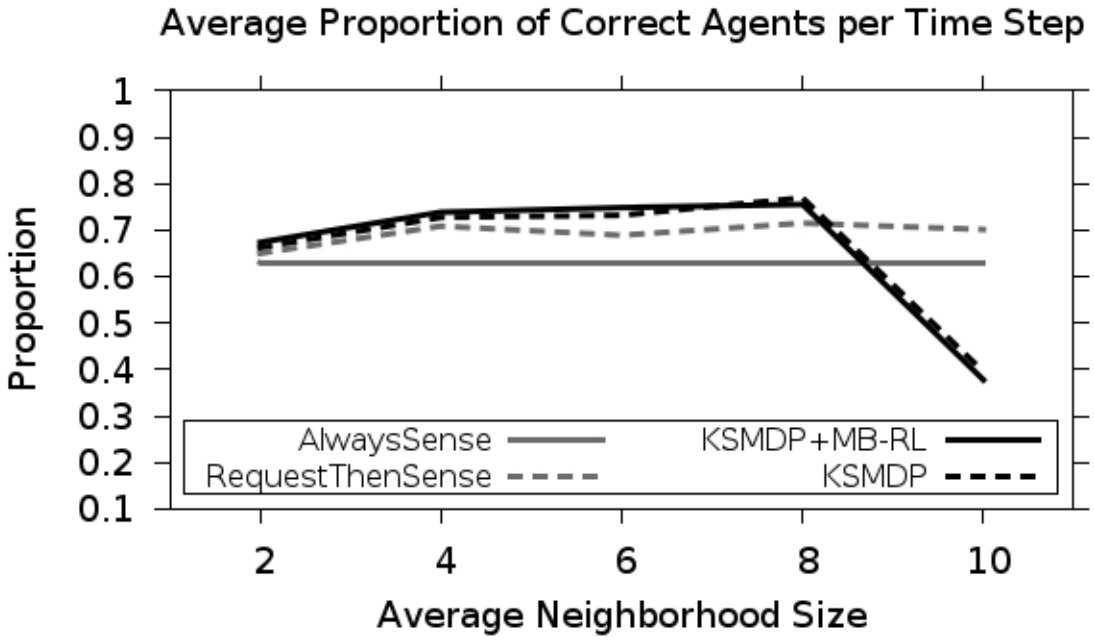

Figure 6.2: Average Proportion of Correct Agents

Next, we consider the average proportion of agents holding correct and highly certain beliefs, presented in Figure 6.2. Maximizing this performance measure was the desired emergent behavior of solving the AHIG. From these results, we additionally observe that not only did our Knowledge State MDP-based solutions (KSMDP+MB-RL and KSMDP) lead to more certainty in the agents' beliefs, but those beliefs were also correct. Thus, agents were *gathering the right information to understand their environments* over time. Additionally, we again find evidence of the benefits of using MB-RL to learn how agent knowledge changes based on different information gathering actions using different sources: the improvement over KSMDP (without MB-RL) for KSMDP+MB-RL *was more pronounced when information was most constrained* (i.e., at lowest neighborhood sizes).

Interestingly, we also observe that for the largest neighborhood size (10) considered in our experiments, our KSMDP solutions actually achieved very few correct agents compared to the baselines, which is in sharp contrast to the other neighborhood sizes. Upon further inspection, what happened is the agents fell victim to **institutional memory**: they converged to highly certain beliefs (as indicated in Figure 6.1) because of the prevalence of shared information (favoring requesting information over continually sensing the environment). This caused the agents to become stuck with outdated beliefs that didn't adapt as the phenomenon changed over time since very few agents continued sensing the phenomenon directly. In the future, we intend to explore how we can adapt our solution to learn to avoid this problem.

Finally, we consider the average total rewards earned by all agents per time step, presented in Figure 6.3. We observe that for all but the lowest neighborhood sizes, our

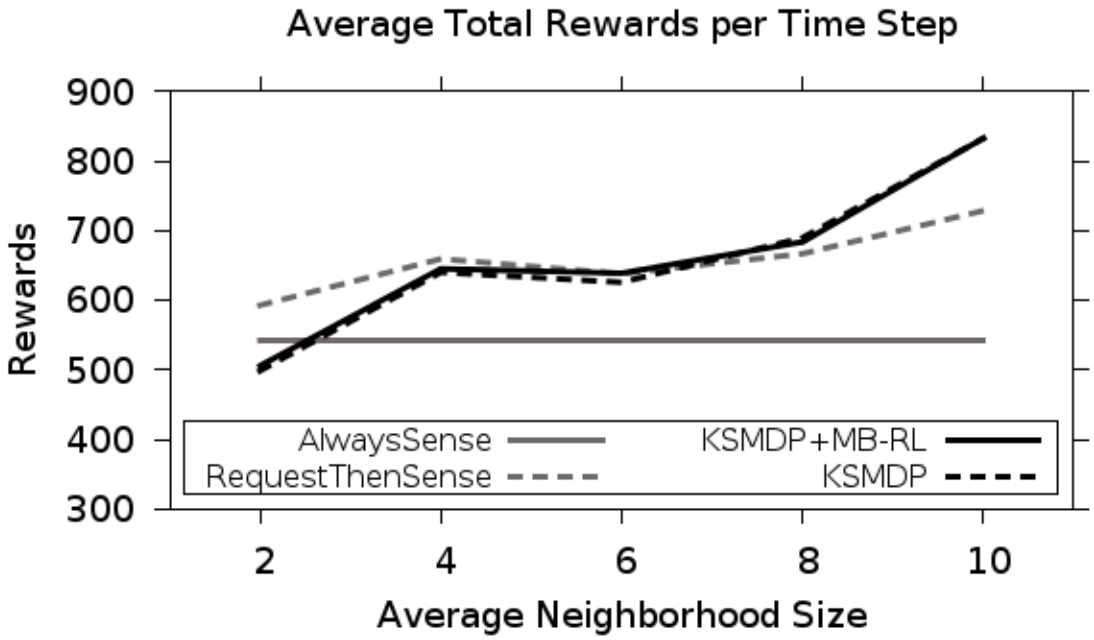

Figure 6.3: Average Total Reward

KSMDP approaches--that directly maximized rewards to plan information gathering actions—earned the highest cumulative rewards due to achieving high certainty while trying to minimize costs. Of note, for the lowest neighborhood sizes (2-4) when information was most scarce, the KSMDP approaches were *willing to accept more information gathering cost in order to achieve higher certainty and correctness*, as displayed in Figures 6.1-6.2, ultimately attaining the agents' primary goal.

6.7. Conclusions

In summary, we introduced the ad hoc information gathering (AHIG) problem occurring when agents must balance relying on different types of information sources (knowing when to sense vs. when to rely on shared information from other agents) in order to understand their complex environment without pre-coordinating with one another. From the tradition of using POMDPs to guide agent decision making, we proposed a transformation called the Knowledge State MDP that enables agents to

control information gathering by reflecting on (fully observable) changes to their knowledge. To address the inability of agents to pre-coordinate in ad hoc environments, we added a MB-RL process to the Knowledge State MDP that enables agents to learn how their knowledge changes when relying on different information sources. This includes learning how and when neighbors might be willing to share information to supplement an agent's own sensing of the environment. Using an experimental study, we investigated the performance of our Knowledge State MDP (with and without MB-RL) in a range of environment configurations (with varying number of information sources), and discovered: (1) our solution gathered better information and earned greater rewards than baseline strategies of trying to maximize the usefulness of either type of information source (sensing vs. shared information), and (2) adding MB-RL enabled agents to best guided their behavior when information availability was most limited (and high quality information gathering was most necessary).

In the future, we intend to: (1) combine our solution with trust and reputation systems to further learn not only *when to rely on different information sources*, but *how much weight to place in received information*, which could help overcome the institutional memory problem (where weight w could be adapted to avoid agents rapidly converging to certain beliefs when shared information is prevalent), and (2) study how to use the Knowledge State MDP to balance information gathering about different phenomena in the environment to avoid *imbalanced knowledge* potentially caused by favoring sources for one phenomenon over the others.

CHAPTER 7 CONCLUSIONS AND FUTURE WORK

In this chapter, we conclude by summarizing the research presented in this dissertation, as well as describing the future research we intend to pursue in continuation of our overall research vision for reflective, deliberative information gathering. We summarize our research again in the context of the two problems addressed under this dissertation in Figure 7.1.

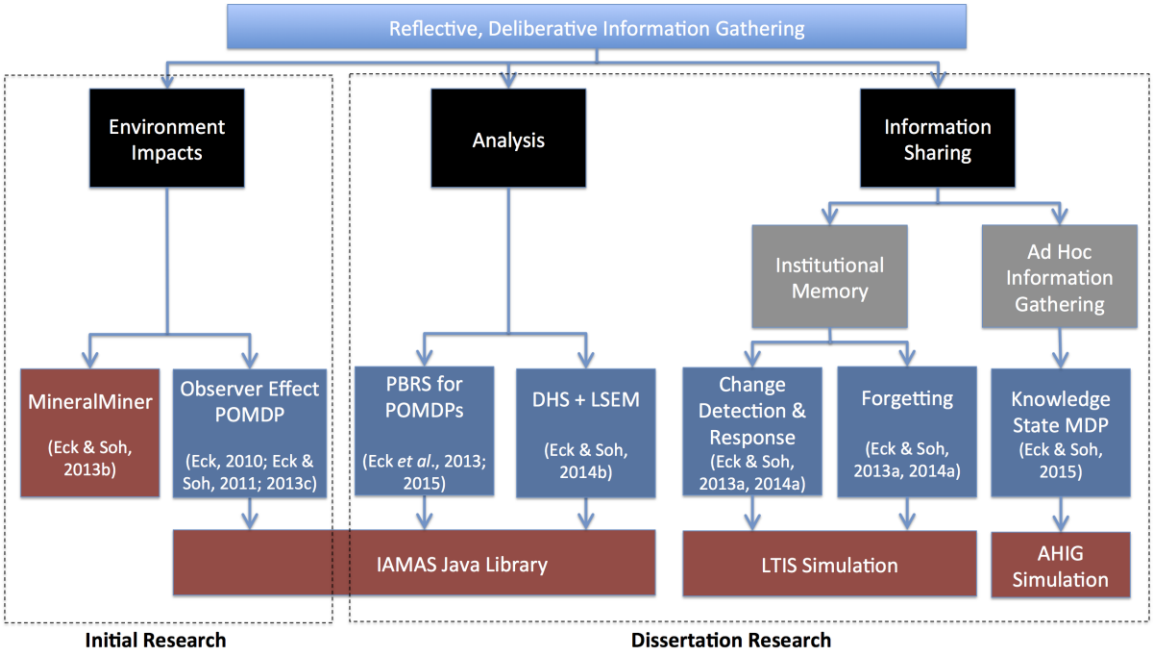

Figure 7.1: Summary of Research

7.1. Summary

In Chapter 1, we introduced our greater research vision of reflective, deliberative information gathering as a means for improving an agent's understanding of its environment in order to improve both the agent's decision making and subsequent task and goal accomplishment. We defined two core problems addressed in this dissertation: the Analysis Problem and the Information Sharing Problem, then outlined our five

solutions. Finally, we summarized the key contributions this dissertation (summarized again in Section 7.3 later in this chapter).

In Chapter 2, we provided a high level overview of prior research from the literature related to our umbrella concept of reflective, deliberative information gathering for intelligent agents and multiagent systems. In particular, we summarized past research introducing the notion of deliberative information gathering, where agents make intentional decisions to control their sensing to refine their knowledge. We especially focused on the use of the active sensing POMDP, related to our solutions in Chapters 3, 4, and 6, for deliberative information gathering. Then, we described prior research on metareasoning and reflection to improve information gathering. Next, we summarized related research on information sharing in complex environments, especially those with resource constraints affecting the ability of agents to share information. Finally, we described how the research presented in this dissertation (and our prior research on reflective, deliberative information gathering) both fit within and extend the state-of-the-art in agent-based information gathering.

In Chapter 3, we presented our first solution to the Analysis problem: potential-based reward shaping for POMDPs. This approach has three key benefits. First, PBRS for POMDPs embeds additional measures reflecting action benefits and costs (including with respect to sensing) in reward optimization by agents to produce agent behavior that best addresses the tradeoff between benefits and costs to improve overall agent behavior. Second, the approach also generalizes to a solution for improving agent planning in devices with constrained computational resources (e.g., wireless sensors, robots) by guiding the agent towards large rewards beyond the myopic planning (i.e., limited

number of planning steps) caused by a lack of computational power. Finally, our solution also represents a novel technique for adding metareasoning to agent reasoning with POMDPs without increasing the size of the agent's state space (and thus does not increase the computational complexity of the reasoning process). Our experimental results demonstrated that PBRS best improves agent planning in large, complex environments, whereas state-of-the-art heuristic and Monte Carlo search approaches performed similarly (or slightly better) in smaller and/or less complex environments.

In Chapter 4, we presented our second solution to the Analysis problem: situationally-aware online POMDP planning using Difference-based Heuristic Selection (DHS) and the Long Sequence Entropy Minimization (LSEM) heuristic. This solution improves information gathering in highly uncertain environments to promote more efficient and effective planning with limited time constraints. In this solution, the LSEM heuristic reflects on the expected certainty in agent knowledge in order to guide agent's planning so that the agent quickly gathers the necessary information to operate in highly uncertain environments. DHS, on the other hand, enables the agent to select between different heuristics measuring different types of information to decide how to plan based on its most pressing need: reducing knowledge uncertainty vs. maximizing rewards. Our results demonstrated that DHS with LSEM can find successful policies in highly uncertain environments two orders of magnitude faster than the best previously reported heuristic search online POMDP planning algorithms, whereas existing state-of-the-art heuristic and Monte Carlo search approaches performed similarly well (or slightly better) in environments with less uncertainty.

In Chapter 5, we moved to the Information Sharing Problem and considered information sharing about non-stationary environment phenomena between large teams of cooperative agents where only a few agents can directly observe the phenomena of interest. This limitation on sensing results in a challenging problem caused by environment non-stationarity: the institutional memory problem where large portions of the team of agents become stuck with outdated beliefs as the environment changes, no matter how much additional information enters the team through additional sensing. We presented two solutions for mitigating this problem: (1) a change detection and response algorithm where agents work together within local sub-teams to quickly detect changes to the observed phenomenon, and (2) a forgetting-based algorithm, where agents independently use belief decay to maintain up-to-date beliefs to avoid problems caused by faulty agents or malicious information. Our experimental results demonstrated that both solutions successfully avoid the institutional memory problem and lead to consistent, accurate beliefs through the team as the environment changes, extending past solutions (that work well in stationary environments) to guide information sharing in non-stationary environments.

Finally, in Chapter 6, we studied another subproblem of the Information Sharing Problem: ad hoc information gathering where agents can share information with peers to augment their information gathering (in addition to sensing the environment directly), but agents have no advance knowledge of their peers' capabilities or willingness to cooperate. As a result of this lack of *a priori* knowledge about peers, agents cannot pre-coordinate their sharing behavior (as we assume for the solutions presented in Chapter 5), but instead agents must learn to work together over time. We presented a solution called

the Knowledge State MDP where agents individually learn the benefits of relying on each type of source to maximize knowledge improvement. Our experimental results demonstrated that our approach results in higher belief certainty and more accurate beliefs than baseline strategies.

7.2. Future Work

In the future, we plan to continue our research on reflective, deliberative information gathering in several ways. At the end of each of our solution chapters (Chapters 3-6), we outlined specific ways we intend to advance our research presented in each chapter. Here, we consider broader opportunities and challenges we intend to address.

Specifically, we envision two primary avenues for future research: (1) applying reflective, deliberative information gathering to real-world applications of intelligent agents and multiagent systems, and (2) extending reflective, deliberative information gathering as a methodology for developing methods for autonomous data analytics in "big data" and "data science" solutions.

First, throughout our research on reflective, deliberative information gathering, we have studied information gathering from a *fundamental perspective* using theoretical analyses and empirical studies using popular benchmarks and simulations. We now want to move towards studying reflective, deliberative information gathering in real-world applications of intelligent agents and multiagent systems. For example, we are currently working on developing intelligent agents capable of interacting with human users to gather information about their preferences, opinions, and knowledge through intelligently adapting self-administered surveys or computer-assisted interviews (Al Baghal *et al*.,

2013; Ruther *et al*., 2013; Atkin *et al*., 2014; Eck *et al*., 2014; Arunachalam *et al*., 2015; Atkin *et al*., 2015; Eck, Soh, & McCutcheon, 2015; Wettlaufer *et al*., 2015). Because respondents might become bored or frustrated with such surveys or interviews, one of the agent's tasks is to manage the progress of the survey or interview to predict potential problems with data collection (e.g., respondents skipping questions, providing false information to quickly finish the survey or interview, or quitting data collection altogether), then adapt the questions being asked of the respondent in order to avoid such problems from occurring or mitigating their impacts on data collection. This applied research is part of on ongoing grant from the NSF (SES-1228937) in partnership with the U.S. Census Bureau and Gallup and will result in better information gathering tools for working with human respondents. We are also interested in applying reflective, deliberative information gathering to other real-world domains, such as search and rescue robotics (as used as a motivating example throughout this dissertation), social network analysis, game playing (e.g., Eck & Soh, 2012a) and computer-supported, collaborative learning systems (e.g., Khandaker *et al*., 2011; Eck, Soh, & Brassil, 2013).

Second, reflective, deliberative information gathering is also closely related to designing autonomous agents capable of performing automated, intelligent "big data" analytics—enabling reasoning about combining the right data from the right sources at the right time to enable agents (and humans working with such agents) to make the right decisions to solve problems in real-time. We intend to further extend our research to develop agents capable of (1) assisting domain experts in their data analyses, (2) performing autonomous analyses (both individually and in agent teams) to discover interesting, novel patterns from data for use by human data consumers, and (3) train

novices how to perform data analytics using a wide array of computational methods. This work also extends our prior design of adaptive knowledge assistants (Eck & Soh, 2012b).

7.3. Contributions

We conclude this dissertation by re-emphasizing its key contributions. Specifically, we have provided:

1. A better fundamental understanding of agent-based sensing in complex environments, valuable for a wide range of intelligent agents and multiagent systems domains. This knowledge can be applied to improve agent reasoning and actuation in different applications, as well as improves our overall understanding of general artificial intelligence.
2. A set of solutions to provide reflective, deliberative information gathering to improve agent-based sensing, including single-agent POMDP solutions and cooperative agent team-based solutions.
3. New techniques for metareasoning by intelligent agents with broader impacts beyond sensing control.
4. Implemented simulation environments mimicking real-world scenarios and applications for studying agent-based sensing.
5. The addition of implementations of many of our solutions to a Java library for artificial intelligence that can be reused for other AI and agent-based projects.

REFERENCES

Adamczyk, P.D. & Bailey, B.P. 2004. If not now, when? The effects of interruption at different moments within task execution. In *Proceedings of the 2004 ACM SIGCHI 2004 Conference on Human Factors in Computing Systems (CHI'04)*. Vienna, Austria. April 24-29. 271-278.

Agmon, N., Barrett, S., & Stone, P. 2014. Modeling uncertainty in leading ad hoc teams. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'14)*, Paris, France, May 5-9, 2014, 397-404.

Al Baghal, T., Phillips, A. L., Ruther, N., Belli, R. F., Eck, A., Stuart, L.C., and Soh, L.-K. 2013. What are you doing now? Audit trails, activity level responses and error in the American Time Use Survey. In *Proceedings of the 68th Annual Conference of the American Association for Public Opinion Research*, Boston, Massachusetts, May 16-19, 2013.

An, B., et. al. 2011. Agent-mediated multi-step optimization for resource allocation in distributed sensor networks. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, Turner, Yolum, Sonenberg and Stone (eds.), Taipei, Taiwan, May 2-6, 2011, 609-616.

Araya-Lopez, M., Buffet, O., Thomas, V., & Charpillet, F. 2010. A POMDP extension with belief-dependent rewards. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS'10)*, Vancouver, B.C., Canada. Dec. 6-9, 2010, 64-72.

Arunachalam, H. Atkin, G., Wettlaufer, D., Eck, A., Soh, L.-K., & Belli, R.F. 2015. I know what you did next: Predicting respondent's next activity using machine learning. In *Proceedings of the 70th Annual Conference of the American Association for Public Opinion Research*, Hollywood, Florida, May 14-17, 2015.

Asmuth, J., Littman, M.L., & Zinkov, R. 2008. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, Chicago, IL, July 13-17, 2008, 604-609.

Atkin, G., Arunachalam, H., Eck, A., Soh, L.-K., & Belli, R. 2014. Designing an intelligent time diary instrument: visualization, dynamic feedback, and error prevention and mitigation. In *Proceedings of the 69th Annual Conference of the American Association for Public Opinion Research*, Anaheim, California, May 15-18, 2014.

Atkin, G., Arunachalam, H., Eck, A., Wettlaufer, D., Soh, L.-K., & Belli, R.F. 2015. Using machine learning techniques to predict respondent type from a priori demographic information. In *Proceedings of the 70th Annual Conference of the American Association for Public Opinion Research*, Hollywood, Florida, May 14-17, 2015.

Bajcsy, R. 1988. Active perception. *Proceedings of the IEEE*, *76*(8), 996-1005.

Barabasi, A.-L. & Albert, R. 1999. Emergence of scaling in random networks, *Science*, Oct. 15, 1999. *286*:509-512.

Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*. *27*(4). 819-840.

Bertsekas, D.P. & Castanon, D.A. 1999. Rollout algorithms for stochastic scheduling problems, *Journal of Heuristics*, *5*:89-108.

Boutilier, C. 2002. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'02)*, Edmonton, Alberta, Canada, July 28 – Aug. 1, 2002, 239-246.

Boyd, S.P. & Vandenberghe, L. 2004. *Convex optimization*. Cambridge University Press: Cambridge, U.K.

Brafman, R. I. & Tennenholtz. M. 2002. R-max – A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research. 3*:213-231,

Casper, J. & Murphy, R.R. 2003. Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on SMC – Part B: Cybernetics. 33*(3):367-385.

Calisi, D., Farinelli, A., Iocchi, L., & Nardi, D. 2007. Multi-objective exploration and search for autonomous rescue robots, *Journal of Field Robotics*, *24*(8/9):763-777.

Chakraborty, D. & Stone, P. 2013 Cooperating with a Markovian ad hoc teammate. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, Saint Paul, MN, May 6-10, 2013, 1085-1092.

Chalupsky, H. *et al*. 2001. Electric Elves: Applying agent technology to support human organizations. In *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence (IAAI '01),* Seattle, WA. Aug. 7-9, 2001, 51-58.

Cohn, R. Maxim, M., Durfee, E., & Singh, S. 2010. Selecting operator queries using expected myopic gain. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'10)*, Toronto, Canada, Aug. 31-Sept. 3, 2010.

Cohn, R., Durfee, E., & Singh, S. 2011. Comparing action-query strategies in semi-autonomous agents. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*, San Francisco, CA, Aug. 7-11, 2011.

Conley, K. and Carpenter, J. 2007. Towel: towards an intelligent to-do list. In *Proceedings of the AAAI'07 Spring Symposium on Interaction Challenges for Intelligent Assistants*.

Cox, M.T. & Raja, A. 2011. Metareasoning: an introduction. In M. Cox & A. Raja (Eds.), *Metareasoning: Thinking about Thinking*, MIT Press, 3-14.

Devlin, S. & Kudenko, D. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Ssytems (AAMAS'11)*, Turner, Yolum, Sonenberg and Stone (eds.), Taipei, Taiwan, May 2-6, 2011, 225-232.

Devlin, S. & Kudenko, D. 2012. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), Valencia, Spain, June 6-8, 2012.

D'Mello, S. & Graesser, A. 2012. AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Transactions on Interactive Intelligent Systems 2*(4), Article 23 (December 2012), 39 pages.

Doshi, F. and Roy, N. 2008. The permutable POMDP: fast solutions to POMDPs for preference elicitation. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, Estoril, Portugal, May 12-16, 2008, 493-500.

Eck, A. 2010. Agent sensing with stateful resources. Masters Thesis. University of Nebraska-Lincoln.

Eck, A. & Soh, L.-K. 2011. Agent sensing with stateful resources (Extended Abstract). In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, Tumer, Yolum, Sonenberg and Stone (eds.), Taipei, Taiwan, May, 2-6, 2011, 1283-1284.

Eck, A. & Soh, L.-K. 2012a. Active sensing for opponent modeling in poker. In *Proceedings of the 2012 Computer Poker Symposium*, Toronto, Ontario, Canada, July 23, 2012.

Eck, A. & Soh, L.-K. 2012b. Adaptive knowledge assistants. In *Proceedings of the First International Workshop on Human-Agent Interaction Design and Models (HAIDM'12)*, Valencia, Spain, June 4, 2012. (co-located with AAMAS'12)

Eck, A. & Soh, L.-K. 2012c. Evaluating POMDP rewards for active perception (Extended Abstract). In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), Valencia, Spain, June 6-8, 2012.

Eck, A. & Soh, L.-K. 2012d. Investigating reward functions for active sensing POMDPs. Unpublished.

Eck, A. & Soh, L.-K. 2013a. Dynamic facts within large team information sharing (Extended Abstract). In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, Ito, Jonker, Gini, and Shehory (eds.), Saint Paul, Minnesota, May 8-10, 2013.

Eck, A. & Soh, L.-K. 2013b. MineralMiner: An active sensing simulation environment. *Multiagent and Grid Systems*, *9*:197-226.

Eck, A. & Soh, L.-K. 2013c. Observer effect from stateful resources in agent sensing, *Autonomous Agents and Multiagent Systems*, *26*(2):202-244.

Eck, A. & Soh, L.-K. 2014a. Intelligent information sharing for localized, non-stationary phenomena. In *Proceedings of the 6th International Workshop on Emergent*

Intelligence in Networked Agents (WEIN'14), Paris, France, May 5, 2014. (co-located with AAMAS'14)

Eck, A. & Soh, L.-K. 2014b. Online heuristic planning for highly uncertain domains. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'14)*, Lomuscio, Scerri, Bazzan, and Huhns (eds.), Paris, France, May 5-9, 2014, 741-748.

Eck, A. & Soh, L.-K. 2015. To ask, sense, or share: Ad hoc information gathering. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*, Bordini, Elkind, Weiss, Yolum (eds.), Istanbul, Turkey, May 4-8, 2015.

Eck, A., Soh, L.-K., & Brassil, C. 2013. Supporting active wiki-based collaboration. In *Proceedings of the 10th International Conference on Computer Supported Collaborative Learning (CSCL'13)*, Madison, Wisconsin, June 15-19, 2013.

Eck, A., Soh, L.-K., Devlin, S., & Kudenko, D. 2013. Potential-based reward shaping for POMDPs (Extended Abstract). In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, Ito, Jonker, Gini, and Shehory (eds.), Saint Paul, Minnesota, May 8-10, 2013.

Eck, A., Soh, L.-K., Devlin, S., & Kudenko, D. 2015. Potential-based reward shaping for finite horizon online POMDP planning. *Autonomous Agents and Multiagent Systems*. Available online first at http://link.springer.com/article/10.1007/s10458-015-9292-6

Eck, A., Soh, L.-K., & McCutcheon, A.L. 2015. Predicting breakoff using sequential machine learning methods. In *Proceedings of the 70th Annual Conference of the American Association for Public Opinion Research*, Hollywood, Florida, May 14-17, 2015.

Eck, A., Stuart, L., Atkin, G., Soh, L.-K., McCutcheon, A., & Belli, R. 2014. Making sense of paradata: Challenges faced and lessons learned. In *Proceedings of the 69th Annual Conference of the American Association for Public Opinion Research*, Anaheim, California, May 15-18, 2014.

Erdos, P. & Renyi, A. 1960. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 17-61.

Floreano, D. & Mondada, F. 1994. Active perception, navigation, homing, and grasping: An autonomous perspective. In *Proceedings of the Perception to Action Conference*, 122-133.

Genter, K., Agmon, N., & Stone. P. 2013. Ad hoc teamwork for leading a flock. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, Saint Paul, MN, May 6-10, 2013, 531-538.

Glinton, R.T., Scerri, P., & Sycara, K. 2009. Towards the understanding of information dynamics in large scale networked systems. In *Proceedings of the 12th International Conference on Information Fusion (FUSION'09)*, Seattle, WA, July 6-9, 2009.

Glinton, R., Scerri, P., & Sycara, K. 2010. Exploiting scale invariant dynamics for efficient information propagation in large teams. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, van der Hoek, Kaminka, Lesperance, Luck, and Sen (eds.), Toronto, Canada, May, 10-14, 2010, 21-28.

Glinton, R., Scerri, P., & Sycara, K. 2011. An investigation of the vulnerabilities of scale invariant dynamics in large teams. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, Turner, Yolum, Sonenberg and Stone (eds.), Taipei, Taiwan, May 2-6, 2011, 677-684.

Gmytrasiewicz, P.J. & Doshi, P. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research, 24*:49-79.

Grass, J. & Zilberstein, S. 1997. Value-driven information gathering. In *Proceedings of the AAAI'97 Workshop on Building Resource-Bounded Reasoning Systems*.

Grass, J. & Zilberstein, S. 2000. A value-driven system for autonomous information gathering. *Journal of Intelligent Information Systems*, *14*:5-27.

Guo, A. 2003. Decision-theoretic active sensing for autonomous agents. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, Melbourne, Australia, July 14-18, 2003, 1002-1003.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision proceses, *Journal of Artificial Intelligence Research*, *13*:33-94.

Hernandez-Leal, P., de Cote, E.M., & Sucar, L.E. 2014. Exploration strategies to detect strategy switches. In *Proceedings of the 14th International Workshop on Adaptive and Learning Agents (ALA'2014)*, Paris, France, May 5-6, 2014.

Hoey, J., et. al. 2007. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *Proceedings of the 5th International Conference on Computer Vision Systems (ICVS'07),* Bielefeld, Germany, March 21-24, 2007.

Hoogendoorn, M., van Lambalgen, R. M., & Treur, J. 2011. Modeling situational awareness in human-like agents using mental models. In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI'11)*, Barcelona, Spain, July 16-22, 2011, 1697-1704.

Kaelbling, L.P., Littman, M.L., & Cassandra, A.R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*. *101*:99-134.

Kaelbling, L.P., Littman, M.L., & Moore, W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*:237-285.

Khandaker, N., Soh, L.-K., Miller, L. D., Eck, A., & Jiang, H. 2011. Lessons learned from comprehensive deployments of multiagent CSCL applications I-MINDS and ClassroomWiki. *IEEE Transactions on Learning Technologies*, *4*(1):47-58.

Klein, J., Moon, Y., & Picard, R.W. 2002. This computer responds to user frustration: theory, design, and results. *Interacting with Computers*. *14*:119-140.

Kocsis, L. & Szepesvari, C. 2006. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML'06)*, Berlin, Germany, Sept. 18-22, 2006, 282-293.

Krause, A., et. al. 2008. Robust submodular observation selection. *Journal of Machine Learning Research*, *9*:2761-2801.

Krause, A. & Guestrin, C. 2005. Optimal nonmyopic value of information in graphical models – efficient algorithms and theoretical limits. In *Proceedings of tbe Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, Edinburgh, Scotland, July 31-Aug. 5, 2005, 1339-1345.

Krause, A. & Guestrin, C. 2007. Near-optimal observation selection using submodular functions. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI'07)*, Vancouver, Canada, July 22-26, 2007.

Krause, A. & Guestrin, C. 2009. Optimizing sensing: From water to the web. *IEEE Computer, 42*(8):38-45.

Kurniawati, H., Hsu, D., & Lee, W.S. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of the 2008 Robotics: Science and Systems Conference (RSS '08)*.

Landeldt, B., Sookavantana, P., & Seneviratne, A. 2000. The case for a hybrid passive/active network monitoring scheme in the wireless internet. In *Proceedings of the 8th IEEE Conference on Networking (ICON'00)*, Singapore, Sept. 5-8, 2000, 139-143.

Lesser, V., *et al*. 2000. BIG: An agent for resource-bounded information gathering and decision making. *Artificial Intelligence*, *118*:197-244.

Mark, G., Gudith, D., & Klocke, U. 2008. The cost of interrupted work: more speed and stress. In *Proceedings of the 2008 ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*, Florance, Italy, Apr. 5-10, 2008, 107-110.

Mihaylova, L. *et al*. 2002. Active sensing for robotics – a survey. In *Proceedings of the 5th International Conference on Numerical Methods and Applications (NM&A'02)*, Borovets, Bulgaria, Aug. 20-24, 2002.

Murdock, B.B. 1993. TODAM2: A model for the storage and retrieval of item, associative, and serial-order information. *Psychological Review*, *100*(2):183-203.

Myers, K.L. *et al*., 2007. An intelligent personal assistant for task and time management. *AI Magazine*. *28*(2):47-61.

Nair, R., Varakantham, P., Tambe, M., & Yokoo, M. 2005. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, Pittsburgh, PA, July 9-13, 2005, 133-139.

Ng, A.Y., Harada, D., & Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping, In *Proceedings of the 16th International Conference on Machine Learning (ICML'99*), Bled, Slovenia, June 27-30, 1999, 278-287.

Ong, S.C.W., Png, S.W., Hsu, D., & Lee, W.S. 2010. Planning under uncertainty for robotic tasks with mixed observability. *International Journal of Robotics Research*, *29*(8):1053-1068.

Padhy, P., Dash, R.K., Martinez, K., & Jennings, N.R. 2006. A utility-based sensing and communication model for a glacial sensor network. In *Proc AAMAS'06*. Hakodate, Japan, May 8-12, 1353-1360.

Pavon, J., *et al*. 2007. Development of intelligent multisensory surveillance systems with agents, *Robotics and Autonomous Systems*, *55*:892-903.

Pineau, J., Gordon, G., & Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, Acapulco, Mexico, Aug. 9-15, 2003, 1025-1032.

Pryymak, O., Rogers, A., & Jennings, N.R. 2012. Efficient opinion sharing in large decentralized teams. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), Valencia, Spain, June 6-8, 2012

Raja, A. & Lesser, V. 2007. A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multiagent Systems*, *15*:147–196.

Rao, A. and Georgeff, M.P. 1995. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, San Francisco, California, June 12-14, 1995, 312-319.

Reitter, D. & Lebiere, C. 2012. Social cognition: memory decay and adaptive information filtering for robust information maintenance. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI'12)*, Toronto, Canada, July 22-26, 2012, 242-248.

Ross, S. & Chaib-draa, B. 2007. AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, Hyderabad, India, Jan. 6-12, 2007, 2592-2598.

Ross, S., Chaib-draa, B., & Pineau, J. 2007. Bayes-adaptive POMDPs. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS'07)*, Vancouver, B.C., Canada, Dec. 3-6, 2007.

Ross, S., Pineau, J., & Chaib-draa, B. 2008. Theoretical analysis of heuristic search methods for online POMDPs. In *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS'08)*, Vancouver, B.C., Canada, Dec. 8-11, 2008, 1233-1240.

Ross, S., Pineau, J., Paquet, S., & Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*. *32:*663-704.

Russell, S. 1995. Rationality and intelligence. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal, Quebec, Canada. 950-957.

Ruther, N., Al Baghal, T., Eck, A., Stuart, L.C., Phillips, A. L., Belli, R. F., and Soh, L.-K. 2013. Examining the relationship between error and behavior in the American Time Use Survey using audit trail paradata. In *Proceedings of the 68th Annual Conference of the American Association for Public Opinion Research*, Boston, Massachusetts, May 16-19, 2013.

Sabater, J. & Sierra, C. 2002. Reputation and social network analysis in multi-agent systems. In *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, Bologna, Italy, July 15-19, 2002, 475-482.

Sabbadin, R., Lang, J., and Ravoanjanahary, N. 2007. Purely epistemic Markov decision processes. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI'07)*, Vancouver, Canada, July 22-26, 2007, 1057-1062.

Sensoy, M., Fokoue, A., Pan, J.Z., Norman, T.J., Tang, Y., Oren, N., & Sycara, K. 2013. Reasoning about uncertain information and conflict resolution through trust revision. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, Saint Paul, MN, May 6-10, 2013, 837-844.

Shani, G., Pineau, J., and Kaplow, R. 2013. A survey of point-based POMDP solvers. *Autonomous Agents and Multiagent Systems*, *27*(1):1-51.

Silver, D. & Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS'10)*, Vancouver, B.C., Canada, Dec. 6-9, 2010, 2164-2172.

Singh, A., *et al*. 2009. Efficient information sensing using multiple robots. *Journal of Artificial Intelligence Research*. *34*:707-755.

Smith, T. & Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI'04)*, Banff, Alberta, Canada, July 7-11, 2004, 520–527.

Stone, P., Kaminka, G.A., Kraus, S., & Rosenschein, J.S. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI'10)*, Atlanta, Georgia, July 11-15, 2010, 1504-1509.

So, R. & Sonenberg, L. 2009. The roles of active perception in intelligent agent systems. In *PRIMA'05, LNAI 4078*, ed. Lukose, D. and Shi, Z. Springer-Verlang: Berlin, Germany, 139-152.

Somani, A., Ye, N., Hsu, D., & Sun Lee, W. 2013. DESPOT: Online POMDP planning with regularization. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS'2013)*, Lake Tahoe, Nevada, Dec. 5-8, 2013.

Sonu, E. & Doshi, P. 2013. Bimodal switching for online POMDP planning in multiagent settings. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'2013)*, Beijing, China, Aug. 3-9, 2013, 360-366.

Sorg, J., Singh, S., & Lewis, R.L. 2011. Optimal rewards versus leaf-evaluation heuristics in planning agents. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*, San Francisco, CA, Aug. 7-11, 2011, 465-470.

Spaan, M.T.J. 2008. Cooperative active perception using POMDPs. In *Proceedings of the AAAI 2008 Workshop on Advancements in POMDP Solvers*.

Spaan, M.T.J., Veiga, T.S., & Lima, P.U. 2010. Active cooperative perception in networked robotic systems using POMDPs. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, Taipei, Taiwan, Oct. 18-22, 2010, 4800-4805.

Spaan, M.T.J. & Vlassis, N. 2005. Perseus: Randomized value iteration for POMDPs. *Journal of Artificial Intelligence Research*, *24*:195-220.

Stein, S., Williamson, S.A., & Jennings, N.R. 2012. Decentralized channel allocation and information sharing for teams of cooperative agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), Valencia, Spain, June 6-8, 2012.

Sutton, R.S. and Barto, A.G. 1998. *Reinforcement learning: an introduction*. MIT Press:Cambridge, MA.

Szita, I. & Szepesvari, C. 2010. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML'2010)*, Haifa, Israel, June 21-24, 2010, 1031–1038.

Teacy, W.T., Patel, J., Jennings, N.R., & Luck, M. 2006. TRAVOS: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multiagent Systems*, *12*(2):183-198.

Watts, D.J. & Strogatz, S.H. 1998. Collective dynamics of 'small-world'networks. *Nature*, June 4, *393*:440-442.

Wettlaufer, D., Arunachalam, H., Atkin, G., Eck, A., Soh, L.-K., & Belli, R.F. 2015. Determining potential for breakoff in time diary survey using paradata. In *Proceedings of the 70th Annual Conference of the American Association for Public Opinion Research*, Hollywood, Florida, May 14-17, 2015

Weyns, D., Helleboogh, A., and Holvoet, T. 2005. The packet-world: a test bed for investigating situated multi-agent systems. In R. Unland, M. Klusch, & M. Calisti (Eds.), *Software Agent-Based Applications, Platforms, and Development Kits*, 383-408.

Weyns, D., Steegmans, E., & Holvoet, T. 2004. Towards active perception in situated multi-agent systems. *Applied Artificial Intelligence*. *18*:867-883.

Widmer, G. & Kubat, M. 1996. Learning in the presence of concept drift and hidden context. *Machine Learning*, *23*:69-101.

Williams, J.D. & Young, S. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, *21*:393-422.

Wooldridge, M. 1999. Intelligent Agents. In: *Multiagent systems: a modern approach to distributed artificial intelligence*, The MIT Press, Cambridge, MA, 27–77.

Yin, D., Hong, L., Xue, Z., & Davison, B.D. 2011. Temporal dynamics of user interests in tagging systems, In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI'11)*, San Francisco, California, Aug. 7-11, 2011, 1279-1285.

Yorke-Smith, N., Saddati, S., Meyers, K.L., & Morley, D.N. 2009. Like an intuitive and courteous butler: a proactive personal agent for task management. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, Budapest, Hungary. May 13-15. 337-344.

Zhang, Z. & Chen, X. 2012. FHHOP: A factored heuristic online planning algorithm for POMDPs. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI'12)*, Catalina Island, USA, Aug. 15-17, 2012, 934-943.

Zilberstein, S. 1996. Resource-bounded sensing and planning in autonomous systems. *Autonomous Robots*, *3*:31-48.

Zilberstein, S. 2008. Metareasoning and bounded rationality. *Proceedings of the AAAI Workshop on Metareasoning: Thinking about Thinking*.

Zilberstein, S. & Russell, S.J. 1993. Anytime sensing, planning, and action: A practical model for robot control. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'93)*, Bajcsy (ed.) Chambery, France, Aug. 28-Sept. 3, 1993, 1402-1407.