11-2010

Agent Sensing with Stateful Resources

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

AGENT SENSING WITH STATEFUL RESOURCES

by

Adam Dewane Eck

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Leen-Kiat Soh

Lincoln, Nebraska

November, 2010

AGENT SENSING WITH STATEFUL RESOURCES

Adam Dewane Eck, M.S.

University of Nebraska, 2010

Advisor: Leen-Kiat Soh

In many real-world applications of multi-agent systems, agent reasoning suffers from bounded rationality caused by both limited resources and limited knowledge. When agent sensing also requires resource use, the agent's knowledge revision is affected due to its inability to always sense when and as accurately as needed, further leading to poor decision making. In this research, we consider what happens when sensing activities require the use of *stateful* resources, which we define as resources whose state-dependent behavior changes over time based on usage. Specifically, sensing itself can change the state of a resource, and thus its behavior, which affects both the information gathered and the resulting knowledge refinement. This produces a phenomenon where the sensing activity can and will distort its own outcome (and potentially future outcomes), termed the *Observer Effect* after the similar phenomenon in the physical sciences. Under this effect, an agent faces a strategic tradeoff between 1) satisfying the need for knowledge refinement, and 2) satisfying the need for avoiding corruption of knowledge due to distorted sensing outcomes. To address this tradeoff, we model sensing activity selection as a Markov decision process (MDP) where an agent optimizes knowledge refinement while considering the state of the resources used during sensing. In this model, the agent uses reinforcement learning to learn a controller for activity selection, as well as how to predict expected knowledge refinement based on resource usage during sensing. Our approach is unique from other bounded rationality and sensing research as we consider how to make decisions about sensing with stateful resources which produce side-effects such as the Observer

Effect, as opposed to simply using stateless resources with no such side-effect. We evaluate our approach in 1) a fully observable robotic mining simulation, as well as 2) a partially observable user preference elicitation simulation. The results demonstrate that considering the Observer Effect during sensing activity selection through our approach yields better knowledge refinement and often better task performance than not considering the effect of stateful resource usage.

ACKNOWLEDGEMENTS

I would like to thank those who have assisted with the creation of this thesis. 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 and Dr. Ashok Samal 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-Lincoln used to complete my research, including the PrairieFire supercomputer made available by the Holland Computing Center upon which the simulations in my experiments were run.

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 ACKNOWLEDGEMENTS

I would also like to acknowledge the granting agencies that have provided financial support for this research. Specifically, this thesis has been supported by a GAANN Fellowship from the United States Department of Education (grant P200A040150), a grant from the NSF (DBI0742783), and more recently, an NSF Graduate Research Fellowship.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

In this chapter, we introduce the research presented within this thesis. We begin by setting up the context of our research within resource-constrained multiagent environments. Second, we introduce the Limited Resource Sensing Problem (LRSP) due to the resource constraints in the environment. Next, we detail the focus of this thesis: addressing the Observer Effect Tradeoff Problem (OETP), a subproblem of the LRSP. Following, we give an overview of our proposed decision theoretic solution to this problem. Afterwards, we highlight the key contributions of this thesis. We conclude with a summary of the remaining chapters.

1.1. Resource Constrained Multiagent Environments

Recent trends in computer science have resulted in an increase in the ability of computational processes to improve our everyday lives. Building on yesterday's advancements in embedded systems technologies, current research in the following fields aims to leverage the capabilities of both small devices and large-scale computing networks to solve real-world problems:

1) *cyber-physical systems*, which manage physical processes in our environments (Lee, 2008), including the control of devices in healthcare (Sun *et. al*, 2007);
2) *wireless sensor networks*, which remotely monitor and measure phenomena in physical environments ranging from glaciers (Padhy *et. al*, 2006) to underwater marine ecosystems (Akyildiz *et. al*, 2005);
3) *ubiquitous and pervasive computing systems*, which allow for "all-the-time everywhere" computation (Saha and Mukherjee, 2003) to support our daily routines, such as car computing systems (Burnett and Porter, 2001) and smart homes (Kidd *et. al*, 1999); and

4) *robotic systems*, which automatically perform common tasks traditionally performed by humans, as well as those difficult for humans to execute, including industrial manufacturing (Monostori *et. al*, 2006) and search and rescue in dangerous spaces (Casper and Murphy, 2003).

Similarly, advancements in software systems have also led to solutions aiming to enhance both the user experience and the ability of users to accomplish their goals, including:

5) *intelligent software systems*, such as recommender systems (Adomavicius and Tuzhulin, 2005), collaborative groupwork applications (e.g., Bull and Greer, 2000; Khandaker *et. al*, 2010), and personal information managers (e.g., Chalupsky *et. al*, 2001; Myers *et. al*, 2007; Yorke-Smith *et. al*, 2009), which often use rich, intelligent interfaces to improve human-computer interactions and aim to a) find information to support user activities; b) scaffold user interactions to improve productivity; c) work hand-in-hand with users to accomplish tasks; d) manage information, schedules, and tasks for users; and/or e) assist user decision making; and
6) *autonomic computing systems*, which aim to reduce the "man-in-the-loop" required to control most computing systems, allowing the system to adapt, manage, configure, heal, and protect itself without the need for human intervention, improving system effectiveness and freeing users to instead focus on high, abstract-level tasks for which their skills are better suited (Kephart and Chess, 2003; Ganek and Corbi, 2003).

One common theme to these trends and applications is the use of capabilities from the field of artificial intelligence, such as planning, scheduling, computer vision, and machine learning to adapt to changes in both the physical environment and user demands/needs. Commonly, software and hardware components are represented by intelligent agents capable of 1) *sensing* to gain information about their environments, 2) *reasoning* to make decisions to guide system

behavior, and 3) *acting* in order to change the environment to meet system goals and objectives. Depending on the applications, often these agents are aware of one another and communicate while operating either cooperatively or competitively, forming a multiagent system (MAS). For example, MASs have been used to control routing in wireless network communications (Dowling *et. al*, 2005) and matchmake collaborating users (Bull and Greer, 2000; Khandaker *et. al*, 2010). Here, a MAS approach is appropriate given the distributed nature of these systems, along with challenges commonly seen in physical environments such as uncertainty (in both sensing and actuation), noise, partial observability, dynamic processes, and multiple actors capable of changing the environment.

However, another common theme to these emerging trends and applications is the lack of resources necessary to completely support the computational processes controlling these systems during all times of operation. These limited resources can be internal or external to the system component responsible for making decisions (e.g., processor power vs. network bandwidth), as well as physical or abstract (e.g., memory and sensors vs. user skills, time, and domain knowledge). Given these constraints on reasoning, agents using techniques from artificial intelligence suffer from *bounded rationality*, requiring reasoning which aims for acceptable levels of performance, often through *satisficing* rather than optimal solutions which require more resources than are available. This problem of bounded rationality was originally studied in the context of human reasoning in the fields of economics (e.g., Conlisk, 1996; Rubinstein, 1998; Simon 1955, 1956, 1997) and cognitive science (e.g., Gigerenzer and Goldstein, 1996; Gigerenzer and Todd, 1999), but has been applied to artificial intelligence for the last twenty years (e.g., Horvitz, 1987; Boddy and Dean, 1989; Zilberstein and Russell, 1993; Russell, 1995; Zilberstein 1996; 2008).

Another constraint on reasoning studied under bounded rationality is a lack of knowledge about choices during reasoning. Thus, another constraint on intelligent agent reasoning is limited information necessary for guiding reasoning. For example, an agent might have all of the CPU cycles, memory, and time (i.e., resources) necessary to promptly make decisions and complete its tasks, but if it relies on information sources which produce accurate information slowly or infrequently needed in order to refine the agent's knowledge, the agent's reasoning is constrained to suboptimality similar to an agent running on a less capable machine with a perfect and quick information source. While this knowledge constraint is studied in conjunction with bounded rationality in the economics (e.g., Conlisk, 1996; Rubinstein, 1998) and cognitive science (e.g., Gigerenzer and Goldstein, 1996; Gigerenzer and Todd, 1999) literature, the impact of incomplete information and need for refining knowledge has been studied more under the guise of perception and belief revision in the computer science domain (e.g., Josang, 2001; Weyns *et. al*, 2004).

1.2. Limited Resource Sensing Problem

However, less understood in the computer science literature is the relationship between these two types of constraints – specifically, the impact on reasoning from sensing activities which are *also* bounded by a lack of resources necessary for gathering information and refining knowledge. For example, in a recommender system, an agent must perform preference elicitation to model its user and guide its recommendations (Adomavicius and Tuzhulin, 2005). Sometimes, this elicitation requires directly interrupting the use to inquire about preferences which can cause frustration (Adamcyzk and Bailey, 2004) and reduce user goodwill and patience with the system (Klein *et. al*, 2002), especially if they are busy (Mark *et. al*, 2008). Likewise, distributed sensing and information sharing in wireless sensor networks and multiagent systems use limited communication resources to perform sensing activities (e.g., Landeldt *et. al*, 2000).

Finally, any sensing activities performed by devices with limited energy resources consume energy, reducing the lifetime of the device (Akyildiz *et. al*, 2002). Ignoring these effects can lead to poor information gathering and subsequent poor reasoning, as well as reduced resource availability for other agent activities, both damaging overall system performance. Some preliminary work has studied this problem in the context of the computational resources (i.e, CPU, memory, and time) necessary for *analyzing* and *interpreting* raw data collected during sensing (Zilberstein and Russell, 1993; Zilberstein, 1996), but to the best of our knowledge, no research has thoroughly investigated the effect on reasoning of using limited (possibly non-computational) resources during the *(physical) process of sensing raw data from the environment*.

This relationship between limited resources and sensing results in a tradeoff between the quality of information gathered during sensing and the cost of resource consumption to gather that information. Both of these factors affect an agent's reasoning as it cannot make good decisions without good information, but reasoning also requires limited resources which could be shared with sensing activities. We term the problem of achieving this balance the **Limited Resource Sensing Problem (LRSP)**. This problem is at least as challenging as the standard bounded rationality problem in artificial intelligence since an agent must perform at least some (bounded) reasoning about its sensing behavior in order to mitigate the effects of consuming resources during sensing.

Like the standard bounded reasoning problem, the LRSP is also made more challenging due to the environments of its applications. As mentioned previously, these limited resource environments common to cyber-physical systems, wireless sensor networks, intelligent software systems, etc. are often characterized by dynamic processes, uncertainty, noise, and multiple actors (i.e., other agents) which change the environment and contend for limited resources.

Furthermore, this problem is also challenging since it is subject to the same "infinite recursion" problem common to metareasoning approaches to bounded rationality (Russell, 1995) since the reasoning at the meta-level about sensing resource consumption requires information which requires sensing which might entail another level of metareasoning control.

1.3. Stateful Resources and the Observer Effect

Within the LRSP, one important subproblem arises when agents use *stateful* resources during sensing. We define stateful resources as resources whose behavior depends on their current state which changes with resource use. When such resources are used during sensing, *the act of sensing itself changes the state of the resource and thus alters and potentially distorts its own outcome (and future outcomes)*. We call this interesting phenomenon the **Observer Effect (OE)** after the similar phenomenon in the physical sciences. For example, in an intelligent user interface application such as the aforementioned intelligent software systems, the interface agent may need to prompt a user to elicit her preferences over a range of options. However, prompts are interruptions which can cause user frustration (Adamcyzk and Bailey, 2004), yielding worsened feelings about the system (Klein *et. al*, 2002) and potentially fewer quality responses. Thus in this case, the user is a stateful resource where her patience is the internal state while the responses constitute the user's behavior (i.e., resource behavior), and the change in the quality of the responses caused by the prompts is the Observer Effect.

From the perspective of the intelligent agent, the Observer Effect creates an important tradeoff in the ability of sensing to support agent reasoning guided by its knowledge; that is, a tradeoff between 1) satisfying the need for knowledge refinement from sensing with stateful resources, and 2) satisfying the need to avoid knowledge corruption due to distorted sensing outcomes caused by the Observer Effect. Solving this tradeoff, which we call the **Observer Effect Tradeoff Problem (OETP)** and is the specific focus of this thesis, entails 1) modeling the

impact of the Observer Effect on knowledge refinement as a function of both resource state and sensing activity, and 2) selecting sensing activities which provide refinement while avoiding knowledge corruption which can lead to bad decision making. This problem is a subproblem of the LRSP because not only are agents using resources during sensing, but resources in a state capable of producing good information during sensing become limited as the states of all resources used change over time through additional sensing. It is an important subproblem because the distortion caused by the Observer Effect is an additional type of cost incurred through resource usage during sensing which makes solving the LRSP more difficult. If an agent does not consider the Observer Effect when using stateful resources during sensing, the agent *cannot effectively solve* the LRSP because it won't be considering all costs of sensing which could prove detrimental to the agent in the long run. For example, pushing a stateful resource into a bad state can cause sensing distortion to snowball as continually using a resource while it is in a bad state could keep the resource in that bad state (e.g., continually interrupting an already frustrated user) which leads to more bad sensing in the future.

Note that considering the Observer Effect and its associated tradeoff is novel from prior work in bounded rationality which generally assumes no such side effects caused by using resources. For example, in traditional solutions to bounded rationality such as anytime algorithms, outcome quality is assumed to be monotonically increasing with resource usage (Zilberstein, 2008), which is valid when those resources are stateless. However, as postulated earlier, additional stateful resource usage during sensing can distort the sensing outcome, which leads to knowledge corruption, not refinement—thus non-monotonic outcome quality in resource usage.

1.4. Observer Effect MDP Solution Overview

To properly address the LRSP in agent sensing, we adopt the *active perception* perspective (Weyns *et. al*, 2004) to sensing where agents actively choose which sensing activities to perform, as opposed to reactively collecting whatever information is provided by the environment to the agent's sensors during its task-oriented actions. Specifically, this perspective provides a vehicle for making decisions about sensing activities to perform, given their need for resources and the consequences of their use. Within this methodology, we propose a decision-theoretic solution which models the problem of selecting sensing activities which require stateful resources as a Markov decision process (MDP), called the **Observer Effect MDP**. Using this model, we develop a sensing activity controller for choosing sensing activities capable of reasoning about and mitigating the Observer Effect by maximizing the expected knowledge refinement performed by the agent's sensing. Specifically, this controller models the relationship between resource state, the current knowledge of the agent, the possible sensing activities, and the value of knowledge refinement produced by sensing in order to select sensing activities which provide a maximal amount of expected knowledge refinement given the current states of resources, thereby avoiding knowledge corruption while meeting the agent's informational needs. Because such a model is difficult to construct *a priori* (e.g., due to a lack of knowledge by agent developers or due to frequent changes in the dynamic environment), we use reinforcement learning to learn such a controller online as the agent interacts with its environment.

This MDP-based solution to the OETP is appropriate given the dependence of resource behavior on its state and the fact that resource state is changed through agent sensing activities. We note that this solution follows a past tradition of using MDPs or partially observable MDPs (POMDPs) to model agent action selection in similar problems, such as metareasoning (Raja and

Lesser, 2007), as well as preference elicitation from users (Boutilier, 2002; Doshi and Roy, 2008). However, our solution is novel in that it considers the side-effects of using stateful resources during agent (sensing) activities (i.e., the Observer Effect and its tradeoff), while similar prior solutions do not.

1.5. Contributions

The research presented in this thesis makes several important contributions to the fields of artificial intelligence and multiagent systems, including:

1. An extension of bounded rationality as studied in artificial intelligence to the sensing activities of the agent through the Limited Resource Sensing Problem,
2. The formalization of the Observer Effect in agent sensing with stateful resources and its associated tradeoff with respect to knowledge refinement,
3. A decision theoretic solution called the Observer Effect MDP for modeling the effects of stateful resource usage during sensing and solving the OETP,
4. Simulation environments mimicking real-world scenarios and applications for studying the Observer Effect and solution approaches, and
5. A Java library offering various general artificial intelligence techniques which can be reused for other AI projects.

First, from a research perspective, as introduced in Section 1.2, our research investigates the Limited Resource Sensing Problem which occurs due to the bounds on agent rationality imposed by the relationship between limited resource and knowledge constraints. Second, we address a previously unstudied side-effect of using stateful resources during sensing which results in a type of distortion in activity outcome—the Observer Effect—not considered by traditional bounded rationality research in artificial intelligence. Third, we propose a novel solution enhancing prior bounded rationality research to handle this side-effect and its

associated tradeoff which can be potentially extended to other problems which exhibit similar characteristics.

From a software perspective, this thesis has also resulted in two simulation environments, including 1) a new Tileworld (Pollack and Ringuette, 1990) environment, similar to those commonly studied in multiagent systems research (e.g., Smith and Simmons, 2004; Weyns *et. al*, 2005), as well as 2) an extension of a previously published user preference elicitation simulation (Doshi and Roy, 2008) adding a model of user frustration and its effect on human-agent interactions. These simulations are both linked to real-world sensing problems and can be reused for other research involving similar applications. Finally, combined with the other research activities of the authors, it has also produced a Java-based library for general artificial intelligence techniques including various algorithms for reinforcement learning, search, and solving (partially observable) Markov decision processes, amongst other techniques.

1.6. Thesis Overview

The rest of this thesis is organized as follows. We describe the Limited Resource Sensing Problem and formalize the Observer Effect Tradeoff Problem within the context of bounded rationality in Chapter 2, followed by a description of our proposed Observer Effect MDP solution methodology in Chapter 3. Chapter 4 describes the various simulations used to study our methodology, as well as the setup of the various experiments used to validate our proposed solution in these applications. Chapter 5 presents the results of those experiments, including a discussion of the lessons learned from our research. In Chapter 6, we summarize relevant related work. In Chapter 7, we describe the future work we intend to perform as a result of this thesis. Finally, in Chapter 8 we conclude by summarizing the ideas presented in this thesis.

Chapter 2 The Problem – Limited Resource Sensing and Observer Effect Tradeoff

In this chapter, we first provide background on bounded rationality – the greater context within which our research is grounded. Next, we introduce the Limited Resource Sensing Problem (LRSP), the general focus of our research, within the context of bounded rationality. Then, we highlight the specific focus of this thesis: the Observer Effect Tradeoff Problem (OETP), a subset of the LRSP where agents use stateful resources for sensing. Finally, we formalize the OETP with mathematical notation for our solution in Chapter 3.

2.1. Bounded Rationality

2.1.1. Bounded Rationality Problem

In our research, we consider the **bounded rationality** perspective on agent (i.e., human, software, or hardware) intelligence. That is, the decision-making abilities of agents are limited by the resources and information available to each agent. These resources can be internal to an agent, such as computational ability and memory, as well as external to an agent, such as time, network bandwidth and latency for communications, a human user during human-computer interactions, and other application-specific resources which are pertinent to the environment in which the agent operates. This perspective is grounded in the economics (e.g., Conlisk, 1996; Rubinstein, 1998; Simon 1955, 1956, 1997) and cognitive psychology literature (e.g., Gigerenzer and Goldstein, 1996; Gigerenzer and Todd, 1999), arising from a descriptive theory intended to describe the manner in which humans make decisions. In contrast to the *classical rational man assumption* in decision problems, i.e. *perfect rationality*, where agents:

1) have perfect knowledge of the available choices,
2) have perfect knowledge of their preferences over choices (and/or the consequences/outcomes of those choices), often in the form of a utility function on choices and/or outcomes, and
3) have unlimited ability to calculate the optimal choice given what is feasible, available, and most preferred,

bounded rationality assumes that agents generally lack at least one of these three characteristics.

2.1.2. Bounded Rationality in Artificial Intelligence

Within the artificial intelligence community, deviations from these characteristics of perfect rationality are studied under different areas in the literature. First, the lack of knowledge about choices (or actions) available to agents is studied under the *planning* and *search* domains (e.g., Fikes and Nilsson, 1971; desJardins *et. al*, 1999; Weiss, 1999). By planning, an agent becomes capable of determining appropriate actions to achieve its goals, possibly by forming new actions through combinations of simpler, singleton actions. By searching, an agent determines which choices are available and feasible in different situations. By combining these abilities, agents can determine what choices (including combinations of actions) are available for its current decision.

Second, the lack of knowledge about preferences over choices (and/or the consequences of those choices) implies that agents do not always inherently know *which actions are* better *to take* in different situations, as opposed to *which actions it* can *take*. This problem is primarily studied in the *machine learning* literature, especially in *reinforcement learning* (Kaelbling *et. al*, 1996; Sutton and Barto, 1998) where agents learn the utility of various actions based on the current situation and the resulting outcome, allowing the agent to form

preferences based on an ordering on utility. However, this lack of knowledge about preferences can also result from a lack of knowledge about the current state of the environment and the situation the agent is facing. That is, if the agent cannot differentiate its current predicament from other scenarios, it is difficult for the agent to know which actions are most appropriate. This type of knowledge deficiency has primarily been studied under the areas of *perception* (e.g., Weyns *et. al*, 2004) and *belief revision* (e.g., Josang, 2001).

Finally, the problem of controlling reasoning when agents have only limited capacity to calculate optimal solutions, possibly due to real-time constraints or a lack of computational ability or memory, is primarily studied under the domain of *metareasoning* (e.g., Russell, 1995; Zilberstein, 2008). Here, agents tradeoff between the quality of their reasoning (e.g., proximity to optimality, breadth or depth of alternatives considered) against the amount of resources (e.g., time, computation, memory) required for computation. This can be done in multiple ways, including the use of *anytime algorithms* (e.g., Horvitz, 1987; Boddy and Dean, 1989; Zilberstein and Russell, 1993; Zilberstein 1996) which use a *performance profile* quantifying this tradeoff to create and revise the quality of reasoning until a satisficing decision is made (i.e., a decision whose outcome yields a sufficient expected utility). One important property of these performance profiles used by anytime algorithms is that the quality of reasoning is monotonically increasing with resource usage (Zilberstein, 2008). Thus, further reasoning (and the use of resources) yields no worse of a solution during the execution of the algorithm. If this assumption is *not* valid, however, the problem of selecting a stopping point for the algorithm becomes a much more difficult problem as the performance profile might have several local (not global) optima, meaning that continued reasoning could produce a worse outcome before a better one (if such exists).

Another approach to metareasoning follows a decision theoretic approach, modeling the selection of reasoning activities as a Markov decision process (MDP) (c.f., Section 3.2.1) where agents choose reasoning activities (which use computational resources) which maximize rewards based on the state of the agent. For example, Raja and Lesser (2007) consider a MDP which chooses reasoning control actions based on internal and external environment characteristics about tasks (e.g., current assignments and incoming rate, respectively). They use offline reinforcement learning to generate a policy for solving the MDP, which is encoded within the agent. Of note, this style of approach does not require monotonic performance profiles for resource use during sensing, instead optimizing reward functions of any shape.

One important subproblem to point out from the metareasoning literature is that the additional reasoning required to perform metareasoning takes resources away from the general reasoning performed by the agent. *Thus, the solution to the problem actually influences the problem itself.* However, although this subproblem closely resembles the original problem being solved (how to reason under limited resources), additional metareasoning is not necessarily a valid solution since this creates an infinite loop of reasoning control (Russell, 1995): metareasoning about reasoning requires further metareasoning to appropriately use computational resources, which requires more resource consumption, leading to further metareasoning, etc.

2.2. Limited Resource Sensing Problem

Together, techniques and solutions from these aforementioned areas of research provide artificially intelligent agents (both software and hardware) the means to reason and make decisions to solve problems, accomplish tasks, and achieve goals in complex environments where perfect rationality is not feasible. However, one drawback to the existing literature is that prior research primarily only considers limitations on the reasoning process itself. In

contrast, decision making is essentially the transformation of knowledge created through sensed information into rational decisions. Thus, the rationality of agents is bounded not only by the resources required during reasoning, but also the information provided to the agent. While the connection between bounded rationality and information gathering has been considered in the aforementioned research on perception, belief revision, etc., little attention[1] has been given to the effects of resource limitations on the process of gathering information. As with reasoning, information is not available for free from the environment and can require resources to extract. This includes resources for both the physical sensing activity (e.g., time, energy, network bandwidth/latency for communications, a human user to interact with), as well as resources for processing the raw data gathered into useful information (e.g., computation and memory). As pointed out by Conlisk (1996) from the economics perspective, bounded rational decision making thus becomes the product of both resource-limited *reasoning processes* and resource-limited *information gathering*.

Therefore, a parallel to the metareasoning problem of bounded rationality is the analog to sensing, which we term the:

> **Limited Resource Sensing Problem (LRSP)**: determining how to gather information used in decision making under limited resources required to gather that information, trading off resource consumption for information (and subsequent decision) quality.

Indeed, this problem shares many properties with the metareasoning problem: 1) a separate decision must be made to balance the tradeoff between the steps taken to support an activity and the quality of the activity's outcome, 2) the meta-level decision requires the same resources as the base-level activity – as a decision process, sensing control might require additional

[1] For a notable exception, please see the end of Section 2.3.3.

information to guide the sensing behavior of an agent, thereby consuming more sensing resources, and 3) adding further meta-behavior to control meta-level decisions induces an infinite loop of control and limited resource consumption.

It is also important to point out the interconnection between the resource needs of the reasoning and sensing processes within an agent's decision making. Not only does reasoning often require sensed information about the environment or the local agent before any decision can be made, but the activities of the two processes also share many of the same resources. For example, gathering information requires time, which limits the amount of time available for reasoning, given real-time constraints. Processing raw sensed data also requires computational resources which can prohibit simultaneous reasoning. Likewise, when sensing and reasoning form a continuous cycle, resource consumption by reasoning can limit availability for concurrent and future sensing activities.

2.3. Observer Effect Tradeoff Problem

2.3.1. Stateful Resources

However, despite their numerous similarities and interconnected relationship, the LRSP is not simply a subset of the better understood metareasoning problem. Consider another categorization of resources: stateful vs. stateless. We define *stateful* resources as resources whose behavior depends on some notion of state for the resource which changes over time based on resource usage, while *stateless* resources always behave the same way regardless of past usage. For example, in an environment with limited network capacity, sensing activities by agents called active monitoring which monitor network state introduce additional traffic into the network, thereby changing the underlying state (e.g., bandwidth) of the resource (Landeldt et. al, 2000). Depending on the network protocols used within the application (e.g., whether or

not they delay packet delivery based on earlier packet loss), different behaviors (e.g., latencies) can result for different network states. Likewise, interrupting human users to gather information about their preferences and activities, such as in a recommender system (e.g., Adomavicius and Tuzhulin, 2005) or personal information management (e.g., Chalupsky *et. al*, 2001; Myers *et. al*, 2007; Yorke-Smith *et. al*, 2009) application, can distract the user's cognitive processes (Klein *et. al*, 2002) leading to frustration (Adamcyzk and Bailey, 2004; Mark et. al, 2008) and affecting future interactions with the user. Therefore, one important subproblem within the LRSP is deciding how to select sensing activities which change the behavior of state-dependent resources used during sensing. Within the metareasoning problem, on the other hand, the computational resources (e.g., time, CPU cycles, memory) considered generally are stateless and thus their behavior does not depend on any notion of state – a CPU cycle will perform the same computations and memory will hold the same information regardless of previous resource usage. *Thus, beyond simple availability, resource history is important for LRSP but not for metareasoning*. In other words, unlike reasoning processes, sensing can *directly* change the environment surrounding the agent. Given this property of the LRSP not found in metareasoning, combined with the similarities of the two problems, we instead hypothesize that the type of metacognitive problem exemplified by the LRSP is instead a *superset* of the classic metareasoning problem.

2.3.2. Observer Effect

In this thesis, we focus on the subproblem of the LRSP where agents must select sensing activities which use such stateful resources. Because these activities change the state of the resource used during sensing, this change in state can impact the behavior of the resource, thus affecting the outcome of the sensing activity. In other words, using stateful resources during sensing produces a phenomenon where *the act of making an observation can distort the*

observation itself (and potentially future observations). We term this phenomenon the **Observer Effect** (OE) after a similar phenomenon in the physical sciences.

This distortion can occur for several reasons, depending on the influence of resource behavior on sensing outcomes. One example of the Observer Effect occurs when sensing accuracy depends on the behavior of the resource, resulting in a situation where *a sensing activity reduces its own accuracy.* In our earlier example of the stateful network resource, the additional traffic produced by active monitoring reduces bandwidth (Landeldt et. al, 2000) which increases congestion and latency (Fowler and Leland, 1991). As a result, observations produced do not reflect the state of the network when sensing is not performed. Thus, from the perspective of the LRSP, using stateful resources can cause a *cost with respect to information quality* (e.g., reduced accuracy) due to the Observer Effect.

Similarly, even if an agent is not monitoring its network but is communicating with other agents to share information, the reduced bandwidth from communications can cause information sent by other agents to be outdated and inaccurate by the time it is received. This problem can also arise when the agent uses a wireless sensor network to gather information about its environment even without interacting with other agents. For example, when using an energy-aware communication protocol (e.g., Arisha *et. al*, 2002; Shah and Rabaey, 2002), as the energy level of the nodes in the sensor network is diminished, the nodes will change their communication behavior. If the protocol chooses to use longer routes through energy-rich hops along the network (e.g., Arisha *et. al*, 2002), this longer route could cause the information transmitted to be stale before it is received by the agent responsible for refining knowledge and making decisions. In this latter example, the stateful resource is the wireless sensor network used by the agent whose behavior depends on its energy level.

Of note, this example illustrates that a stateless resource (energy) can affect agent sensing performance, seemingly contradicting our implicit assumption that only stateful resources lead to the Observer Effect. *However, this phenomenon only occurs when that stateless resource is part of the state of a stateful resource* (wireless sensor network) *used by the agent*. Specifically, the behavior of the *specific energy units* used by the sensor network do not change based on past usage. However, the behavior of the *sensor network* does change based on the amount of energy available to the sensors. Please note that this still differs from stateless resource usage during metareasoning as a stateless resource is part of a stateful resource used during sensing and thus contributes towards a side-effect from resource usage.

Another example of the Observer Effect occurs when the *quantity* of information provided by sensing depends on the behavior of the resource. For instance, in our earlier preference elicitation example, prompting the user to elicit her preference is an interruption which affects the user's feelings towards the system (Klein *et. al*, 2002) which can lead to less willingness to provide responses in the present and future. Similarly, in our energy-aware wireless sensor network example, if the network's protocol is unsuccessful and key sensors run out of energy, the network is not able to transmit as much information back to the agent. Thus, from the perspective of the LRSP, using stateful resources can also cause a *cost with respect to information quantity* (i.e., reduced quantity) due to the Observer Effect.

2.3.3. Observer Effect Tradeoff

Considering all of these examples, we see that the Observer Effect is an important challenge during resource-based sensing. Specifically, because the ultimate purpose of sensing is to gather information to refine the agent's knowledge which is used to produce good decisions during reasoning, the Observer Effect leads to the following difficult subproblem of the LRSP:

> **Observer Effect Tradeoff Problem (OETP):** determining how to gather information with stateful resources to refine knowledge used in decision making while balancing the tradeoff between 1) satisfying the need for knowledge refinement from sensing with stateful resources, and 2) satisfying the need to avoid knowledge corruption due to distorted sensing outcomes caused by the Observer Effect.

That is, as an agent chooses to perform more sensing activities to provide more information to support its reasoning, the benefits might be offset by a decrease in sensing performance caused by the Observer Effect from increasing resource usage, leading to wrong decisions and incorrect agent behavior. On the other hand, if an agent chooses to perform less sensing to reduce the Observer Effect by avoiding resource state change in the hope of maintaining sensing performance, the agent might end up with insufficient or outdated knowledge, again leading to wrong decisions and improper agent behavior. Thus, the Observer Effect places stress on the sensing activity selection of agents to create information used to refine the agent's knowledge, necessary to properly achieve its goals. Based on this tradeoff, we propose the following research hypothesis:

> **OE Hypothesis**: Considering the current state of stateful resources used during sensing will allow agents to effectively balance the Observer Effect Tradeoff, compared to approaches which do not consider resource state.

While this hypothesis seems intuitive, it is unclear whether it will actually hold in practice for several reasons: 1) the relationship between resource state and knowledge refinement could be difficult to model, especially without using a large number of observations to build such a model, and 2) incorrect actions taken to balance the tradeoff could make things worse, causing

the sensing distortion and knowledge corruption to snowball out of control (an **Avalanche Effect)**.

The distinction between the LRSP and traditional metareasoning due to the Observer Effect is important because state-dependent resource behavior implies that the performance profile of sensing is not always monotonically increasing, and in fact, is often nonmonotonic in general. In other words, while additional sensing activities which require more resource usage might lead to better knowledge refinement in some situations, this will not always be the case after an undesired resource state change. Thus, traditional metareasoning approaches such as anytime algorithms cannot be applied to the problem of making decisions about stateful resource usage during sensing. Of note, Zilberstein (1996; with Russell, 1993) has applied anytime algorithms to the problem of using stateless computational resources to *process* sensed information from raw sensory data whose performance is monotonic, but this does not apply to the physical activity of sensing nor stateful resource usage. Instead, we require a solution that handles non-monotonicity of sensing performance, such as the MDP-based approaches to metareasoning (e.g., Raja and Lesser, 2007).

2.4. OETP Formalization

Given our definition of the LRSP and the OETP, we now formalize the OETP in mathematical notation to setup our solution in the next chapter. The following definitions are summarized in Table 2.1. Formally, an agent over time must make decisions from a sequence $D = (d_i)$. Each decision d_i requires information from the environment $info(d_i)$. Such information is available either 1) in the agent's internal knowledge base K or 2) from a sensing activity selected from a set of possible activities $AC = \{ac_j\}$ performed on a source of information selected from a set of possible sources $S = \{s_k\}$. Together, the valid combinations of sensing choices (i.e., activity/source pairs) form a set of two-tuples $C = \{\langle ac_j, s_k \rangle\}$.

Table 2.1 OETP Definitions

Symbol	Definition
$\boldsymbol{K}$	The agent's current knowledge
$\boldsymbol{D = (d_i)}$	Decision sequence faced by the agent
$\boldsymbol{info(d_i)}$	Set of information required for decision d_i
$\boldsymbol{AC = \{ac_j\}}$	Set of sensing activities
$\boldsymbol{S = \{s_k\}}$	Set of information sources
$\boldsymbol{C = \{\langle ac_j, s_k\rangle\}}$	Set of choices of possible sensing activity/source pairs
$\boldsymbol{R = \{r_l\}}$	Set of stateful resources
$\boldsymbol{RN(ac_j, s_k)}$	Set of resources needed by $\langle ac_j, s_k\rangle$
$\boldsymbol{St}$	Set of all possible resource states
$\boldsymbol{St_{r_l}}$	Set of possible states of r_l
$\boldsymbol{\sigma(r_l)}$	Current state of r_l
$\boldsymbol{\delta(r_l, \sigma, ac_j, s_k)}$	State transition function for r_l
$\boldsymbol{info(ac_j, s_k, NR, \sigma)}$	State-dependent set of information provided by $\langle ac_j, s_k\rangle$
$\boldsymbol{\otimes}$	Knowledge refinement operator
$\boldsymbol{K'}$	Refined knowledge from information provided by sensing
$\boldsymbol{KR(ac_j, s_k, NR, \sigma, K_t, d_t)}$	State-dependent value of knowledge refinement produced by $\langle ac_j, s_k\rangle$ with respect to d_i
$\boldsymbol{V(K, d_t)}$	Value of knowledge with respect to d_i

Performing these sensing choices requires the use of stateful resources $R = \{r_l\}$ with possible states $St = St_{r_1} \cup St_{r_2} \cup \ldots \cup St_{r_{|R|}}$, where St_{r_l} is the set of possible states for resource r_l. The current state of a resource is denoted by $\sigma(r_l)$ with $\sigma: R \rightarrow St$. The set of specific resources needed by a sensing choice $\langle ac_j, s_k\rangle$ are denoted by $RN(ac_j, s_k)$ with $RN: AC \times S \rightarrow 2^R$. Using each stateful resource potentially changes the state of the resource, depending on the activity and source chosen. This transition function is represented by $\delta(r_m, \sigma, ac_j, s_k)$ with $\delta: R \times St \times AC \times S \rightarrow St$. Finally, performing a sensing activity ac_j on an information source s_k produces $info(ac_j, s_k, RN, \sigma)$ depending on the current state of the resources used by activity/source pair due to the Observer Effect. This information is used to refine the knowledge of the agent through the domain dependent knowledge operator $\otimes$ (for example, c.f., possibility updates and belief state updates in Sections 3.2.2, 4.1.2, or 4.2.2). Putting everything together, we see

that performing a sensing activity results in both a resource state change and knowledge refinement:

$$\sigma'(r_l) = \delta(r_l, \sigma, ac_j, s_k) \quad (1)$$

$$K' = K \otimes info(ac_j, s_k, NR, \sigma) \quad (2)$$

Recall that the purpose of solving the OETP is to balance 1) the need for knowledge refinement through additional sensing to support decision making, and 2) avoiding knowledge corruption due to distorted sensing outcomes caused by the Observer Effect. To make choices about sensing activities that lead to knowledge refinement and corruption, we need a way to quantify the changes in knowledge produced by sensing. Specifically, the agent is concerned with the *value* of a change in knowledge, we measure as the difference between the value of the revised and previous knowledge with respect to the current decision d_i using the function $KR\colon AC \times S \times 2^R \times St \times K \times D \to \mathbb{R}$:

$$KR(ac_j, s_k, RN, \sigma, K, d_i) = V(K', d_i) - V(K, d_i) \quad (3)$$

where $V(K, d_i)$ with $V\colon K \times D \to \mathbb{R}$ measures the domain-dependent value of the subset of an agent's knowledge necessary for making a given decision. For example, this value of knowledge might be the confidence the agent has in the correctness of its knowledge pertinent to the decision which is increased or decreased after considering new information collected during sensing, or the possibility the agent ascribes to the correct state of the environment (c.f., Section 4.1.1).

Given these definitions, we can describe the primary goal of each agent under the OETP:

Given K, d_i, RN, σ, C

Choose $\text{argmax}_{(ac_j, s_k) \in C} KR(ac_j, s_k, RN, \sigma, K, d_i)$ (4)

Until $info(d_i) \subseteq K'$ (5)

In words, the goal of the agent is to select sensing choices (i.e., activity/source pairs) which maximize the value of refinement in agent knowledge with respect to its current reasoning decision until the agent has the knowledge it needs to successfully make a decision in order to achieve its goals. In doing so, the agent effectively balances the Observer Effect Tradeoff between the need for knowledge refinement to make decisions and the need to avoid knowledge corruption due to the Observer Effect. In the next chapter, we provide a methodology for making such sensing choices in order to solve the OETP.

Finally, we note that other constraints might be added to the selection process, depending on the application and environment. For instance, in applications where specific states of a resource should be avoided, where these states can be predicted by those which provide expected knowledge distortion, an additional constraint can be included. Here, the agent also stops sensing no when expected refinement is possible, modeled as:

$$KR\left(ac_i, s_j, NR, \sigma, K, d_i\right) \leq 0 \qquad (6)$$

For example, this constraint might be useful in a user preference elicitation application (c.f., Section 4.2) where the user's state is her frustration level increased by interruptions (Adamcyzk and Bailey, 2004; Mark et. al, 2008) for sensing. Here, too high of frustration should be avoided or the user might lose faith in the system (Klein *et. al*, 2002) and quit wanting to use the system altogether.

Chapter 3 The Solution – Observer Effect MDP

In this chapter, we present our proposed decision theoretic solution to the Observer Effect Tradeoff Problem (OETP) introduced in Chapter 2. We begin by providing background on active perception, including the specific framework we adopt for use as a vehicle for sensing activity selection. Next, we describe Markov decision processes (MDP) and demonstrate how the OETP formalization defined in Section 4 of Chapter 2 can be mapped to a MDP which we brand the Observer Effect MDP. Afterwards, we detail how a sensing activity selection controller following the Observer Effect MDP can be developed through reinforcement learning in order to solve the OETP. Finally, we highlight the novelty of this proposed solution.

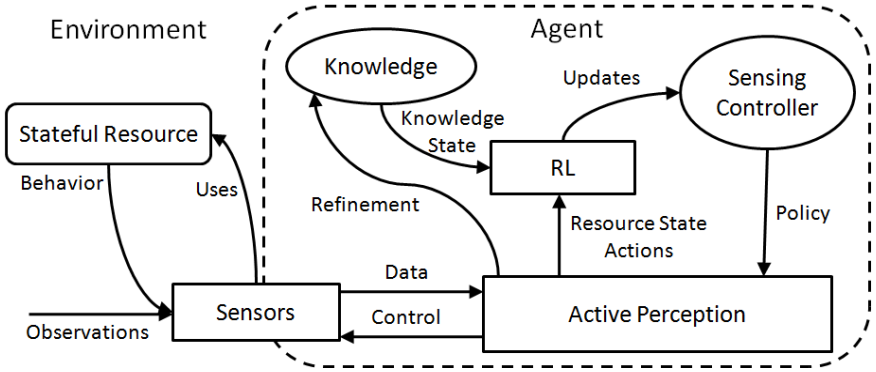

Figure 3.1 Methodology Overview

The overall solution methodology is summarized in Figure 3.1. In short, an agent uses sensors to collect information (in the form of observations) from the environment. Sometimes, this requires the use of stateful resources, and the observations collected depend on the behavior of the resources. To control the agent's sensors and its use of stateful resources, the agent uses active perception which also processes the data from the sensors to revise the agent's knowledge. To make such control decisions, the agent models the sensing activity selection process as an Observer Effect MDP and uses reinforcement learning to build and revise a sensing controller. This sensing controller is responsible for creating a policy for active perception that chooses sensing activities while balancing the OETP.

3.1. Active Perception

Within the context of the LRSP, an agent is tasked with balancing 1) the need for information of sufficient quality to refine its knowledge to support its reasoning with 2) the costs of resource consumption during sensing (which occur whether or not the resources used are stateful or stateless). Balancing such a tradeoff entails making decisions about what sensing activities to perform. Thus, an agent cannot simply rely on observations produced by task-level actions and instead must explicitly reason about sensing. Therefore, in our research in solving the LRSP, we adopt the **active perception** perspective (e.g., Bajcsy, 1988; Floreano and Mondada, 1994; Weyns *et. al*, 2004; So and Sonenberg, 2009) to sensing which focuses on making such decisions. As part of our research, we also extend existing active perception research to provide resource-aware decisions in order to handle the LRSP.

Specifically, we follow the domain-independent active perception framework proposed by Weyns *et. al* (2004), which separates the perception process into three steps: 1) sensing , 2) interpretation, and 3) filtering. First, during the sensing step, an agent uses physical sensors to extract raw observations from its environment. The specific observation made by the agent depends on 1) the foci of its sensors (i.e., sensing activities chosen), and 2) the perceptual laws of the environment, which determine what the agent is actually able to observe. For example, in a network monitoring scenario (e.g., Landeldt et. al, 2000), an agent might choose to monitor a specific region of the network by sending packets to another agent in that region. The perceptual laws in this scenario include the routing behavior of packets in the network (e.g., the number of paths the packets take through the network to reach their destination), as well as the limitations in the transmission of packets, such as message buffer behavior and transmission rates. The second step, interpretation, uses a set of domain-dependent descriptions to transform the raw data collected by the sensors into an internal representation understood by

the agent's reasoning process. In our example, the agent transforms packet responses from the other agent into important metrics, such as packet latency, network bandwidth, etc. Finally, to avoid overloading the agent's reasoning with more information than required (as not all information perceived by the agent's sensors is necessarily relevant to the current or future decisions), the agent applies a set of chosen filters to select only the information it needs. In our example, the agent might select a filter to only use latency measurements.

Within this framework, the reasoning process controlling active perception centers on two decision processes: 1) selecting the set of foci for the sensing step which determines exactly what sensing activities the agent will perform, and 2) selecting the set of filters during the filtering step which determine what information is passed from perception to knowledge refinement for reasoning. In our research, we are interested in the former as it is the foci (i.e., sensing activities) chosen by the agent that determine how *stateful* resources will be used during sensing. Furthermore, by carefully selecting sensing activities, the agent can potentially reduce the amount of filtering necessary by only choosing sensing activities which provide the exact information needed by the agent (if possible), thereby being efficient during sensing (an extension to our research we intend to investigate in the future).

3.2. Observer Effect MDP

In constructing a controller for selecting which sensing activities to perform, where those activities require stateful resources, we assume that the behavior of the stateful resources used is stochastic, a common assumption about the environment in multiagent systems. This assumption allows us to consider resource behavior as a *stochastic process*. This is an especially valid assumption in multiagent systems as actions by other agents which influence the state of resources might not be observable by each agent, thus making the resources' behavior appear random even if it is truly deterministic. Therefore, from the perspective of each agent, the

problem of making choices about sensing activities which change the state of such a process is itself a *stochastic decision process*. To simplify the model, we further assume that the behavior of a resource depends only on its current state, allowing us to model this decision process as a Markov decision process. Without this assumption, an agent could have to account for any number of prior states in a resource's history, making the model very complex and difficult to solve. Thus, as a first-step solution, we assume state independence from distant history and will later investigate how to adapt our solution to relax this assumption when necessary.

3.2.1. MDP Background

Specifically, a **Markov decision process (MDP)** (Kaelbling *et. al*, 1998) models a stochastic decision process as a tuple $\langle S, A, T, R \rangle$ where $S = \{s\}$ is a set of (*fully* observable) states of the environment, $A = \{a\}$ is a set of choices (i.e., actions) available to the agent, and $T(s, a, s') \in [0,1]$ is the probabilistic transition function representing the likelihood that the environment changes from state s to s' if the agent makes choice a, that is $T(s, a, s') = P(s' \mid s, a)$, and $R(s, a) \in \mathbb{R}$ is a function modeling the reward of making a choice dependent on the current state of the process.

Given a MDP modeling the decision process facing the agent, the goal of the agent is to build a policy π mapping states to choices which optimize the rewards received by the agent for its choices. The values of states used to compute policies are represented by a set of Bellman equations optimizing discounted, expected future rewards:

$$V^*(s) = \max_{\pi} E[\sum_{t=1}^{\infty} \gamma^t r_t] = \max_{a \in A} R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s'), \forall s \in S \quad (7)$$

where r_t is the reward for making choice $\pi(s)$ by following policy π in state s at time t and $\gamma \in [0,1]$ is a discount factor for weighting the consideration of expected future rewards. From these values, the agent computes an optimal policy using the value-iteration algorithm which solves the set of equations:

$$\pi(s) = \text{argmax}_{\text{a}}\, R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s'), \forall s \in S \quad (8)$$

by iteratively computing the (immediate *and* discounted future) value of making each choice in each state, represented by $Q(s,a)$ and used to represent the value of $V(s)$ if a particular choice is made:

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V(s'), \forall s \in S \quad (9)$$

$$V(s) = \max_{\text{a} \in \text{A}} Q(s,a) \quad (10)$$

until the $V(s)$ calculations converge, a select number of iterations have occurred, or some other stopping condition is met, then choosing the best choice for each state as the policy choice for that state. This policy then serves as a controller for guiding the agent's choices.

When the state of the environment is *not* directly observable (i.e., is hidden from the agent), a more appropriate model is a *partially*-observable Markov decision process (POMDP) (Kaelbling *et. al*, 1998), which augments a MDP to form a six-tuple $\langle S, A, O, T, \Omega, R \rangle$ where S, A, T, and R are as in the MDP discussed above, $O = \{o\}$ is a set of observations produced when a choice is made, and $\Omega(s', a, o) \in [0, 1]$ is the probabilistic observation function representing the likelihood that o is observed after making choice a leading to (hidden) state s', that is $\Omega(s', a, o) = P(o \mid s', a)$. In this decision process, the hidden state of the process must be estimated based on the observations produced by choices. Here, the agent maintains a belief state vector $b(s) \in [0, 1]$ describing the likelihood that the current state of the process is each $s \in S$. This vector is updated through belief revision based on recent observation o after choice a:

$$b'(s') = \frac{1}{Z} \Omega(s', a, o) \sum_{s \in S} T(s,a,s') b(s) \quad (11)$$

where Z is a normalization factor to insure that the new belief values both remain between 0 and 1 as well as sum to 1 (since this vector represents a probability distribution over states), and b' is the new belief state vector.

Since in partially-observable environments the agent does not know *exactly* which state the process is actually in, it must now consider the possibility of each possible state in its belief state. This makes developing a policy for a POMDP much more complicated and computationally expensive than building a policy for a MDP because policy and value iteration for a POMDP would require creating a policy over an infinite number of belief states. Different solutions for POMDPs handle this problem in different ways. For example, some compute a policy offline when more time and computational power can be applied to building a policy, such as creating decision trees where branches are observations and choices of maximal expected value are nodes (Kaelbling *et. al*, 1998). Another type of offline approach further tries to avoid the computational complexity of finding an exact solution, instead *approximating* regions of the value function through approaches such as Point-based Value Iteration (PBVI) (e.g., Pineau *et. al*, 2003; Doshi and Roy, 2008) which prescribe a choice based on which region is closest to the current belief state of the agent. More recently, researchers have worked on online approaches for applications where an agent should develop new policies as it interacts with the environment, as opposed to offline like those already discussed. The tradeoff with using an online approach, however, is that the agent has even fewer computational resources and possibly harder real-time constraints for calculating a policy than in an offline approach. These online approaches (Ross *et. al*, 2008) include limiting decision trees to a short depth (since the complexity of building such trees is exponential in the depth of the tree), as well as heuristic search (Ross and Chaib-draa, 2007) through the possible paths of the decision tree.

Table 3.1 Transformation from OETP to Observer Effect MDP

Observer Effect MDP	OETP Transformation	Description
$\boldsymbol{s} = \langle \boldsymbol{R_s}, \boldsymbol{K_s} \rangle \in \boldsymbol{S}$	$R_s = \langle \sigma(r_1), \sigma(r_2), \dots, \sigma(r_{\lvert R \rvert}) \rangle$ $K_s = \text{state}(K)$	The **sensing states** are combinations of resource states and knowledge state.
$\boldsymbol{a} \in \boldsymbol{A}$	$a = \langle ac_j, s_k \rangle \in C$	The **active perception choices** are the valid sensing activity/source pairs.
$\boldsymbol{T}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$	$\delta(r_l, \sigma, ac_j, s_k)$ K'	**Sensing state changes** depend on the changes in resource and knowledge states due to a chosen sensing activity/source pair.
$\boldsymbol{R}(\boldsymbol{s}, \boldsymbol{a})$	$KR(ac_j, s_k, RN, \sigma, K, d_i)$	The **reward** for making choices given the current sensing state is the value of knowledge refinement as the result of sensing.

3.2.2. Observer Effect MDP Mapping

Given this description of MDPs, we can naturally transform the Observer Effect Tradeoff Problem formalization described in Section 2.4 into a MDP–which we term the **Observer Effect MDP**–as follows (summarized in Table 3.1). We define the current *sensing state* $s \in S$ as tuple $\langle R_s, K_s \rangle$ which is a *factored* state consisting of the current state of all resources R_s and the current state of the agent's knowledge K_s from the OETP, with

$$R_s = \langle \sigma(r_1), \sigma(r_2), \dots, \sigma(r_{\lvert R \rvert}) \rangle \quad (12)$$

$$K_s = \text{state}(K) \quad (13)$$

where $\text{state}(K)$ is an application-specific description of the current knowledge of the agent. For example, in a user preference elicitation scenario, the resource state could represent the user's frustration, and the knowledge state could represent the amount of evidence supporting the agent's belief about the user's preference. In our model, we include both resource state and knowledge state in the sensing state because both play a role in the value of knowledge revision (Eq. 3): resource state through the Observer Effect and knowledge state through the improvement in knowledge. We define the *active perception choices* A in the MDP to be the set

of possible sensing activity/source pairs C in the OETP. In our example, the choices are the different ways the agent can gather information about the user's preference (e.g., asking the user directly, observing her behavior in a task). Furthermore, we define the transition function $T(s, a, s')$ in the MDP as a probability measure over the changes to resource state $\sigma(r_l)$ and knowledge state state(K') by a sensing activity $\langle ac_j, s_k \rangle$ determined by equations (1-2). In our example, interrupting the user to ask about her preference increases frustration (Adamcyzk and Bailey, 2004; Mark et. al, 2008), changing her state. Based on the user, her frustration could increase by different amounts at different times (e.g., one time by a little and another by much) for the same interruption. Also, the information gathered through the interruption revises the agent's knowledge, changing the knowledge state. Finally, we define the $R(s, a)$ reward function in the MDP as the value of knowledge revision we are concerned with in the OETP: $KR\left(ac_j, s_k, RN, \sigma, K, d_i\right)$ (Eq. 3). For our example, this could be the increase in the possibility (Josang, 2001) the agent ascribes to the correct preference of the user.

Using this approach, we see that building a policy for the MDP optimizes the reward function based on making choices given the current state of the process. Since our problem mapping sets the reward function to be the value of knowledge refinement, following the MDP's policy optimizes the value of knowledge refinement function $KR\left(ac_j, s_k, RN, \sigma, K, d_i\right)$ given a current reasoning-level decision d_i from the OETP. This is made possible due to the fact that the reward function explicitly considers the current state of the process and the action chosen, which is necessary for calculating the expected value of knowledge refinement due to the Observer Effect. Thus, the policy created by the MDP provides the necessary means to construct a controller for choosing sensing activities to balance the OETP. However, as the OETP has a set of constraints on when to stop choosing sensing activities, we require the following in the controller: the controller should stop if 1) more knowledge refinement is unnecessary, or 2)

positive knowledge refinement is not expected to occur (if this second optional constraint is included, c.f. Section 2.4)). Both of these "safeguards" are in place in order to avoid knowledge corruption. Otherwise, the controller should simply follow the MDP's policy.

Considering the relationship between this $R(s,a)$ reward function and resource usage in sensing, $R(s,a)$ can be viewed as a state-dependent sensing performance profile mapping sensing activities (corresponding to resource usage) into sensing performance (value of knowledge refinement). As this performance profile is optimized by solving the corresponding MDP, it is not restricted to be monotonic. Thus, the performance profile can model the Observer Effect, matching the solution requirement set forth in Section 2.3.3.

3.3. Learning a Sensing Activity Controller

When an explicit, parameterized Observer Effect MDP model of the active perception decision process is not provided by the agent designer, an agent must instead *learn* how to make sensing activity choices. This lack of an *a priori* model might occur due to a lack of prior knowledge about the domain or because the underlying environment is inherently dynamic and the MDP model's parameters change over time. To perform such learning, we turn to the field of reinforcement learning.

3.3.1. Reinforcement Learning Background

Reinforcement learning (RL) (Kaelbling *et. al*, 1996; Sutton and Barto, 1998) is a process by which agents learn how to act in an environment by optimizing the outcome of performing actions, where outcomes depend on the underlying state of the environment, based on some response signal from the environment signifying those outcomes. Often, this response is an immediate reward or cost, and the outcome learned is the (possibly discounted future) utility of performing the chosen action. Using the learned outcome function, agents can then build an

optimal policy mapping states of the environment to the best actions to take in order to achieve desired outcomes (e.g., utility maximization). Looking at reinforcement learning from the perspective of MDPs, we see that the goal of learning is to build the policy π with *experienced* rewards used to (explicitly or implicitly) *learn* a reward function $R(s,a)$ necessary for creating the policy, as opposed to simply using a *given* reward function when learning is unnecessary.

RL algorithms generally come in two types (Kaelbling *et. al*, 1996): 1) **model-based RL**, where the agent first learns an explicit model of the environment (often an MDP) and then uses that model to determine an optimal action policy, and 2) **model-free RL**, where the agent learns mappings of state/action pairs to outcomes to guide policy creation without building an explicit descriptive model of the environment.

Please note that both types of learning generally start with knowledge about the *structure* of the environment, including the relevant environment states and potential actions the agent can take (i.e, choices it can make). The difference between the two types of RL is whether or not they learn the *parameters* to the underlying environment model (e.g., state transitions in a MDP) used to construct the controller. Specifically, model-based RL algorithms *do* learn model parameters, while model-free RL algorithms *do not*.

For example, one simple yet popular model-based RL algorithm is RMax (Brafman and Tennenholtz, 2002), a polynomial-time, probably approximately correct (PAC) RL algorithm which uses evidence counting to learn the parameters of the underlying MDP. Specifically, it starts by assuming all states transition to the same fictitious state with probability 1 and all $R(s,a)$ values are some maximal value $R_{\max}$. By assuming only transitions to a fictitious state, the algorithm narrows the space of learning until the most relevant transitions are observed, after which it learns for those transitions. On the other hand, assuming a maximal $R_{\max}$ value for the rewards encourages exploration of the state/action pairs not yet encountered because

an agent tries to maximize its rewards. This is because state/action pairs that have been encountered will have updated values which cannot be greater than this maximal value and therefore will not be immediately exploited. To learn while interacting with the environment, the agent counts the number of transitions from each state to every other state dependent on each action. Once the sum of the counts from a particular state exceeds a threshold, the state transitions from that state are updated using these counts as likelihoods. Furthermore, the $R(s,a)$ values are updated after the first encounter of the state/action pair, assuming that the reward function is static (for an extension for nonstatic rewards, c.f. Section 3.3.2). Thus, this algorithm learns not only the reward function, but also the transition function for the underlying MDP of the environment, making it a model-based algorithm. Learning these transition probabilities allows the agent to directly solve the MDP to build a policy maximizing its rewards which serves as a controller for making choices about actions.

One simple yet popular model-free RL algorithm, on the other hand, is Q-Learning (Watkins, 1989). Specifically, Q-Learning learns an approximation of the underlying MDP's utility function in tabular form based on rewards received from the environment. Similar to the value iteration algorithm described previously, it maintains a $Q(s,a)$ entry for every state/action pair representing an approximation of the utility of taking action a in state s. However, instead of updating $Q(s,a)$ iteratively given a reward function $R(s,a)$, Q-Learning instead updates its $Q(s,a)$ approximation whenever the state/action pair is encountered and an explicit reward is received from the environment. These updates follow the formula:

$$Q'(s,a) = \big(1-\alpha\big)Q(s,a) + \alpha[r(s,a) + \gamma \max_{\mathrm{a}'\in\mathrm{A}} Q(s',a')] \qquad (14)$$

where α is the learning rate determining how much weight to assign to updates (i.e., how *aggressively* to learn), $r(s,a)$ is the explicit reward just received for taking action a in state s, and γ is again a discount factor for future values. Note that this formula does not consider the

probabilities of transitioning between states s and s' after taking action a and instead waits until it knows the next state and only considers the best action maximizing the myopic future reward for the next state of the environment. Thus, Q-Learning does not require any parameter information about the environment, including a MDP model of the decision process. Instead, it just uses rewards actually received to learn the expected, discounted, myopic utility which can subsequently be used in a controller which selects actions that maximize this learned utility.

As in MDPs and POMDPs, when the environment is partially observable, a different set of algorithms are required to provide the agent with the capability to learn a controller for selecting actions. These algorithms are called partially observable reinforcement learning (PORL) and also come in model-based and model-free variants. Here, model-based PORL algorithms again *learn a model* of the environment– often a PODMP (e.g., Ross *et. al*, 2007)– which is used to generate a controller for selecting actions (e.g., using the POMDP solutions described at the end of Section 3.2.1). Model-free PORL algorithms, on the other hand, *directly learn a controller* which directly maps observations or belief estimates to actions to perform (e.g., Wierstra, 2007).

3.3.2. Reinforcement Learning for the Observer Effect MDP

As alluded to earlier, given our assumption that an explicit, parameterized model of the Observer Effect MDP is not provided to agents who use stateful resources during sensing, agents must use reinforcement learning to develop a controller for choosing sensing activities. While building such a controller, the agent can use either model-based RL to actually learn the parameterized model for the Observer Effect MDP (e.g., state transition probabilities) then solve the MDP to generate its controller, or the agent can use model-free RL to learn the controller directly. We conjecture that either type of RL algorithm is acceptable (c.f., Chapters 4 and 5 for experiments testing this conjecture), but note that one type might be more appropriate

depending on the specific domain and application to which the agent is deployed. For example, if the environment is very dynamic, using a model-free RL algorithm might be more appropriate than a model-based RL algorithm as the parameters learned by the latter might become outdated as the environment changes, decreasing the usefulness of the learned environment model and potentially leading to improper decisions by the controller. In such a scenario, model-free algorithms can better adapt to the dynamic environment as they do not need to "unlearn" as much outdated information. If the environment is more static, however, leveraging the additional environment model learned by a model-based RL algorithm could result in better proactive behavior as the agent is able to consider the probability of each state transition before taking an action, as opposed to waiting until after the action to observe the exact transition taken (e.g., in Q-Learning (Watkins, 1989)) and thus potentially better long term reward maximization.

However, although the two types of algorithms differ in whether or not they learn a model of the Observer Effect MDP parameters, one requirement is that whatever RL algorithm is chosen, it must learn the reward function $R(s, a)$ which represents the expected value of knowledge refinement of a given sensing activity/source pair. Without learning the reward function, the agent will not be capable of considering the Observer Effect (whose effect on knowledge is captured in the reward function) during its sensing activity selection and thus cannot solve the OETP. How the agent performs this learning depends on the specific RL algorithm chosen. In our previous RMax (Brafman and Tennenholtz, 2002) example, an agent traditionally only updates the expected reward for a given state/action pair the first time a reward is received, expecting the reward value for a state/action pair to be static and consistent. As the value of knowledge refinement might be dynamic, we can extend the reward update in RMax to follow a similar counting-based learning strategy as it uses for learning state transition

probabilities to learn a stochastic reward function. In our Q-Learning (Watkins, 1989) example, on the other hand, the agent can learn the reward function in one of two ways: 1) *implicitly* by just learning the $Q(s,a)$ values which approximate the sum of current and myopic future rewards, or 2) *explicitly* by only considering immediate rewards while learning (i.e., $\gamma = 0$), which reduces Q-Learning to:

$$R'(s,a) = (1-\alpha)R(s,a) + \alpha r(s,a) \quad (15)$$

representing a geometric averaging scheme for learning rewards where $R(s,a)$ replaces $Q(s,a)$ since only immediate rewards are considered instead of discounted, myopic utility.

Recall that in the Observer Effect MDP for solving the OETP, the reward function $R(s,a)$ is the knowledge refinement value function $KR(ac_j, s_k, NR, \sigma, K, d_i)$ calculated as (Eq. 3) (c.f., Section 2.4). In a particular application of the Observer Effect MDP, the specific measure for the value of knowledge denoted by $V(K,d_i)$ in (Eq. 3) used to calculate the value of knowledge refinement is dependent on the knowledge framework used by the agent, as well as the domain of the application. For example, if the agent maintains a possibility measure over its beliefs (e.g., Josang, 2001), the value of knowledge refinement (if positive, or corruption if negative) from a sensing activity is the increase (or decrease, respectively) of the possibility the agent assigns to the true state of the environment (calculated once the correct belief is later known). Similarly, if the agent instead models its task-level decision process (separate from the active perception sensing activity selection process) as a POMDP (Kaelbling *et. al*, 1998), the value of refinement (or corruption) of agent knowledge can be measured as the increase (or decrease) in the belief state value for the true state of the environment (again calculated once this state is known by the agent). Concrete examples of how this reinforcement learning process works in the Observer Effect MDP for various RL algorithms in specific applications is provided in Section 4.1.2 and 4.2.2 when the various simulation environments used in our experiments are detailed.

3.4. Solution Novelty

To conclude this chapter, we briefly highlight the novelty of our proposed solution methodology, which is two-fold. First, it extends prior work in bounded rationality (e.g., Raja and Lesser, 2007) using a MDP and RL to choose activities which use resources and ultimately impact the reasoning of the agent (ours through knowledge refinement). The extension comes from the fact that our methodology considers the impact of using stateful resources which produce side-effects such as the Observer Effect, as opposed to simply using stateless resources with no such side-effects. Furthermore, we believe that this extension could be important to determining how to handle other types of side-effects from resource usage outside of sensing, such as interacting with a human user during the reasoning portion of mixed-initiative systems (e.g., Ferguson and Allen, 2007). We intend to investigate this avenue of research as future work.

Second, our proposed solution represents a foci selection mechanism within active perception which considers the impact of using stateful resources during sensing. In the original work in the active perception framework adopted by our solution (Weyns *et. al*, 2004), the specific mechanism for foci selection is not specified but is left open to the specific problem being solved. Thus, we contribute one such selection mechanism for use with stateful resources. Furthermore, this mechanism offers a starting point for our more general research into using active perception as a vehicle for sensing activity selection balancing the need for information to refine knowledge for agent reasoning and the costs of resource use under the LRSP.

Chapter 4 Experimental Setup – MineralMiner and UserRec

In this chapter, we detail the experimental setup used to investigate and validate our Observer Effect MDP solution methodology from Chapter 3 for solving the Observer Effect Tradeoff Problem (OETP), a subproblem of the Limited Resource Sensing Problem (LRSP), described in Chapter 2. Specifically, we consider two simulation environments, both implemented using the Repast Agent Simulation Toolkit (North *et. al*, 2006): 1) MineralMiner, a fully-observable environment, and 2) UserRec, a partially-observable environment. For both simulation environments, we 1) describe the environment, 2) detail how we apply the Observer Effect MDP to the environment, including the specific RL algorithms used, and 3) present the objectives of our experiments, along with the specific parameters defining the experimental setup. Finally, we provide a comparison of the two environments highlighting their unique properties.

4.1. MineralMiner: A Robotic Mining Simulation

4.1.1. Environment Description

The first simulated environment considered in our experiments is *MineralMiner*, a modified Tileworld (Pollack and Ringuette, 1990) similar to Packet-World (Weyns *et. al*, 2005) and RockSample (Smith and Simmons, 2004). In this environment, multiple agents are randomly placed in a 2D grid which they must navigate. Specifically, competitive agents are assigned overlapping collection tasks over time with firm deadlines to find and collect rare minerals (gold, uranium, and silver) from various mines randomly distributed throughout the grid. A task is considered completed when enough of a specified mineral has been collected and deposited in a special cell called the *base*. To navigate the grid, agents can move in each cardinal direction which requires the consumption of limited energy from the agents' batteries. To avoid running

out of energy, agents must recharge their batteries at a single *recharge station* in the grid, an action which converts uranium into energy. Because the recharge station is essential to maintaining the agents' ability to perform tasks within the grid, it emits a special radio gradient allowing the agents to navigate towards it from any location. To conserve energy when it has no tasks, the agent can also perform a *wait* action and do nothing. In order to extract minerals from a mine, the agent must perform a *collection* action, which requires a *drill* which must be rented from a *toolshed* also located in the grid and later returned.

Overall, an agent operates by first locating the necessary landmarks (base, toolshed, recharge station) required to operate in the grid, then finding mines to provide the minerals necessary for its current tasks. Once relevant mines are identified, the agent collects the necessary minerals and returns them to the base. An agent prioritizes its actions based first on its need to either recharge its battery or rent tools, then on which tasks expire first. If an agent fails to accomplish a task before its deadline, that task is discarded and any minerals collected are saved for later tasks.

Agents begin with no knowledge of the environment and must use sensing actions which can be fit into two categories to explore their surroundings. First, adapted from Packet-World (Weyns *et. al*, 2005), agents perform sensing activities for exploring the grid and finding the recharge station which require only agent energy, a stateless resource whose usage does not distort the outcome of sensing. Second, more interesting to our research, agents also perform three sensing activities with a stateful resource (called an *electronic microscope*, also rented from the toolshed) for testing the contents of a mine: 1) *basic mine test*, 2) *advanced mine test*, and 3) the aforementioned *wait*. The two different tests require different amounts of microscope energy (powered by a slowly self-recharging battery) and produce different levels of accurate response (c.f., Table 4.1), where the advanced test requires more energy but is more

accurate. The specific amount of energy required is random up to a maximum value, representing an unknown amount of work required to perform a test. Furthermore, each test is also affected by the current energy level of the microscope, where the ability of the microscope to perform tests decreases with energy usage. However, the wait action allows the microscope to recharge, increasing the accuracy of the next test.

Information provided by sensing is used to refine the agent's knowledge about the grid, including the contents of each cell (e.g., landmark, supply, obstacle, empty) as well as the type of mineral in each supply. Agent knowledge takes the form of evidence-based opinions (Josang, 2001) for each possible cell contents and is refined by counting evidence in favor of or against each possibility based on observations made during sensing. Agents continue sensing about a cell until the expectation (Exp) of the top opinion in that cell is above a confidence threshold, where expectation is calculated as:

$$Exp = \frac{b+RA*u}{b+d+u} \tag{16}$$

where b and d are the amounts of evidence in favor of and against the belief, respectively, u is the amount of uncertain evidence, and RA is the relative atomicity of the belief (i.e., the uniform likelihood of the belief against all others). In this environment, each possible sensing outcome with the microscope yields evidence in favor of one mineral type and against the others. The uncertainty count starts with and remains a 1 (never incremented by sensing which always yields a type of mineral), representing a prior estimate that the likelihood of all types is equal and non-zero for any given mine. Finally, the relative atomicity of the three types is each 1/3 since there are three possible types of minerals in each mine.

4.1.2. Observer Effect MDP Instantiation

In the MineralMiner environment, each microscope is a stateful resource used during sensing whose behavior depends on its state (energy): *using a microscope reduces the available energy in its battery which leads to less accurate observations*. Thus, an agent must tradeoff the need for knowledge refinement necessary for accomplishing its current tasks against knowledge corruption due to Observer Effect-produced faulty observations. To handle this tradeoff, we model the process of selecting microscope-based sensing actions as an Observer Effect MDP, whose mapping is shown in Table 4.1. Please note that we represent the state of knowledge as the number of previous observations for a belief equal to $b + d$ in equation (16) since these combine with the constant prior $u = 1$ to be a weighting factor on the influence of the next update. Finally, we represent the value of knowledge refinement as the change in the agent's top expectation for the mineral type in each supply, calculated using equation (16).

Table 4.1 MineralMiner Observer Effect MDP

Observer Effect MDP	Mineral Miner Characteristics
Stateful Resource	Electronic Microscope
Sensing States $S = \{< R_s, K_s >\}$	$R_s =$ Microscope Energy, $K_s =$ # of Previous Observations
Activity Choices A	Advanced/Basic Mine Test, Wait
Transition Probabiliites $T(s, a)$	Change in energy from a test or self-recharge during wait, increased number of observations
Knowledge Refinement Reward $R(s, a)$	Increase or decrease in the believed possibility of the true mineral in the supply

To learn how to operate in this Observation Selection MDP, we consider three RL algorithms: 1) Q-Learning (Watkins, 1989), 2) RMax (Brafman and Tennenholtz, 2002), and 3) REINFORCE (Williams, 1992). These algorithms were not chosen as an exhaustive study of state-of-the-art RL algorithms, but rather due to the range of types of RL algorithms they represent, as well as their popularity and simplicity or their special properties.

As we have already previously described Q-Learning and RMax in Sections 3.3.1 and 3.3.2, we do not go into further detail here. REINFORCE (Williams, 1992), on the other hand, is a class of model-free RL algorithms which use neural networks to learn both an action selection controller and the reward function $R(s, a)$. Specifically, the agent learns 1) a stochastic neural network which determines the probability it should select each action depending on the state of the environment, and 2) a neural network which approximates the reward function, dependent on the current state and action taken. The latter is learned using traditional supervised learning methods, while the former is learned by reinforcing the weights of the network based on the reward received for taking an action. In our experiments, we use the following update function (Williams, 1992) to train the network using backpropogation (Rumelhart *et. al*, 1986):

$$\alpha\big(r(s, a) - R(s, a)\big)e \tag{17}$$

where α, $r(s, a)$, and $R(s, a)$ are as defined previously (c.f., Section 3.3.1) and e is the computed eligibility of weight updates (Williams, 1992). Of note, we reinforce using $r(s, a) - R(s, a)$ rather than just the recent $r(s, a)$ as our learned $R(s, a)$ provides a baseline for rewards which allows the algorithm to clamp down on variance during learning.

Comparing these three algorithms, Q-Learning and RMax are simple yet popular model-free and model-based RL algorithms, respectively. However, both of these use discrete states, so some information is lost discretizing the true sensing state. In MineralMiner, the state of the microscope resource (energy) is continuous, so we discretize the R_s values into discrete bins across their range [0, 1]. REINFORCE, on the other hand, allows for continuous states since it is neural network-based, allowing us to keep the continuous values for R_s during learning.

4.1.3. Experimental Setup

Objectives: Within the MineralMiner environment, we conduct experiments to evaluate the use of the Observer Effect MDP for controlling agent sensing with stateful resources. Specifically, we have two objectives to evaluate two hypotheses:

Objective MM1: Evaluate the Observer Effect Hypothesis in MineralMiner

We evaluate the OE Hypothesis (c.f., Section 2.3.3) by comparing the three previously mentioned RL algorithms for learning how to choose sensing activities according to the Observer Effect MDP against three baseline agents which follow sensing policies which do not consider resource state or the Observer Effect: 1) *Advanced* and 2) *Basic*, where the agent always chooses the advanced and basic mine test sensing actions, respectively, and 3) *Random*, where the agent randomly chooses one of the three sensing actions, including wait.

The metric we use to compare the various approaches is *sensing performance*: the average knowledge refinement per sensing activity. This metric was chosen because it is knowledge refinement which agents aim to optimize in the Observer Effect MDP. In MineralMiner, average knowledge refinement is defined as the change in the expectation of the agent's opinion of the correct mineral type in the mine tested (known by the agent for sure once it drills the mine).

Objective MM2: Evaluate the Performance Hypothesis in MineralMiner

> **Performance Hypothesis**: *Improving agent sensing performance will lead to improved agent task performance through proper decisions informed by knowledge refined through sensing.*

This Performance Hypothesis is important to motivate research on agent sensing: the primary value of sensing is in knowledge refinement for the sake of informing decisions to support task and goal accomplishment. While we assume this hypothesis holds in many MAS environments,

it could be the case in some that it does not due to several factors, including ease of task accomplishment, faulty (i.e., non-perfect) actuators, or a lack of need for sensing due to good *a priori* knowledge about the environment.

To evaluate the Performance Hypothesis we consider the correlation between sensing performance and *task performance*, measured as the total number of tasks completed by all agents. We choose this measurement because the primary goal of agents is to complete as many tasks as possible. If sensing performance does lead to improved task performance, we should expect to see a positive correlation between these metrics.

Environments: In order to understand the impact of the Observer Effect on agent behavior, we consider six environments with varying levels of OE in the microscopes, using different amounts of state-dependent accuracy error by varying the noise factor (NF):

$$error = \frac{(100 - energy)}{100} * NF \quad (18)$$

where a larger NF produces a larger error and Observer Effect. Thus, as the energy (i.e., state) of a microscope decreases due to usage, error increases and accuracy decreases.

Parameters: The values of the parameters to the MineralMiner simulation important to our experiments are presented in Table 4.2. To reduce the variance of the results, we run the experiments 30 times (each with a different random seed) and average the results.

4.2. UserRec: A User Preference Elicitation Simulation

4.2.1. Environment Description

The second simulated environment used in our experiments is *UserRec* which simulates a user preference elicitation problem. This simulation is based on an environment originally proposed by Doshi and Roy (2008) to evaluate a new PBVI-based algorithm for solving the preference elicitation POMDP (Boutilier, 2002; Doshi and Roy, 2008). We begin this section by

Table 4.2 MineralMiner Experiment Parameters

Parameter	Value
Grid Size	20 x 20
# Supplies	20/type
# Agents	30
# Tasks	50
# Agents/Task	5
Microscope Recharge Rate	10%/time unit
Max Advanced Test Energy	50%
Max Basic Test Energy	25%
Advanced Test Accuracy	0.8
Basic Test Accuracy	0.4
Belief Confidence Threshold	0.65
Learning Rate α	0.3
# Discrete Sensing States	400
NF	0, 0.1, 0.2, 0.3, 0.4, 0.5

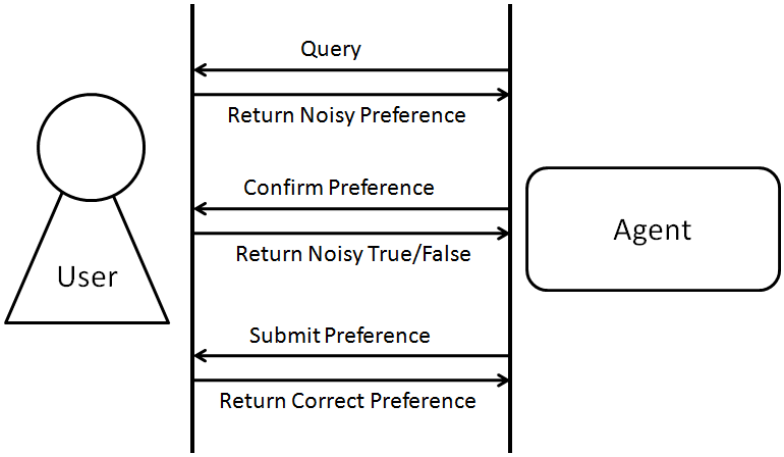

Figure 4.1 Human-Agent Interaction in UserRec

describing their environment, followed by our extension to more realistically model the real-world scenario by including effects of user frustration.

Doshi and Roy's World. In the original environment proposed by Doshi and Roy (2008), an intelligent agent is tasked with supporting a human user. See Figure 4.1 for the interactions between an agent and its human user. To provide such support, an intelligent agent interacting with a human user must elicit the user's preference over a set of items (e.g., goods, scenarios, goals) in order to provide support to the user (e.g., customized user recommendations

(Adomavicius and Tuzhulin, 2005)) over a sequence of episodes. To do so, the agent can perform two sensory actions:

1) *query*, which asks the user to state their current preference, and
2) *confirm*, whereby the agent asks the user if its belief about the user's top preference is correct.

The user then responds to the agent's query or confirm actions based on its actual preference, providing observations used to revise the agent's beliefs (i.e., knowledge) about current user preference. Once the agent is confident that it understands the user's preference, it can perform a third *submit* action which performs the intelligent support desired by acting on the user's preference. Each series of interactions with the user to elicit her preference represents an *episode* which ends with the submit action. After a submit, the user's preference is reset and a new episode begins.

To accurately model the real world scenario, the user's responses are noisy in that they stochastically provide incorrect responses (according to a fixed probability for each of the two sensory actions mentioned above, as shown in Table 4.3), inducing uncertainty in the agent's beliefs. Because of this noise, an agent may perform a series of *query* actions before committing to a *confirm* action to increase its confidence in its beliefs—in other words, to reduce uncertainty in its beliefs. Furthermore, there exists a slight probability (also shown in Table 4.3) that the user will switch its preference during an episode of interactions with the agent. Thus, it is possible that while an agent is trying to model its user's preference, the agent receives a true response from the user but later on this response is no longer accurate, making it difficult for the agent to reconcile all responses in its beliefs about user preference.

Table 4.3 Example Environment Parameters (Doshi and Roy, 2008)

Environment Parameter	Value
Correct Query Response Likelihood	50%
Correct Confirm Response Likelihood	80%
User Preference Change Likelihood	1%
Number of Possible User Preferences	10

Table 4.4 Example Task-Level Reward Structure for Agent Actions (Doshi and Roy, 2008)

Action	Reward
Query Preference	-2
Confirm Correct Preference	-1
Confirm Incorrect Preference	-5
Submit Correct Preference	100
Submit Incorrect Preference	-200

Furthermore, sensory actions taken by an agent incur a fixed cost (i.e., negative reward). Thus, an agent tries to avoid performing sensing actions unnecessarily if it believes that it has gathered enough information about the user's preference to successfully submit the user's preference. To help an agent decide whether to perform an action and which actions to take, an example reward structure was proposed in (Doshi and Roy 2008), and reproduced here in Table 4.4. For example, in the given reward structure, an agent is penalized more when its confirm and submit actions are incorrect (i.e., the wrong preference is identified), and queries are more costly than a correct confirm as it takes more effort from a user to respond over a set of preferences than to agree or disagree with the agent's top belief.

In the design of agent behavior to solve the preference elicitation problem in this environment, Doshi and Roy (2008) consider a POMDP model of the environment called the Preference Elicitation PODMP (Boutilier, 2002), described in Table 4.5. Specifically, an agent is given a fully parameterized PODMP model of the environment (created by an assumed domain expert) from which it builds a policy (using variants of the PBVI algorithm) to choose which actions to perform given its current belief state. Here, the belief state represents the agent's knowledge about user preferences as it measures the likelihood ascribed by the agent to the fact that each possible preference is the user's actual preference.

Table 4.5 Preference Elicitation POMDP Model

POMDP Model	Preference Elicitation Problem
States	User preferences
Actions	Query, Confirm, Submit
Observations	User preference or true/false
State transitions probabilities	Small chance of preference change during an episode
Observation probabilities	Likelihood of correct/incorrect responses to sensing actions
Reward	Reward structure (e.g, Table 4.4)

Our Extension: User Frustration. To more accurately model the real-world dynamics within the environment, we have added *user frustration* which persists across elicitation episodes and is increased by interruptions by the agent (i.e., the sensing actions) (Adamcyzk and Bailey, 2004; Mark et. al, 2008). Frustration is also increased when the agent improperly acts on its beliefs about user preference (i.e., submits a wrong preference), as bad actions decrease user trust in the system and desire to use the system (Klein *et. al*, 2002). In turn, frustration disrupts her cognitive state (Klein *et. al*, 2002), and she takes less time to respond in order to quickly return to what she was already doing before being interrupted due to increased time pressure from lost time due to interruptions (Mark et. al, 2008). On the other hand, frustration and its side effects are decreased when the agent provides proper intelligent support from correct beliefs about user preference, reducing the workload of the user.

After searching through the human-user interaction (HCI) literature, including research from the intelligent user interface (IUI) community, we could not find any quantifiable mathematical models for computer user frustration to include in our implementation. Thus, we propose the following as a first step approximation for user frustration to create a starting point for our research.

In order to keep things simple while still including user frustration within our simulated application, we model user frustration as a *cumulative* effect, where agent actions either increase or decrease frustration within the user. We represent a user's frustration level as an

amount within [0, 100] symbolizing a percentage ranging from completely unfrustrated to completely frustrated, respectively. Specifically, a user's frustration level is additive with the individual consequences of each agent action. We provide an example of these values in Table 4.6. Please note that we have based these values on the reward values given in Table 4.3, but modified by:

1) changing positive values to negative and vice-versa due to frustration being more similar to a cost instead of a reward,

2) reducing the magnitude of values so user frustration doesn't increase too quickly (since it ranges from 0 to 100) and so that correct/incorrect submissions do not dominate sensing actions but are still relatively more impactful, and

3) changing ratios so that the penalty/benefit for incorrect/correct preference submissions offset rather than incorrect submissions dominating.

Table 4.6 Example Frustration Structure for Agent Actions

Action	Frustration Increase
Query Preference	1
Confirm Correct Preference	0.5
Confirm Incorrect Preference	2.5
Submit Correct Preference	-10
Submit Incorrect Preference	10

To make our extension even more interesting, based on her current frustration level, the user both *changes her response time* (i.e., responds faster to return to her work), as well as *responds less accurately* (due to disrupted cognitive state). Here, we model both the change in response delay and accuracy reduction as being proportional to current user frustration by making these effects linear with frustration[2]. Specifically, user delay defaults to a *maximum*

[2] We acknowledge that these effects might not be linear in practice, but given a lack of available models, we deemed this an appropriate starting point. For future work, we intend to investigate other models (e.g., linear regressions, exponential/polynomial models) as well as any developed through HCI research.

value (MAX_DELAY) when the user is not frustrated at all and is calculated as this maximum minus the product of frustration ($Frust$) with a *frustration delay factor (FDF):*

$$delay = MAX\_DELAY - FDF * Frust \quad (19)$$

(ceilinged to produce an integer number of simulation ticks representing time), while the decrease in user response accuracy ($noise$) is calculated as the product of frustration with a *frustration noise factor* (*FNF):*

$$noise = FNF * Frust \quad (20)$$

Finally, because a human user cannot be expected to continue to work while completely frustrated, we also end our simulations early (i.e., before all episodes are finished) if the user is at maximum frustration at the end of a certain number of episodes in a row. We call this number the user's *boiling point count* as it represents the amount of time they are willing to have their frustration above a maximum boiling point (100% frustration).

4.2.2. Observer Effect POMDP Instantiation

In the UserRec environment, each user is a stateful resource used during sensing whose behavior depends on her state (frustration): *interrupting a user to perform sensing increases frustration which leads to less accurate observations*. Thus, as in the MineralMiner environment, an agent must tradeoff the need for knowledge refinement necessary for accomplishing its current tasks against knowledge corruption due to Observer Effect-produced faulty observations. To handle this tradeoff, we again model the process of selecting user-interrupting sensing actions as an Observer Effect MDP. However, since user frustration is a hidden environment property, an agent must instead use an Observer Effect POMDP instead of a MDP. Here, the agent relies on observable response delays to estimate the hidden, cumulative frustration of the user. The mapping of this Observer Effect POMDP is shown in

Table 4.7 and we compare our Observer Effect POMDP with Doshi and Roy's (2008) Preference Elicitation POMDP in Table 4.8.

Table 4.7 UserRec Observer Effect POMDP

Observer Effect POMDP	UserRec Characteristics
Stateful Resource	Human User
Sensing States $S = \{< R_S, K_S >\}$	R_S = User Frustration, K_S = # of Previous Observations
Activity Choices A	Query, Confirm
Observations O	User Response Delay
Transitions Probabilities $T(s, a, s')$	Change in frustration from a sensing action, Increased number of observations
Observation Probabilities $\Omega(o, s', a)$	Likelihood of possible response delay values dependent on user frustration
Knowledge Refinement Reward $R(s, a)$	Increase or decrease in the belief state probability assigned to the true user preference

Table 4.8 UserRec Observer Effect POMDP vs. Preference Elicitation POMDP

POMDP Model	Observer Effect POMDP	Preference Elicitation POMDP
States S	Sensing State	User Preference
Choices A	Query, Confirm	Query, Confirm, Submit
Observations O	User Response Delay	User Responses
Transitions Probabilities $T(s, a, s')$	Change in Sensing State	Change in User Preference
Observation Probabilities $\Omega(o, s', a)$	Likelihood of Delay Values	Likelihood of Correct/Incorrect Responses
Reward $R(s, a)$	Value of Knowledge Refinement	Task Reward

An agent using the Observer Effect PODMP for sensing activity selection in the UserRec simulation operates as follows. First, to be fair in our experiments (described in the following section), the agent uses the same knowledge representation and revision procedure as the Preference Elicitation POMDP[3]. Specifically, the agent uses the user goal change and response accuracy probabilities to calculate a probability distribution over user preferences representing the likelihood that the agent ascribes to each preference being the correct one. This

[3] Please note that we did not have to use the same knowledge representation and revision procedures in the Observer Effect POMDP. Instead, we could have used any framework, including the possibility logic framework (Josang, 2001) used in the MineralMiner experiments. However, if we had not used the same framework, it would be difficult to later compare the results between using the Preference Elicitation POMDP and the Observer Effect POMDP.

distribution is revised using equation (11) from Section 3.2.1. However, it is important to note while this knowledge does represent a belief state in the Preference Elicitation POMDP, it is *not* the belief state of the Observer Effect PODMP since the latter's states represent user frustration, not preference. Finally, using this form of knowledge (i.e., preference probabilities), we again represent knowledge state in the Observer Effect POMDP as the number of previous observations (c.f., Section 4.1.2) because an increased number of observations drives the belief differences between states supported and unsupported by observations further apart, and thus effects the magnitude of possible change in belief after the next observation.

In order to make decisions about what sensing activities to perform to gather information used to revise the agent's knowledge about user preferences, the Preference Elicitation POMDP and the Observer Effect POMDP differ in their behavior. In the former, an agent builds a controller based on maximizing expected task rewards (Table 4.4) for selecting sensing activities. In the latter, the agent builds a controller based on maximizing the expected value of knowledge refinement provided by sensing. Thus, the Preference Elicitation POMDP focuses on *task rewards* while the Observer Effect MDP focuses on *knowledge refinement*. This is an important distinction as task rewards are considered to be constant and independent of user frustration, whereas considering the frustration-dependent value of knowledge refinement captures the possible distortion of sensing outcomes caused by the Observer Effect.

Finally, we want to point out that both approaches do use the task reward structure from the Preference Elicitation POMDP to decide whether or not to sense. At an abstract level, both approaches choose the submission action if the expected task reward of acting is greater than the expected task reward of sensing based on the probabilities assigned to each possible user preference in the agent's knowledge. That is, an agent chooses to submit once the highest probability assigned to a user preference is large enough (determined by the relative values in

the task reward structure, e.g., 66% for the values in Table 4.4). If the agent does not choose to submit, then the two approaches differ in how they select a sensing activity, as described in the previous paragraph. Furthermore, an agent using the Observer Effect POMDP also chooses a submission action if it believes that the expected value of knowledge refinement will be negative, following the optional constraint to the OETP described in Section 2.4 in order to avoid further frustrating the user, risking the user no longer being willing to use the system and/or increased corruption to agent knowledge due to the Observer Effect. We present a diagram of the reasoning processes of both approaches in Figure 4.2 for comparison.

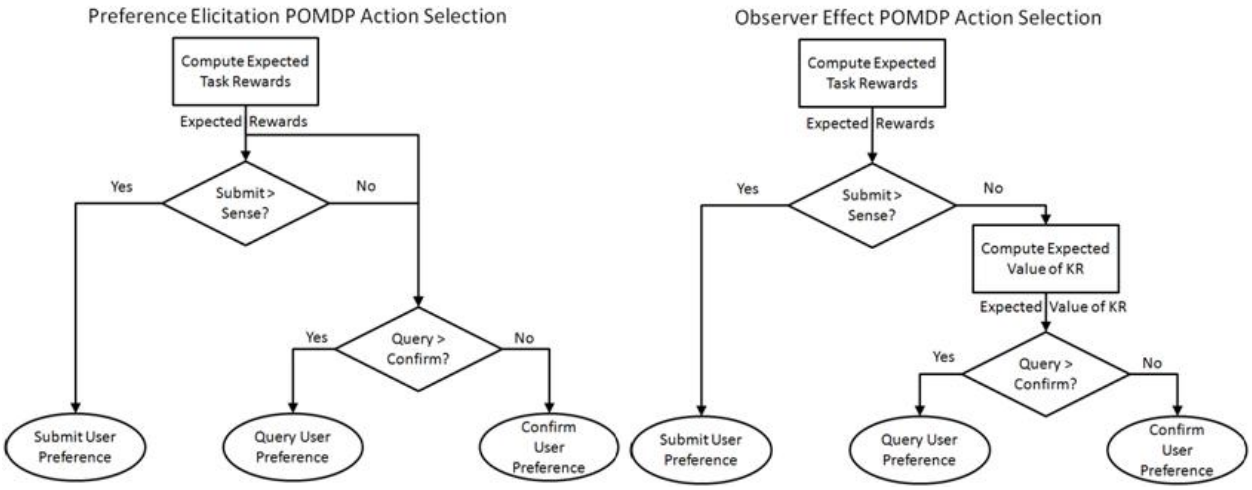

Figure 4.2 Action Selection in the Observer Effect and Preference Elicitation POMDPs

To build a controller for the Preference Elicitation POMDP to follow the selection process shown in Figure 4.2, the agent simply solves the POMDP model provided to the agent (e.g., by using PBVI (Pineau *et. al*, 2003)). To build a controller for the Observer Effect POMDP, on the other hand, we consider two PORL algorithms: 1) Bayes-Adaptive POMDPs (Ross *et. al*, 2007), and 2) Recurrent Policy Gradients (Wierstra *et. al*, 2007). For the latter, we use PORL because we do not assume that any knowledge about stateful user behavior is provided to the agent *a priori*.

First, Bayes-adaptive POMDPs (BA-POMDPs) (Ross *et. al*, 2007) is a state-of-the-art model-based PORL approach. Like the aforementioned fully-observable RMax algorithm (Brafman and Tennenholtz, 2002) (c.f., Sections 3.3.1 and 3.3.2), this model relies on experience tracking (in Dirichlet distributions) to model the conditional transition and observation probability functions. These distributions are represented by two sets of vectors 1) $\phi = \{\phi_{ss'}\}$ the conditional transition distributions where $\phi_{ss'}$ records the number of times the environment transitioned from state s to s' with entries for each action $a \in A$, and 2) $\psi = \{\psi_{s'o}\}$ the observation probability distributions where $\psi_{s'o}$ records the number of times the environment produced observation o after transitioning to state s' with entries for each action $a \in A$. However, because the agent does not know exactly what the state of the environment is at any point in time, the entries in these vectors cannot be counted deterministically. Instead, the agent considers the possibility that a number of possible ϕ and ψ are correct. To handle this difficulty, the states in the BA-POMDP are augmented with the Dirichlet distributions themselves, yielding a state space $S' = S \times \Phi \times \Psi$, where Φ represents the power set of ϕ and Ψ represents the power set of ψ. Thus, the new (larger) POMDP contains all possible state/learned model pairs as its states, allowing the agent to consider all possible models and all possible states with a single belief state. The agent learns which model is best by favoring more likely models and states with higher values in the belief state after each belief update. To build a policy controlling agent actions using BA-POMDPs, an agent solves each model for each complex state and chooses actions accordingly. Specifically, we solve each model online using 1-step decision trees (c.f., Section 3.2.1) and choose the action which has the highest expected utility over all models, weighted by the probability of the complex BA-POMDP state to which that model corresponds. To learn the reward function for use with BA-POMDPs, we take a similar experience counting approach as that taken with the RMax RL algorithm in the

MineralMiner environment (c.f., Section 3.3.2). Specifically, we discretize the value of knowledge refinement ranging from [-1.0, 1.0] into discrete bins and count the number of times each discrete bin is experienced for each sensing activity and environment state. A minor detail is that since the state of the environment is hidden and estimated by the POMDP belief state, we actually increment fractional counts based on the amount of belief we have in each state, rather than a full 1.0 count for a known state in fully-observable environments.

Second, we also employ a state-of-the-art model-free PORL approach called the Recurrent Policy Gradient (RPG) algorithm (Wierstra *et. al*, 2007). This algorithm is an extension for partially-observable environments of the REINFORCE (Williams, 1992) algorithm described previously in Section 4.1.2. Specifically, for its neural networks, it trains long short-term memory (LSTM) recurrent neural networks (RNNs). Here, a RNN (e.g., Jaeger, 2002) is a special type of neural network which allows for bidirectional connections between individual neurons and layers in the network, as opposed to the more commonly used feed-forward networks (which we use in REINFORCE) which only have connections directed from inputs towards outputs through the various layers. While RNNs can be used to learn over time sequences due to their ability to retain information within the hidden layer, possible due to the bidirectional communications between individual neurons, the amount of time information can be retained in the network is often limited to only a few time steps before it is lost in new information. To overcome this problem, LSTM RNNs (Hochreiter and Schmidhuber, 1997; Gers *et. al*, 2000) include memory cells within the hidden layer of the network, allowing arbitrarily long retention of information. This is very important as it allows the LSTM RNN to find relationships and patterns amongst widely separated inputs in the history. Within the context of PORL, this allows the learner to more accurately predict appropriate actions because it implicitly discovers and considers patterns of environment behavior based on observing the results of previous actions.

This occurs within the network's memory and does not require building an explicit parameterized environment model as done by model-based PORL. Training the action controller and reward function neural networks in RPG follows the same update rule as REINFORCE (Eq. 17). However, since we are now using LSTM RNNs, we use a variant of backpropogation-through-time (Werbos, 1990) for LSTMs (Hochreiter and Schmidhuber, 1997; Gers *et. al*, 2000) for network training, instead of standard backpropogation as we do for REINFORCE. Further, since the state of the environment is now hidden, these networks take the observable delay in user response as an input, in place of resource state.

4.2.3. Experimental Setup

Objectives: Within the UserRec environment, we conduct experiments similar to those for MineralMiner (c.f., Section 4.1.3) to evaluate the use of the Observer Effect POMDP for controlling agent sensing with stateful resources in partially observable environments. Again, we have two objectives to evaluate two hypotheses:

Objective UR1: Evaluate the Observer Effect Hypothesis in UserRec

We evaluate the Observer Effect Hypothesis (c.f., Section 2.3.3) by comparing the two previously mentioned PORL algorithms (c.f., Section 4.2.2) for learning how to choose sensing activities according to the Observer Effect POMDP against two baseline agents which follow sensing policies that do not consider resource state or the Observer Effect: 1) *PBVI*, where the agent follows the general approach of Doshi and Roy (2008) by solving the Preference Elicitation POMDP using PBVI (Pineau *et. al*, 2003), and 2) *Random*, where the agent randomly chooses one of the two sensing actions when it needs information.

As in the Mineral Miner experiments (Objective MM1), the metric we use to compare the various approaches is *sensing performance*: the average knowledge refinement per sensing activity. In UserRec, average knowledge refinement is defined as the change in the agent's

belief state about the correct user preference (determined at the end of each episode after submitting to the user).

Objective UR2: Evaluate the Performance Hypothesis in UserRec

Again, we evaluate the Performance Hypothesis to determine whether or not changes in sensing performance (evaluated in Objective UR1) equate to improved task performance by the agent. To evaluate the Performance Hypothesis in UserRec, we again consider the correlation between sensing performance and task performance, where the latter is measured as the number of correct preference submissions by the agent, totaled across all episodes. We choose this measurement because the primary goal of the agent is to correctly determine the user's preference and provide intelligent support. However, we also consider a second task performance metric: average task utility, measured as the average amount of task reward received for each episode of interaction with the user. This measurement is appropriate because the secondary goal of the agent is to maximize its utility (i.e., minimize sensing costs) while interacting with the user. Overall, if sensing performance does lead to improved task performance (confirming the Performance Hypothesis), we should expect to see a positive correlation between the sensing performance metric (i.e., average knowledge refinement) and these two task performance metrics (i.e., correct submissions and average reward).

Environments: In order to understand the impact of the Observer Effect on agent behavior in our simulated preference elicitation problem, we consider four environments in UserRec varying the rate of frustration change in the user while keeping the frustration delay and noise factors (i.e., FDF in (Eq. 19) and FNF in (Eq. 20)) constant. In other words, here we vary the state transitions while keeping the magnitude of the Observer Effect constant. This differs from the strategy taken in the MineralMiner experiments (c.f., Section 4.1.3) where we do the opposite. We take this approach for two reasons: 1) given a lack of quantitative models of user frustration,

we hypothesize that actions frustrate different users differently but are unsure whether or not a similar level of frustration effects different users differently, and 2) this approach allows us to study two different ways of varying the Observer Effect, when considered in conjunction with the MineralMiner experiments. Specifically, we consider four different sets of frustration increases caused by agent activities which interrupt the human user. We call these four sets 1) *Zero*, representing a user who does not become frustrated (i.e., an environment with no Observer Effect); 2) *Patient*, representing a user who does not easily become frustrated; 3) *Angry*, representing a user who is much quicker to frustration than patient (by a factor of 5), and 4) *Task-oriented*, representing a user unbothered by sensing activities but cares about whether or not the agent is supporting her tasks. The specific values for the sets are shown in Table 4.9.

Table 4.9 UserRec Frustration User Types

Action	Zero Frustration User	Patient User	Angry User	Task-oriented User
Query Preference	0	0.2	1	0.2
Confirm Correct Preference	0	0.1	0.5	0.1
Confirm Incorrect Preference	0	0.5	2.5	0.5
Submit Correct Preference	0	-1	-5	-5
Submit Incorrect Preference	0	1	5	5

Parameters: The values of the parameters to the UserRec simulation important to our experiments are presented in Table 4.10, as well as reused from Tables 4.3-4.4 (c.f., Section 4.2.1) and 11. Please note that we limit the number of interactions per episode to 10 in order to avoid over-sensing and frustrating the user too quickly while learning how to control sensing. As in the MineralMiner experiments, to reduce the variance of the results, we run the experiments 30 times, each with a different random seed, and average the results.

Table 4.10 UserRec Experiment Parameters

Parameter	Value
# Episodes	200
Max # Interactions / Episode	10
Boiling Point Count	3
MAX_DELAY	10.0
FDF	0.1
FNF	0.005
Learning Rate α	0.3
# Discrete Sensing States	50

4.3. Simulation Environment Comparison: MineralMiner vs. UserRec

We conclude this chapter by comparing the properties of the two simulation environments considered in this thesis: MineralMiner and UserRec. A summary of this comparison is provided in Table 4.11.

Table 4.11 MineralMiner vs. UserRec Comparison

Property	MineralMiner	UserRec
Resource State Observability	Fully observable	Partially Observable
MAS	Multiple agents	Single Agent
Stateful Resource	Microscope (energy)	Human user (frustration)
Causes of State Change	Microscope Usage, Recharge over time	Interrupting the user, Submitting user preference
Knowledge Representation	Possibility Logic (Josang, 2001)	Preference Elicitation POMDP belief states (Doshi and Roy, 2008)
Value of Knowledge Refinement	Change in possibility of true state	Change in belief probability of true state
Agent Reasoning	Plan-oriented	Preference Elicitation PODMP policy
RL Algorithms	Q-Learning (Watkins, 1989) RMax (Brafman and Tennenholtz, 2002) REINFORCE (Williams, 1992)	BAPOMDP (Ross *et. al*, 2007) RPG (Wierstra *et. al*, 2007)

Concerning the Observer Effect MDP, the most notable difference between the two applications is the observability of the resource state. In the MineralMiner environment, the agent can always read the current energy level of the microscope (the resource state), while in the UserRec environment, the user's frustration level is hidden from the agent and must be

estimated based on the delay in user responses. Thus, MineralMiner represents a fully observable environment, while UserRec is a partially observable environment. Considering both types allows us to test the Observer Effect MDP in a range of environments agents are likely to experience in real-world applications. Given that these environments differ in their observability, they also use different types of reinforcement learning algorithms: traditional, fully observable RL in MineralMiner and more recent PORL in UserRec. Finally, the means through which the resource states change also differ between the two environments. Specifically, while resource state is changed by sensing activities in both, it is also changed over time through energy recharging in MineralMiner, while task-oriented outcomes (i.e., intelligent support through preference submissions) can change resource state in UserRec.

Considering the internal structure of the agent, we note that the reasoning processes and subsequent knowledge representations also differ between the two environments. In MineralMiner, the agent follows a more strict plan-oriented reasoning process, performing actions while following a set of priorities (c.f., Section 4.1.1), while in UserRec, the agent represents the preference elicitation problem as a Preference Elicitation POMDP (Boutilier, 2002; Doshi and Roy, 2008) whose reward structure assists the agent in determining whether to sense or submit the user's preference. Furthermore, the agent in MineralMiner uses evidence counting of observations in opinions following a possibility logic framework (Josang, 2001) to track the mineral type in each mine, while the agent in UserRec uses the given Preference Elicitation POMDP's parameters to refine a belief state representing probabilities over possible user goals. In both environments, the amount of change in the correct opinion/belief after a sensing activity is the measurement used for the value of knowledge refinement. This value is calculated after the correct belief is known for sure: once the agent drills a mine in MineralMiner and after submitting the user's preference in UserRec.

Finally, the two environments differ in the number of agents involved. MineralMiner represents a teleological, multiagent environment where different agents can all rent the same microscope at different times and thus influence its state when the microscope is first rented by the next agent (unless it has fully recharged before rented again). UserRec, on the other hand, represents a single agent environment where only one agent uses and changes the state of the human user resource.

Chapter 5 The Results – MineralMiner and UserRec Experiments

In this chapter, we evaluate the results of the experimental setup described in Chapter 4 of this thesis, used to validate our Observer Effect MDP solution from Chapter 3 to solve the Observer Effect Tradeoff Problem from Chapter 2. We begin by evaluating the results of the *fully observable* MineralMiner experiments, followed by the results of the *partially observable* UserRec experiments. Finally, we conclude the chapter with a discussion of the results across all experiments, including common trends and important discoveries.

5.1. MineralMiner Results

In this section, we present the results of the MineralMiner experiments described previously in Section 4.1. First, we briefly validate the existence of the Observer Effect in this environment, taking the form of reduced microscope accuracy. Next, we follow our two objectives to evaluate two hypotheses: the Observer Effect Hypothesis (c.f., Section 2.3.3) and the Performance Hypothesis (c.f., Section 4.1.3). Specifically, we evaluate the former by considering the sensing performance of the various sensing activity selection approaches. Then, we evaluate the latter by considering the task performance of the approaches, as well as the correlation between sensing and task performance. Finally, we provide a brief discussion summarizing the lessons learned from the MineralMiner experiments. Please note that for both sensing and task performance, we evaluate both across the overall results, as well as the results over time since we are considering learning algorithms which should improve over time.

5.1.1. Observer Effect Validation

Before we start the results analysis, we briefly validate the state-dependent behavior of the microscope resource in the simulation results, exemplifying the Observer Effect. This is an

important step because 1) if the Observer Effect does not exist, then there is no reason to consider it within our solution approaches, and 2) if it does not exist, then we know for sure that the ability of the RL approaches to consider the Observer Effect is not the cause of their difference in performance (presented in the following sections). Recall that in the MineralMiner simulation (c.f., Section 4.1.2, 4.1.3), the Observer Effect takes the form of reduced microscope test accuracy due to noise caused by a less-than-perfect energy level in the microscope during testing. The amount of this noise is governed by a noise factor NF which we vary in our experiments to represent different levels of Observer Effect.

To validate the existence of the Observer Effect in our MineralMiner experiments, we present the average microscope accuracy over all environments and approaches in Figure 5.1. We observe that for all of the approaches, as the noise factor NF increases, the average accuracy of microscope decreases. Thus, our experiments do contain the Observer Effect as desired.

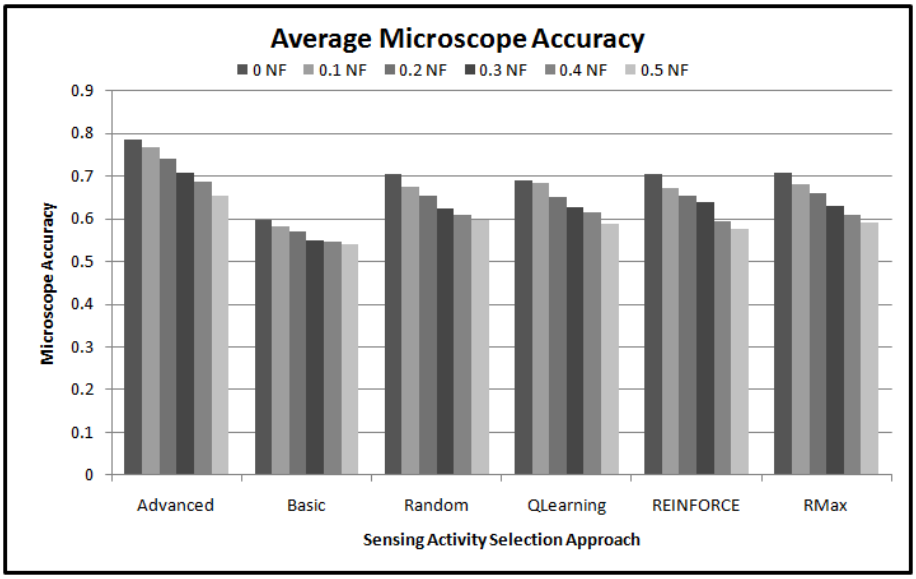

Figure 5.1 Average Microscope Accuracy for MineralMiner

5.1.2. Objective MM1

Now that we have demonstrated that the Observer Effect is present in the experiments analyzed here, we begin the MineralMiner results analysis by considering the sensing performance of the various sensing activity selection approaches, represented by the average value of knowledge refinement per sensing activity. We present these results across all approaches and environments in Figure 5.2. We note that these results are statistically significant ($\rho < 0.005$) with the two-way ANOVA results shown in Table 5.1.

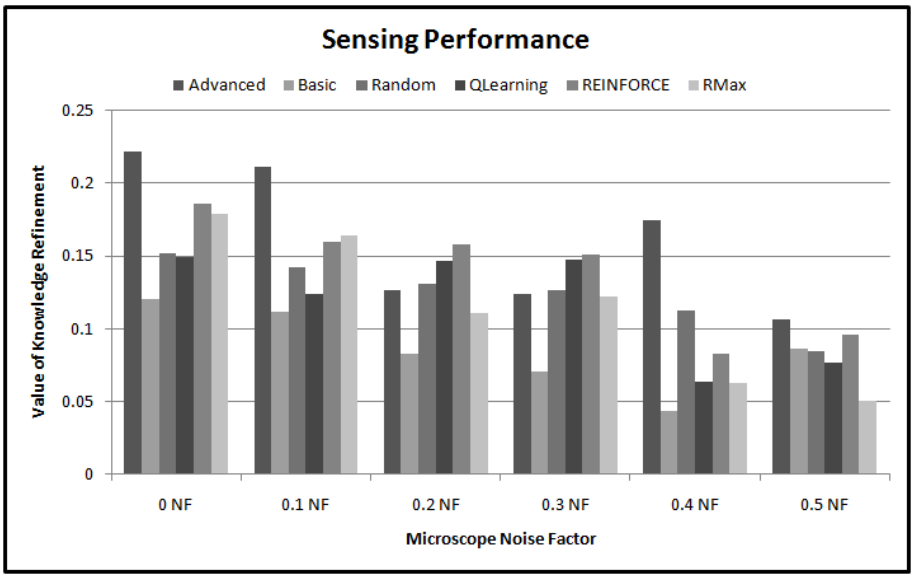

Figure 5.2 Sensing Performance in MineralMiner

Table 5.1 Two-way ANOVA Results for Sensing Performance in MineralMiner

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Environment	0.356758	5	0.071352	63.73987	4.14E-58	2.222674
Approach	0.51282	5	0.102564	91.6225	5.09E-80	2.222674
Interaction	0.032532	25	0.001301	1.162461	0.264806	1.516665
Within	1.168674	1044	0.001119			
Total	2.070784	1079				

From the sensing performance data shown in Figure 5.2, we observe that agents using RL (Q-Learning, REINFORCE, and RMax) to solve the Observer Effect MDP perform better knowledge refinement than agents which do not consider the Observer Effect (Advanced, Basic,

and Random) for environments with NF values of 0.2 and 0.3 but do not outdo the simpler approaches above and below these values. We believe this is due to the impact of the Observer Effect at each of these levels:

- For smaller NF values (i.e., 0 and 0.1), we contend that there is not enough Observer Effect present to effectively distort sensing to make a difference in knowledge refinement. This is evidenced by the very similar results of the average value of knowledge refinement for the three simple approaches between the 0 and 0.1 NF levels. Thus, small amounts of Observer Effect (0.1 NF) are insignificantly different than no Observer Effect (0 NF) and considering this detrimental effect is not necessary here.
- As the Observer Effect increases (i.e., 0.2 and 0.3 NF), the sensing performance of approaches which do not consider this effect decreases. Thus, increased Observer Effect makes it more difficult to refine knowledge. Yet, the RL-based approaches are able to counter this problem by learning the relationship between sensing state, sensing actions, and the value of knowledge refinement in order to achieve very similar sensing performance as when the Observer Effect is negligible.
- For the largest NF values (i.e., 0.4, 0.5), the Observer Effect appears to increase to a point where considering its effect on knowledge refinement is not enough to counter this challenge as the RL algorithms all perform worse than in smaller NF amounts, following a similar trend as the non-RL approaches. We do note that the Advanced approach appears to perform very well for the 0.4 NF level of Observer Effect, going against its trend of doing worse as OE increases. However, we believe that this is an anomaly as it decreases again for 0.5 NF, demonstrating a continued trend of decreasing sensing performance.

Based on these results, we conclude that considering resource state and the Observer Effect can improve sensing performance and *confirm the Observer Effect Hypothesis*, but note that low levels of Observer Effect do not necessitate this consideration and high levels make it difficult to do so. In the future, we intend to investigate what happens around the point when this difficulty appears in order to find a way of predicting its occurrence, as well as find ways to adapt our methodology to mitigate more Observer Effect.

Next, we consider the sensing performance of the various approaches over time. Specifically, we break each experiment into time periods (called Task Groups) based on groupings of the successive tasks performed by the agents, where each period breakpoint is defined by the deadline of the fifth[4] task in a Task Group. This analysis of the time series of the results is an important consideration because we employ reinforcement learning algorithms to solve the Observer Effect MDP: in general in artificial intelligence research, we desire that a learning approach improve its performance over time.

We present the sensing performance over time results from the 0.3 NF environment in Figure 5.3 as an example (c.f., Appendix A.1 for the other environments). From this figure, we make two key observations with respect to the Observer Effect Hypothesis which holds for the other environments as well:

- None of the approaches consistently outperforms the others across all task groups. Thus, while we can draw statistically significant conclusions from the overall results, these results do not indicate that at any given point in time, one approach is guaranteed to do better than the others.
- Further, none of the approaches consistently increase their performance over time, including the RL approaches. However, except for the highest Observer Effect

[4] We chose a task group size of five in order to split the 50 task groups into a nice number of 10 Task Groups. In general, any group size could have been employed.

environments (0.4 and 0.5 NF) where the Observer Effect becomes too difficult to handle, the RL approaches do increase their performance over time when compared to the Random approach. We give examples of this behavior and its exception in Figures 5.4 and 5.5 for the 0.3 NF and 0.5 NF environments, respectively. For this observation, we compare the RL approaches against Random because without learning, the former degrade into the latter approach as they start with equal rewards for all actions in all states (or near equal random values in the case of REINFORCE's neural networks). Thus, comparing the performance of the RL approaches versus Random shows the contribution of reinforcement learning to the agent's sensing performance.

- Similarly, we note that for the 0.2 NF environment shown in Figure 5.6, the RL approaches generally get better over time against Random until the very last task group. We believe this is an anomaly as the Random approach performs very little sensing in this last task group resulting in less knowledge refinement to compute an average value for, an interesting phenomenon which does not occur anywhere else in our simulations.

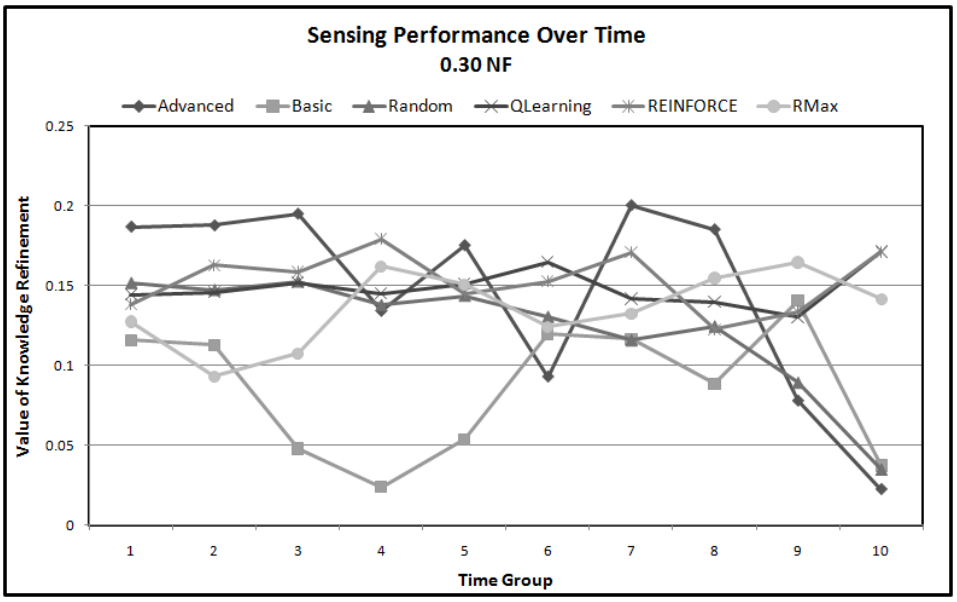

Figure 5.3 Sensing Performance over Time in MineralMiner (0.3 NF)

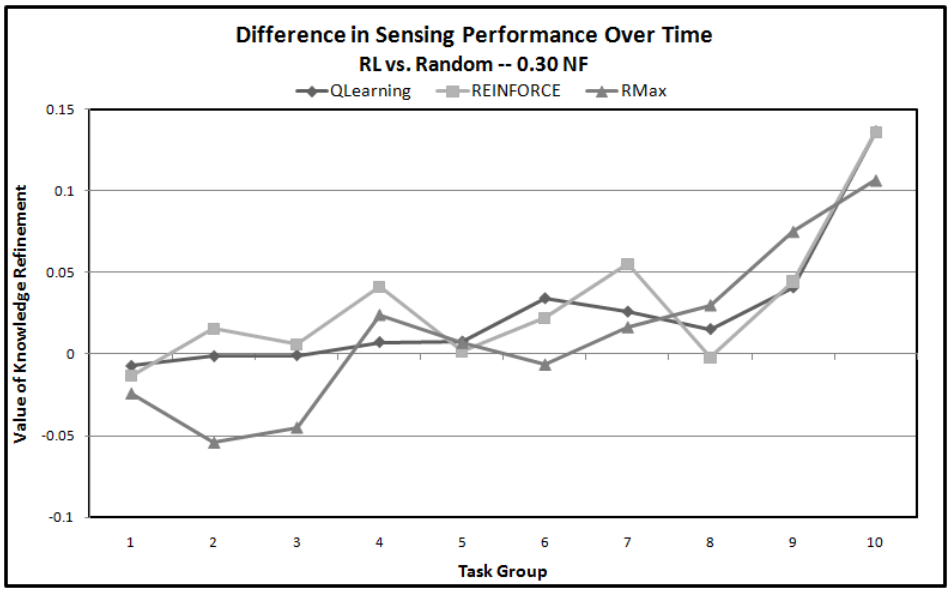

Figure 5.4 Sensing Performance of RL vs. Random over Time in MineralMiner (0.3 NF)

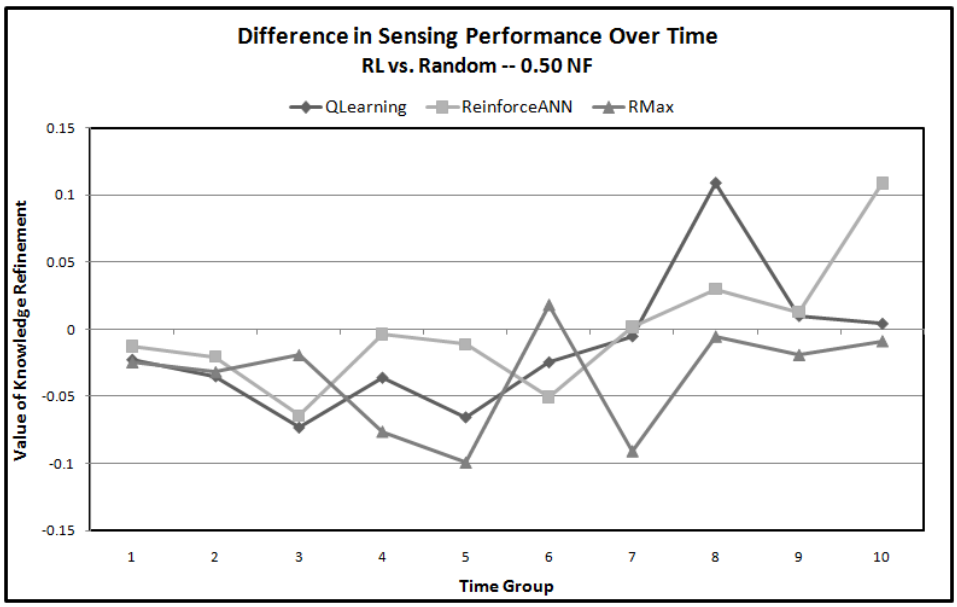

Figure 5.5 Sensing Performance of RL vs. Random over Time in MineralMiner (0.5 NF)

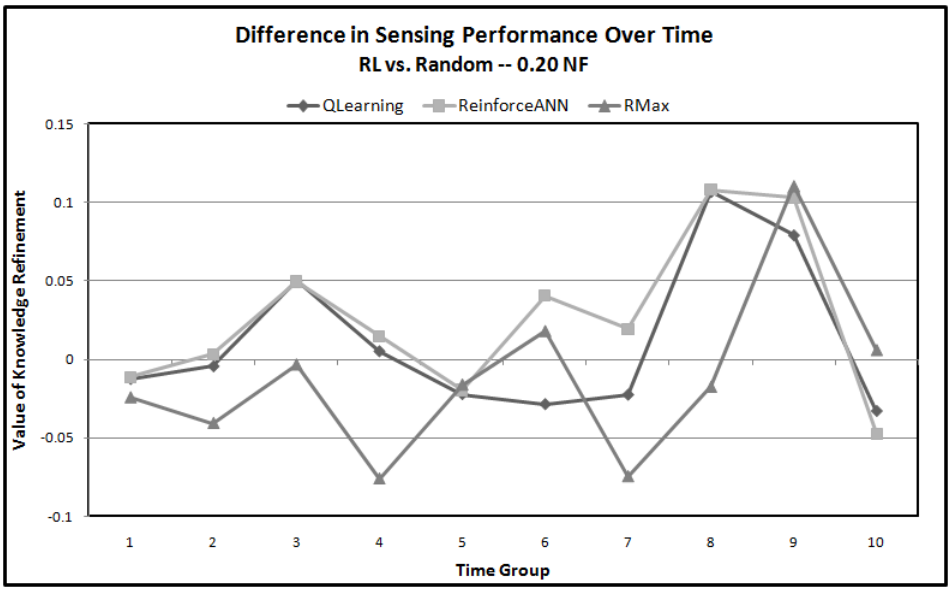

Figure 5.6 Sensing Performance of RL vs. Random over Time in MineralMiner (0.2 NF)

5.1.3. Objective MM2

Next, we consider the task performance for the various approaches to sensing activity selection, represented as the number of tasks accomplished by the agents. We also evaluate the Performance Hypothesis by considering the relationship between sensing and task performance for each approach. We begin by presenting the task performance results in Figure 5.7. We note that unlike the sensing performance results, these results are not statistically significant as shown in the two-way ANOVA results in Table 5.2.

Table 5.2 Two-way ANOVA Results for Task Performance in MineralMiner

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Environment	589.1269	5	117.8254	0.057094	0.997901	2.222674
Approach	12142.52	5	2428.503	1.176761	0.318501	2.222674
Interaction	1631.79	25	65.27159	0.031628	1	1.516665
Within	2154521	1044	2063.717			
Total	2168884	1079				

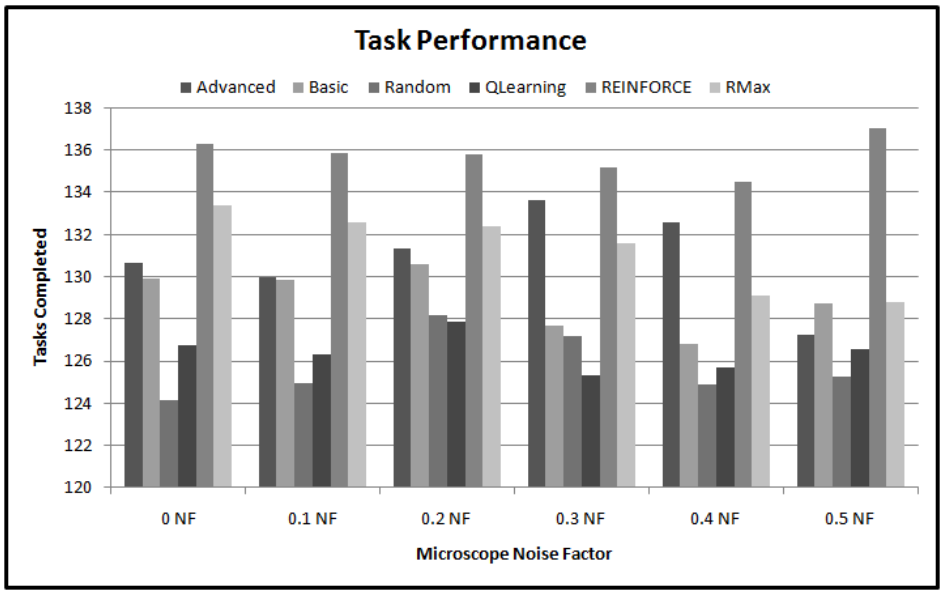

Figure 5.7 Task Performance in MineralMiner

Considering the task performance of the sensing activity selection approaches, we observe the following:

- REINFORCE and RMax generally have the best task performance, except in the 0.3 and 0.4 NF environments when Advanced performs between the two approaches. We believe that the better performance by these two RL approaches is due to better sensing by the agents through RL, even if that better sensing doesn't always correspond to higher average value of knowledge refinement (although for REINFORCE, this average is higher than the other approaches in general). The improved performance by Advanced appears to just be a random outlier (recall that we lack statistical significance for these results).
- Unlike the sensing performance results from the previous section, there are no observable general trends between the amount of Observer Effect and task performance. In other words, unlike sensing performance, increasing the Observer Effect does not generally result in worse task performance in the environments considered. This indicates that the agents are able to suffer from some sensing

distortion causing lower agent sensing performance (e.g., at the higher Observer Effect environments 0.4, 0.5 NF) but still create good enough knowledge to accomplish some tasks. However, we hypothesize that if we pushed the Observer Effect further past 0.5 NF, we would eventually observe that the task performance of agents decreases with increasing Observer Effect. We believe that the 0.5 NF was not sufficient for this behavior because in Figure 5.1 we observe that the average accuracy of the microscope was still above 50% for this environment but should continue to decrease as the noise factor increases.

Next we look at the task performance results over time, shown using the 0.3 NF environment again as an example in Figure 5.8 for all of the approaches and in Figure 5.9 comparing the RL approaches to Random. We observe:

- Over time, all of the approaches follow the same general trend: worse task performance. This is due to some agents running out of energy or having insufficient minerals collected necessary for recharging their batteries and renting tools, as well as mines running out of mineral supply. Within this trend, none of the approaches consistently outperform the others, as with sensing performance.
- Unlike sensing performance, RL does not continually improve task performance over Random, even though sensing performance is increased (Figure 5.4). This is explained in the correlation results analyzed next.

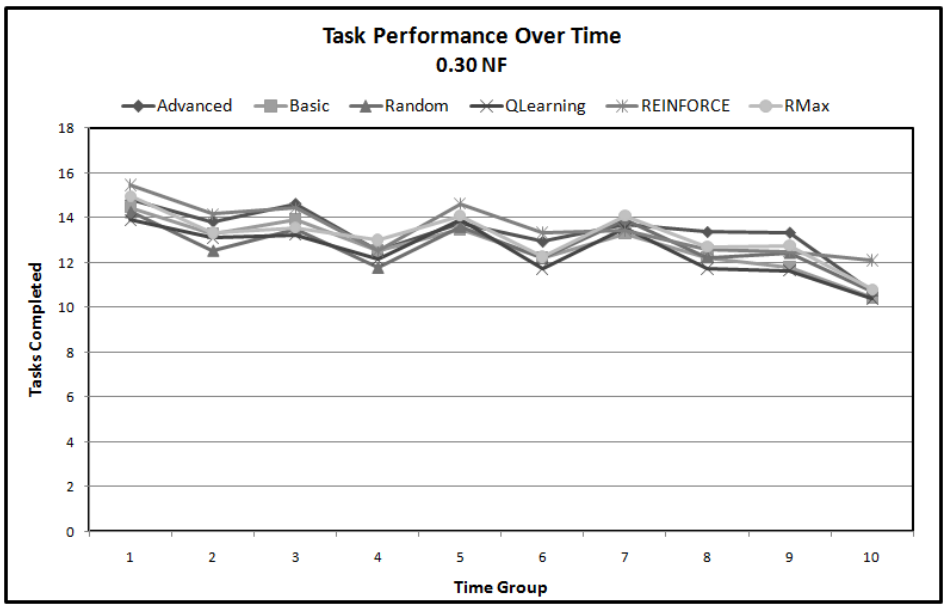

Figure 5.8 Task Performance over Time in MineralMiner (0.3 NF)

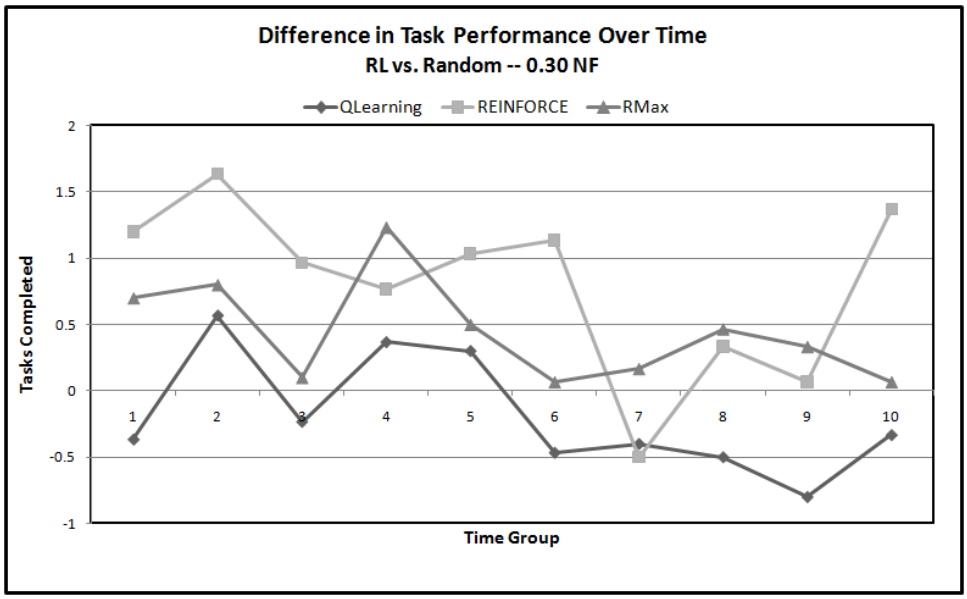

Figure 5.9 Task Performance of RL vs. Random over Time in MineralMiner (0.3 NF)

To further explain our task performance results, we consult the correlation between sensing and task performance, shown in Table 5.3. From these correlation results, we observe:

Table 5.3 Correlation Between Sensing and Task Performance in MineralMiner

Approach	0 NF	0.1 NF	0.2 NF	0.3 NF	0.4 NF	0.5 NF
Advanced	0.1631	0.0252	-0.0984	-0.4004	-0.2987	-0.3787
Basic	-0.1844	-0.2241	-0.2427	-0.1219	-0.0878	-0.1758
Random	0.2250	0.2477	0.1439	0.0443	-0.0534	0.0558
Q-Learning	0.0433	-0.1070	-0.0779	0.1340	-0.0019	-0.0437
REINFORCE	0.1153	0.1917	0.1301	0.2321	0.1410	0.2608
RMax	0.0221	0.0982	-0.0036	-0.1183	-0.0984	-0.2465
Total	0.0654	0.0349	-0.0259	-0.0404	-0.0218	-0.0978

- For all but the REINFORCE approach, there exists no general correlation between sensing and task performance. We believe that this occurs for the following reason: the trend for sensing performance is decreasing as Observer Effect increases, while no similar trend exists for task performance. Thus, although sensing performance becomes worse as more Observer Effect is present, task performance is mostly unchanged. This seems to imply that even relatively bad sensing can provide some level of information for knowledge revision which allows the agents to build good enough knowledge to accomplish some tasks, but not all.
- However, for the REINFORCE approach, we observe a general positive correlation between sensing and task performance. Thus, since REINFORCE generally has some of the best sensing performance (Figure 5.3), it is able to convert its improved sensing into accomplishing more tasks. Compared to our previous observations, this implies that although all sensing seems sufficient to accomplish some tasks, better sensing can yield even better tasks.

Based on these results, we reject the Performance Hypothesis because of the general lack of correlation between sensing and task performance. However, we note that one RL

algorithm is able to leverage improved sensing performance to raise task performance. In the future, we intend to further investigate why this algorithm (REINFORCE) achieved this result and how we might be able to replicate that in other approaches to sensing activity selection.

5.1.4. MineralMiner Results Discussion

From the results presented above, we have confirmed the Observer Effect Hypothesis for a *fully observable* environment where the Observer Effect is relevant (i.e., non-negligible but not overpowering). We also note that sensing performance is positively correlated with task performance for the REINFORCE approach, thus improved sensing by considering the Observer Effect can yield better task performance, a desired emergent behavior. Therefore, the Observer Effect and its Tradeoff are challenges to sensing necessary to consider for stateful resources. Furthermore, our Observer Effect MDP is beneficial for doing so.

However, one question remaining is what type of RL algorithm is best for solving the Observer Effect MDP? In Section 3.3.2, we proposed that any RL algorithm is appropriate, leaving the choice up to the agent designer. However, from these results, we can draw some conclusions about the various algorithms used, chosen as examples of various types of RL algorithms. In both sensing and task performance, we observe that the REINFORCE algorithm performed the best, which we attribute to its ability to handle continuous states, such as microscope energy in our sensing state. The other two algorithms instead discretize sensing state which loses some information useful for distinguishing between states. Finally, we note that model-based RMax generally outperformed model-free Q-Learning, which we attribute to the increased amount of information about the environment (transition probabilities) learned by RMax. Therefore, we now hypothesize that model-based and/or continuous RL algorithms are most appropriate for solving the Observer Effect MDP and intend to further investigate with a wider range of RL algorithms.

5.2. UserRec Results

In this section, we present the results of the UserRec experiments described in Section 4.2. Since this simulation environment is based on previous work by Doshi and Roy (2008), we first briefly test our implementation by comparing the overlapping experiments between our work and theirs to validate that the simulation was implemented properly. Then, as with the MineralMiner results, we also briefly validate the existence of the Observer Effect within the UserRec experiments. We follow with our two objectives to evaluate the Observer Effect Hypothesis (c.f., Section 2.3.3) and the Performance Hypothesis (c.f., Section 4.1.3). Again, we evaluate the former by considering the sensing performance of the various sensing activity selection approaches. Then, we evaluate the latter by considering the task performance of the approaches, as well as the correlation between sensing and task performance. We conclude this section with a brief discussion summarizing the lessons learned from the UserRec experiments.

5.2.1. Implementation Validation

We begin by briefly validating our implementation of the user preference elicitation simulation described by Doshi and Roy (2008) by demonstrating that we receive similar results on the experiments conducted in the original paper. Here, we consider the Zero Frustration User environment which does not include the Observer Effect but does use the same simulation parameters as Doshi and Roy (2008), provided in Tables 4.3 and 4.4 in Section 4.2.1. Specifically, we consider the PBVI approach.

In the original paper (Doshi and Roy, 2008), the agent's performance is evaluated based on the median reward received over all episodes. In our experiments, we observed a median reward per run (with a given 95% confidence interval) of 92.8667 +- 0.3483 (averaged over all 30 runs) which matches the median reward achieved by Doshi and Roy between 90 and 95. Thus,

from these results, we can deduce that our implementation produces similar results to those presented in the original paper.

5.2.2. Observer Effect Validation

As in the MineralMiner results (c.f., Section 5.1.1), we also briefly validate the existence of state-dependent behavior of the user resource and the Observer Effect within our simulation results. Recall that in the UserRec experiments (c.f., Section 4.2.1), the user is a stateful resource whose behavior depends on her current frustration level. User frustration results in both faster responses (i.e., reduced delays) to interruptions used by the agent to elicit the user's preference, as well as less accurate responses by the user (c.f., Section 4.2.1 for justification). Here, the latter represents the Observer Effect in this simulation. The changes in user frustration are governed by the type of user which we vary in our experiments.

To validate the existence of state-dependent behavior–including the Observer Effect–in our UserRec experiments, we present the average user response delay and accuracy over all environments and approaches, shown in Figures 5.10-5.11, respectively. We observe that for all of the approaches, when the user can become frustrated (non-Zero User Frustration Types: Patient, Task, and Angry), both her response delays and accuracy are reduced when compared to the *non*-frustrated Zero User Frustration Type, as desired. Thus, our experiments do contain state-dependent resource behavior, including the Observer Effect.

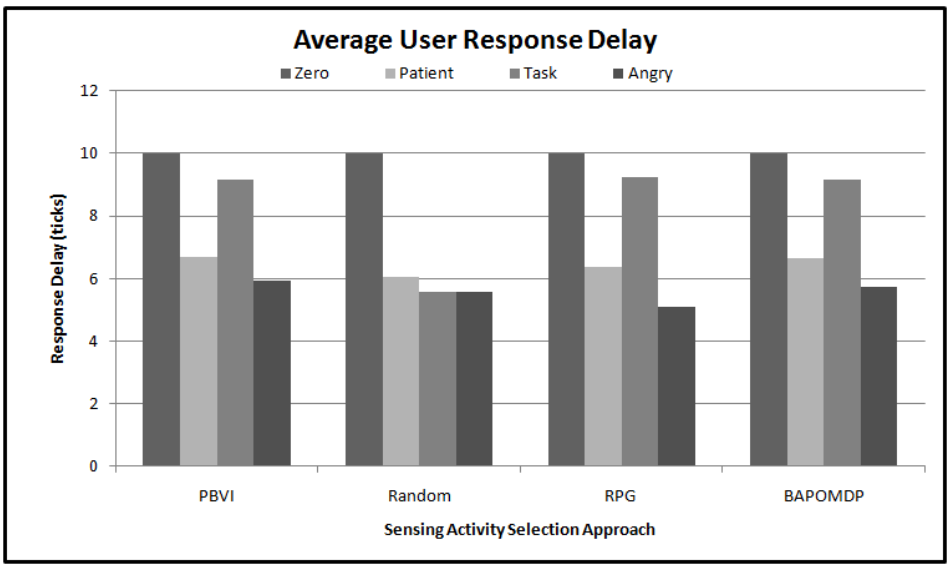

Figure 5.10 Average User Response Delay in UserRec

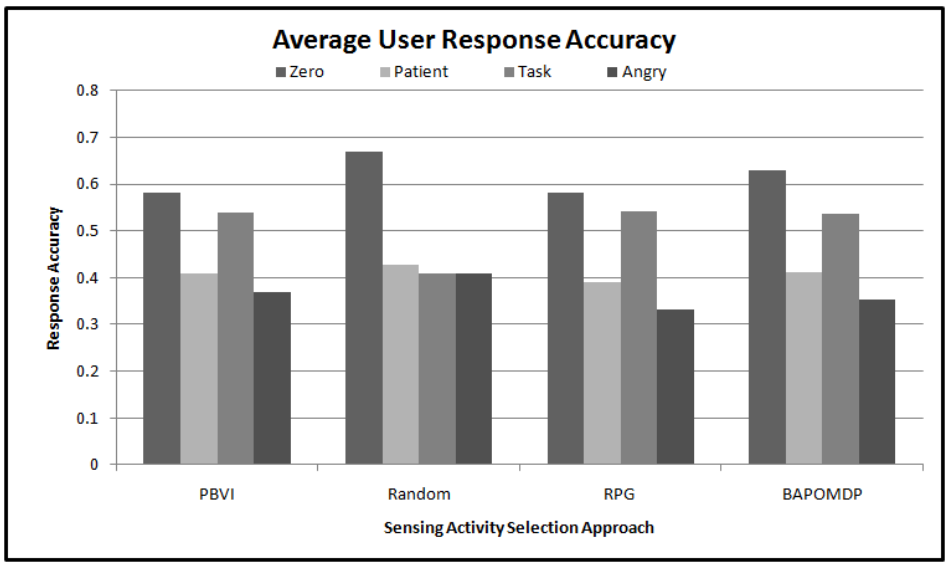

Figure 5.11 Average User Response Accuracy in UserRec

5.2.3. Objective UR1

Next, we start the UserRec results analysis by considering the sensing performance of the various sensing activity selection approaches, again represented by the average value of knowledge refinement per sensing activity. We present these results across all approaches and environments in Figure 5.12. We note that unlike the MineralMineral results (c.f., Section

5.1.2), these results are not statistically significant as shown in the two-way ANOVA results in Table 5.4.

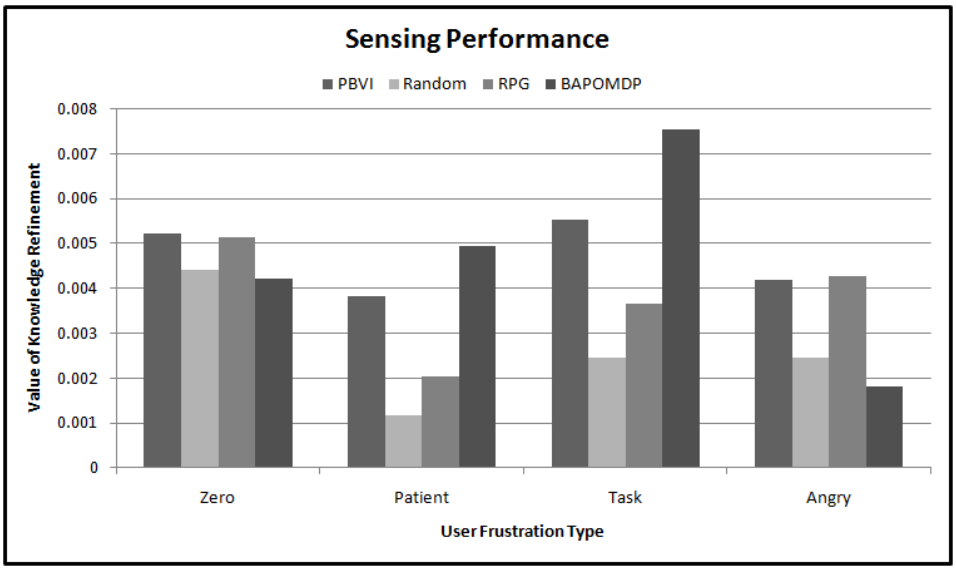

Figure 5.12 Sensing Performance in UserRec

Table 5.4 Two-way ANOVA Results for Sensing Performance in UserRec

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Environment	0.000523	3	0.000174	0.469968	0.703361	2.624124
Approach	6.68E-05	3	2.23E-05	0.059996	0.980729	2.624124
Interaction	0.000132	9	1.47E-05	0.03965	0.999993	1.900058
Within	0.17215	464	0.000371			
Total	0.172872	479				

From these results, we again observe that an agent using PORL to solve the Observer Effect POMDP (BAPOMDP) performs better knowledge refinement than agents which do not consider the Observer Effect (PBVI, Random) when it is necessary to consider this effect and the effect is not over-powering; that is, when the user is not of type Zero or Angry (see further discussion for explanation). Specifically:

- In the Zero Frustration User environment, there is no Observer Effect because the user does not become frustrated by human-agent interactions, so her response delay and accuracies do not vary over time. Thus, there is no reason to consider the Observer

Effect, causing the non-PORL approaches to perform very similarly to the PORL approaches. However, this similarity is important because it shows that *considering the Observer Effect even when it is not present does not degrade agent sensing performance.*

- In the Patient and Task-oriented User environments, the user is frustrated by interruptions caused by agent sensing activities, thus producing an Observer Effect. Here, we observe that the model-based BAPOMDP approach is able to outperform the other approaches in sensing-performance. However, we also note that the model-free RPG approach does not. We believe that this is due to the extra parameter learning performed by the model-based approach, giving it more information about the environment to properly make decisions (i.e., just learning the expected value of knowledge refinement is not enough).
- In the Angry User environment, we see similar performance to the Zero Frustration User environment where the non-PORL and PORL approaches (PBVI vs. RPG, Random vs. BAPOMDP) perform similarly. This is due to the fact that none of the solutions last near the 200 episodes before stopping because they exceed the user's frustration boiling point, confirmed in Figure 5.13 which presents the average duration of each simulation run. Thus, we believe the PORL solutions do not have enough time to learn how to handle user frustration before the user is too frustrated to continue. In fact, it appears that the little bit of learning done by BAPOMDP actually drives down sensing performance (compared to Random) due to the difficulty of learning over a small number of episodes during the high frustration of the user. We also note that the shortness of the experiments due to exceeding the frustration boiling point for this

environment compared to the others makes it difficult to draw any strong conclusions from its results.

Based on these results, we again conclude that considering resource state and the Observer Effect can improve sensing performance and *confirm the Observer Effect Hypothesis*, but note that, as observed in the MineralMiner results (c.f., Section 5.1.2), low levels of Observer Effect do not necessitate this consideration and high levels make it difficult to do so. In the Angry User environment, we posit that the PORL approaches' (especially BAPOMDP) poor performance was due to a lack of episodes to properly learn how to handle user frustration during sensing. In the future, we intend to investigate possible ways to speed up agent learning to avoid this problem, or strategies to lower user frustration when it appears the user is about to reach her boiling point and stop using the system.

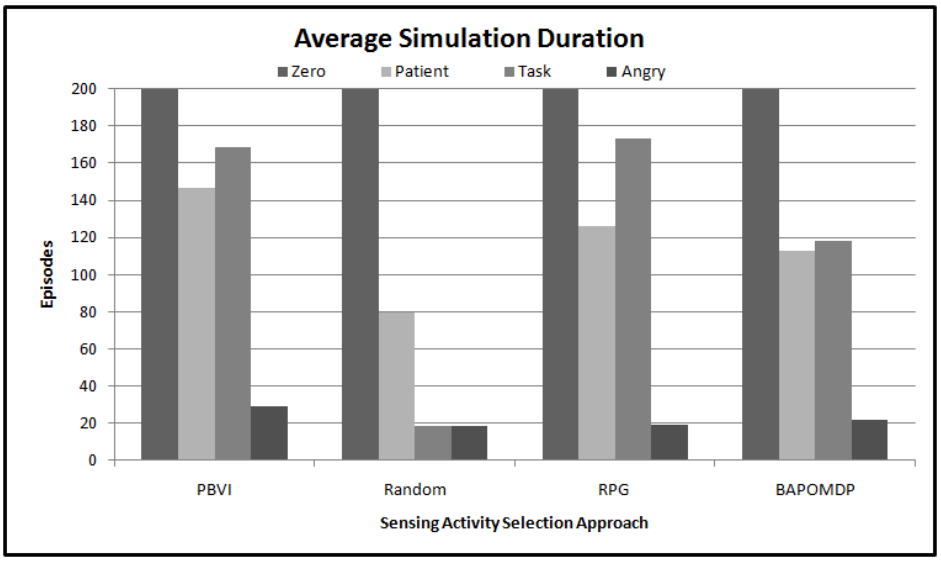

Figure 5.13 Average Simulation Duration in UserRec

Next, we consider the sensing performance of the various approaches over time. In the UserRec experiments, we evaluate the time series over a sequence of episodes split into Episode

Groups, where each group consists of 20 successive episodes[5]. We present these results from the Task-oriented User and Patient User environments in Figures 5.14-5.15 as examples (c.f., Appendix A.2 for the other environments) as these were the most interesting results. Please note that in these figures, Random has fewer data points due to the fact that none of its simulation runs ever survived all 200 episodes before exceeding the user's frustration boiling point. From these figures (and those given in the appendix), we make the following observations with respect to the Observer Effect Hypothesis:

- For the Task-oriented User Frustration environment, each of the approaches generally retained the same rank in comparison to the others and the best PORL approach was almost always on top. This differs from the results from MineralMiner (c.f., Section 5.1.2) where no such dominance occurred. Thus, for this environment, the Observer Effect Hypothesis holds over time, as well. However, we note that this only occurred for this environment and not the other three (e.g., Figure 5.15), so in general, the UserRec results match those from MineralMiner in that no PORL approach always performs the best at all time points. Thus, the Observer Effect Hypothesis still generally holds just for cumulative results.
- In contrast, in the Patient User Frustration environment, each of the approaches tends to do worse over time. This is caused by a steady increase in user frustration over time in this environment, shown in Figure 5.16. This frustration increase, in turn, is caused by frustration from the agent's sensing activities being similar in magnitude to the reduction in frustration resulting from correct submissions (c.f., Table 4.9 in Chapter 4). Thus, correct submissions do not remove enough frustration in the user, in spite of the fact that all of the frustration increase values are small. However, it does appear the

[5] As in MineralMiner, we chose a group size of 20 in order to have 10 groups for the time series.

BAPOMDP approach has a resurgence towards the end of the simulations due to a decrease in frustration at the end of the simulations in Figure 5.16 for this approach and none of the others, but this might be an outlier as we cannot find an appropriate justification.

- In the Task-oriented User Frustration environment, on the other hand, the reduction in frustration from correct submissions is greater while the frustration increase values from sensing stay the same. This enables correct submissions to offset frustration incurred from sensing interruptions, even though the potential increase in frustration from wrong submission is greater. Thus, the frustration levels of the user in the Task-oriented environment remain low over time, shown in Figure 5.17. From this result, we can conclude that *the relative rates of change in resource state (user frustration) play an important role in the behavior of the resource and its impact on sensing*, as one may expect.
- Because the Random approach increases user frustration so quickly due to poor sensing choices, its simulation runs do not last as long as the others. Thus, for many time points, there is no data for the Random approach, so we cannot evaluate how the PORL approaches perform against the baseline Random approach over time as we did for MineralMiner.

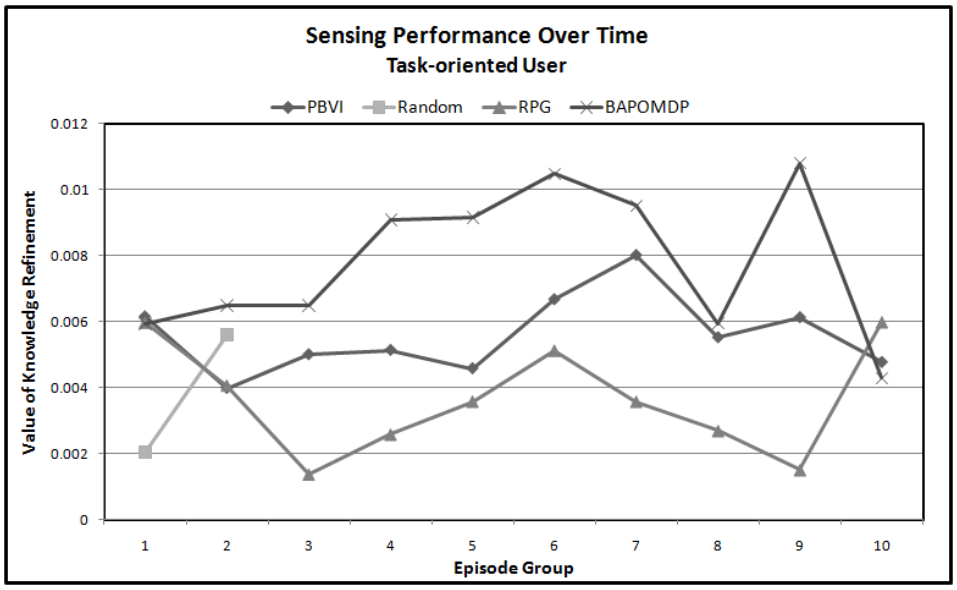

Figure 5.14 Sensing Performance over Time in UserRec (Task-oriented User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

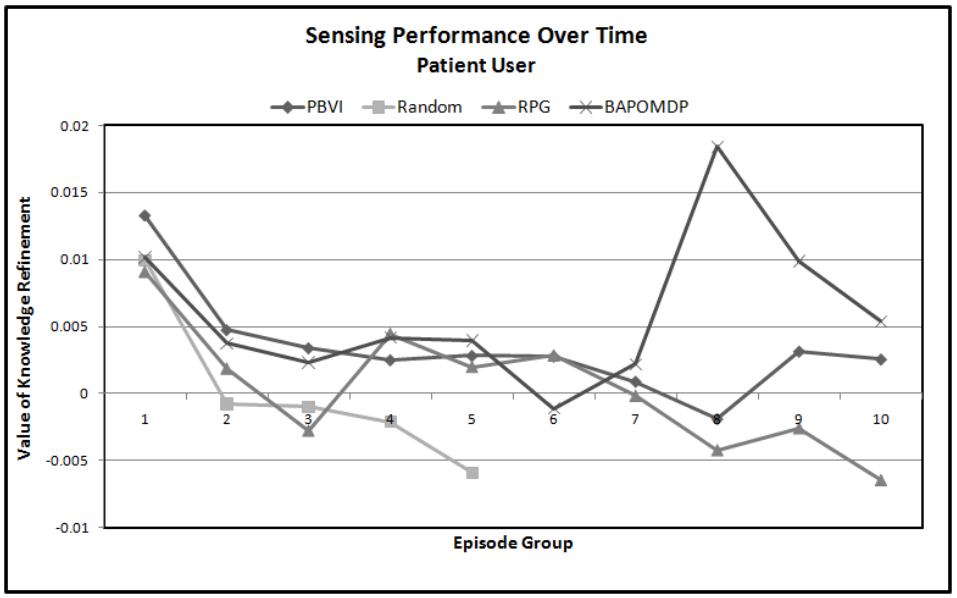

Figure 5.15 Sensing Performance over Time in UserRec (Patient User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

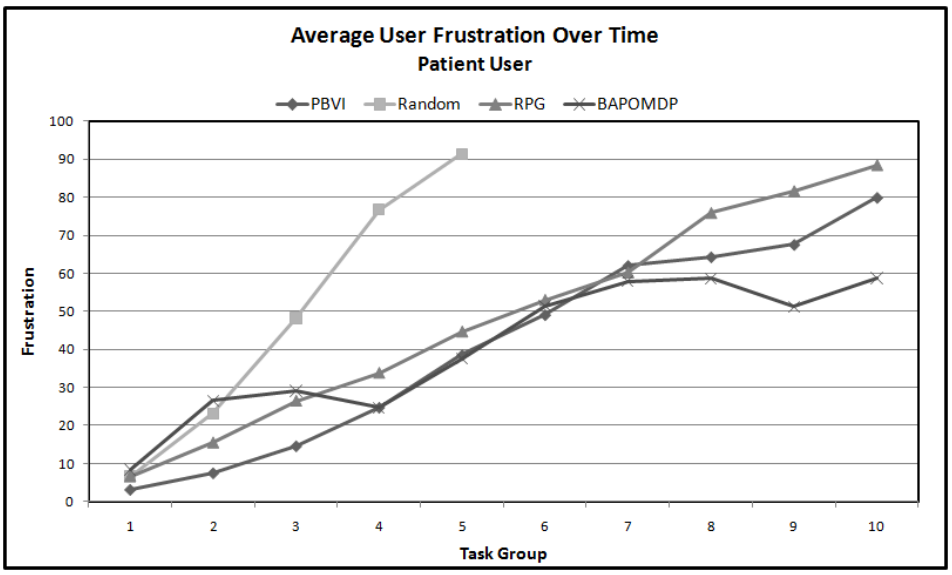

Figure 5.16 User Frustration over Time in UserRec (Patient User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

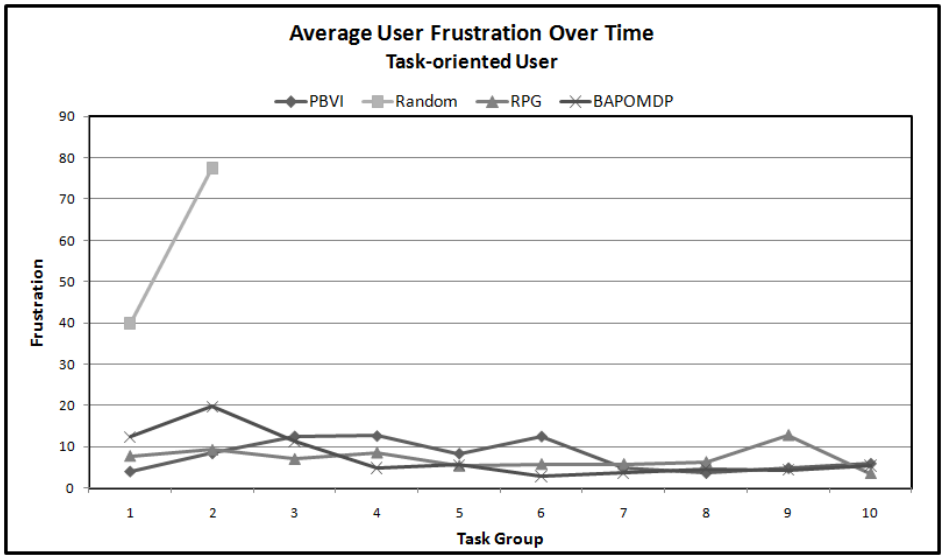

Figure 5.17 User Frustration over Time in UserRec (Task-oriented User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

5.2.4. Objective UR2

Next, we consider the task performance of the sensing activity selection approaches in UserRec, represented as 1) the number of correct submissions by the agents, and 2) the average

task reward[6] for a submission. Here, the first measures how often the agent was successful, and the second measures how efficient they were (since negative task rewards are received for sensing). We also evaluate the Performance Hypothesis by considering the relationship between sensing and both types of task performance for the various approaches. We begin by presenting the task performance results in Figures 5.18 and 5.19. We note that unlike in MineralMiner (c.f., Section 5.1.3), these results are statistically significant ($\rho < 0.005$ but with a significant interaction between approach and environment) as shown in the two-way ANOVA results in Tables 5.5 and 5.6.

Table 5.5 Two-way ANOVA Results for Task Performance (Correct Submissions) in UserRec

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Environment	1348963	3	449654.4	429.682	1.7E-133	2.624124
Approach	69654.77	3	23218.26	22.18697	1.93E-13	2.624124
Interaction	55537.28	9	6170.808	5.896718	8.22E-08	1.900058
Within	485567.6	464	1046.482			
Total	1959723	479				

Table 5.6 Two-way ANOVA Results for Task Performance (Average Task Reward) in UserRec

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Environment	936346.4	3	312115.5	190.1171	2.17E-80	2.624124
Approach	87955.38	3	29318.46	17.85859	5.51E-11	2.624124
Interaction	44672.59	9	4963.622	3.023463	0.001603	1.900058
Within	761749.2	464	1641.701			
Total	1830724	479				

Considering the task performance of the sensing activity selection approaches, we observe the following:

[6] Recall that task reward is defined as an amount earned by the agent for performing each action. Please do not confuse this with the value of knowledge refinement reward used in the Observer Effect MDP/POMDP. The former occurs at the task level of agent activities, while the latter is at the sensing level.

- Unlike in MineralMiner (c.f., Section 5.1.3), including the Observer Effect does play an important role in agent task performance. We observe that the best task performance is achieved when there is no Observer Effect (Zero User Frustration Type) and overall task performance decreases as the impact of Observer Effect increases: Patient and Task-oriented are worse than Zero and Angry is the worst. This implies that for some applications, the Observer Effect does affect the ability of agents to accomplish their tasks and goals.
- Comparing the various baseline approaches, PBVI generally performs the best (or closely behind), due its focus on task performance through maximizing task rewards. Random, on the other hand, does very poorly due to over-frustrating the user, evidenced by shorter simulation durations (Figure 5.13) as previously described.
- Considering the PORL approaches, we observe that RPG generally performs close to PBVI. This demonstrates that *focusing on sensing performance is another means to good task performance*, in spite of the fact the RPG approach doesn't explicitly consider this metric when choosing how to sense. While it does not appear upon first glance that this conclusion holds for BAPOMDP as well, which performs worse than both PBVI and RPG, we discuss next that this is due to decreased performance while BAPOMDP is learning, after which it exploits its learning to perform at least as well as the other approaches. Thus, if we were to extend the number of episodes, the gap between BAPOMDP and the better approaches would shrink.

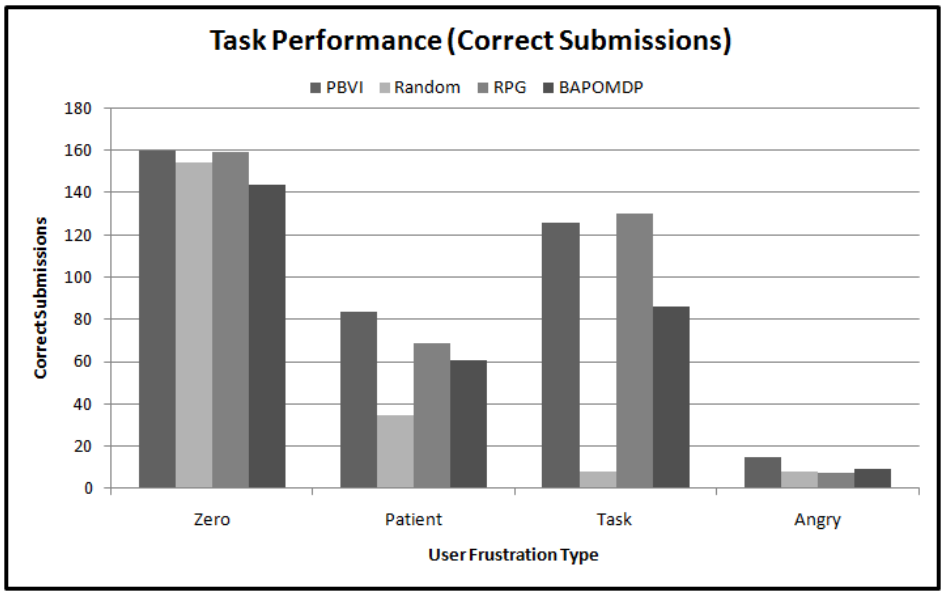

Figure 5.18 Task Performance in UserRec (Correct Submissions)

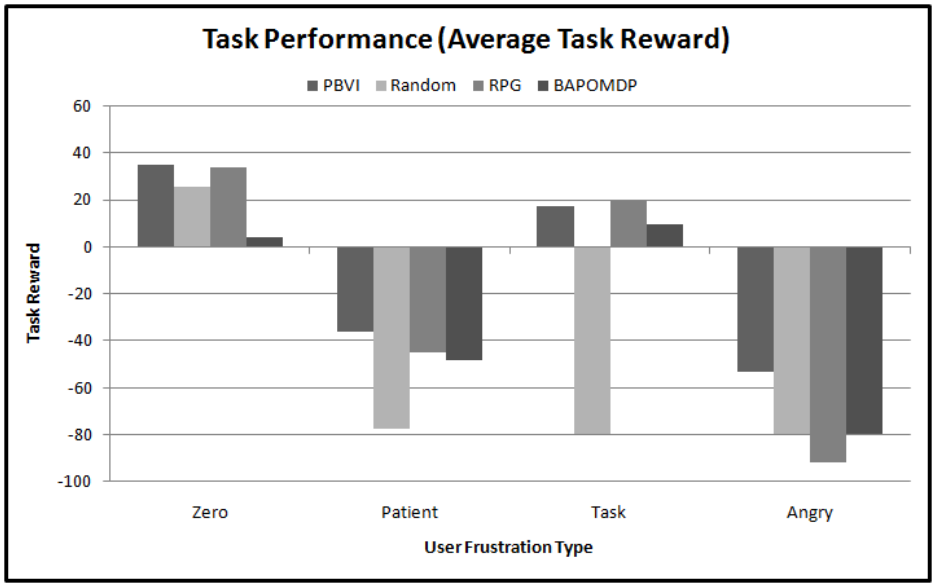

Figure 5.19 Task Performance in UserRec (Average Task Reward)

For task performance over time, we present these results for the Task-oriented and Patient User Frustration environments in Figures 5.20 and 5.21. Please note that we only present the correct submissions results because the average task reward results look the same but with different y-axis scales. Again, all of the other results are available in Appendix A.2. From these results, we observe:

- Task performance is strongly related to user frustration for each episode group. For each of the approaches, we observe from Figure 5.17 that in the Task-oriented User Frustration environment, user frustration remained low (except for Random which over-frustrated the user). This low frustration leads to better sensing performance (Figures 5.12 and 5.14) and also better task performance (Figure 5.20).
- For the Patient environment, we also observe a dependence on frustration in task performance for each episode group. From Figure 5.16, we observe that user frustration increases over time, leading to not only worse sensing performance over time (Figure 5.15), but also worse task performance (Figure 5.21).
- Again, as with MineralMiner, none of the approaches outperforms the others consistently over time. However, we do note that BAPOMDP suffers from worse performance at the beginning of the simulations in both environments, then performs at least as well as the others for the rest of the episode groups. We believe that this is due to the effect of learning on the approach – while the agent is learning, it performs worse than the PBVI and RPG approaches, but still better than Random indicating that it has learned something positive. Then, once the agent has learned enough, it is able to successfully exploit this knowledge to have good task performance relative to all other approaches considered. A longer learning period for BAPOMDP than RPG makes sense because the former is a model-based PORL approach and thus learns a parameterized POMDP model, while the latter is model-free PORL approach and learns only the value of knowledge refinement function and a controller for sensing activity selection.

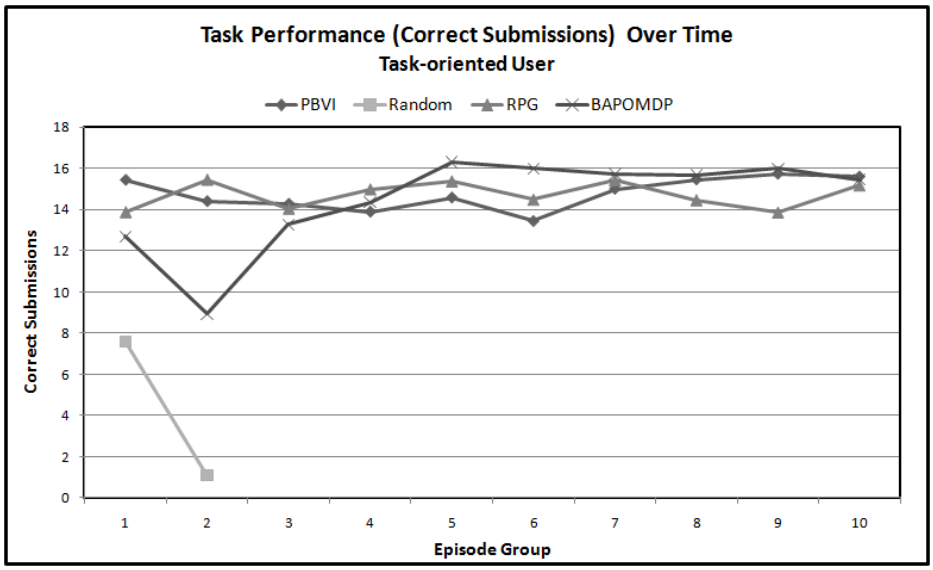

Figure 5.20 Task Performance over Time in UserRec (Correct Submissions, Task-oriented User)

Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

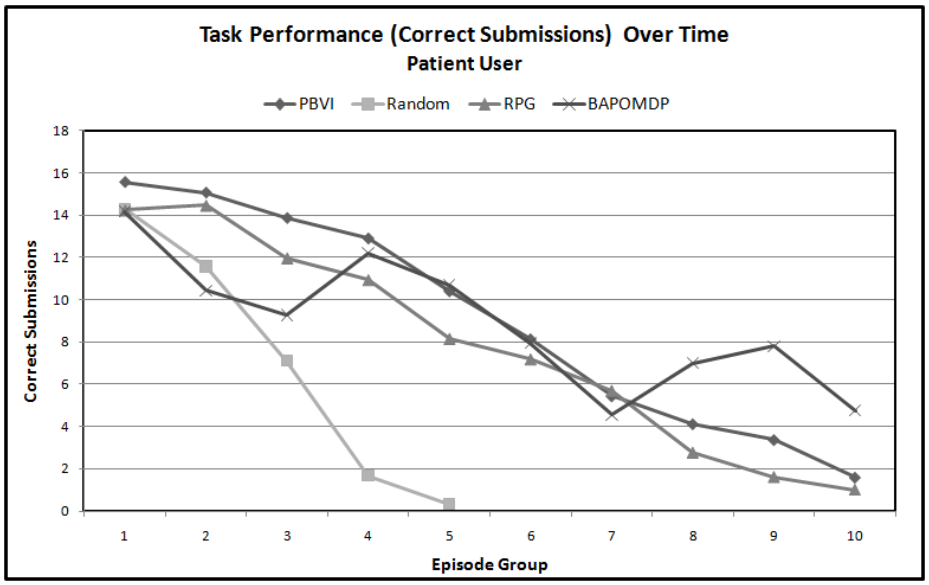

Figure 5.21 Task Performance over Time in UserRec (Correct Submissions, Patient User)

Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

Finally, we present the correlation results between sensing performance and both types of task performance: 1) correct submissions and 2) average task reward in Tables 5.7 and 5.8, respectively. From these results, which again are not statistically significant, we observe:

Table 5.7 Correlation between Sensing and Task Performance (Correct Submissions) in UserRec

Approach	Zero	Patient	Task	Angry
PBVI	-0.0656	0.1375	0.1323	0.1318
Random	0.1543	-0.2924	0.1838	0.1660
RPG	-0.0218	-0.0611	-0.2386	0.4516
BAPOMDP	0.0471	0.1826	0.2095	0.1334
Total	0.0246	0.1083	0.1047	0.2019

Table 5.8 Correlation between Sensing and Task Performance (Average Task Reward) in UserRec

Approach	Zero	Patient	Task	Angry
PBVI	0.02572	0.1357	0.1370	0.1800
Random	0.2884	-0.3842	0.1280	0.1006
RPG	-0.0280	-0.1404	-0.2142	0.3512
BAPOMDP	0.0787	0.1941	0.1881	0.1573
Total	0.0541	0.0695	0.0991	0.1967

- In general, a positive correlation exists between sensing and task performance when the user can become frustrated (i.e., all environments except Zero), but not when there is no Observer Effect (i.e., Zero User Frustration Type). Thus, considering sensing performance is important when stateful resource behavior affects sensing outcomes, and increasing sensing performance is a potential means to increase task performance. However, because these correlations are not very large (< 0.2), increasing sensing performance is not guaranteed to increase task performance.
- However, contrary to this general trend, RPG's sensing performance is not positively correlated with task performance. This is further evidenced when comparing the sensing and task performances of this approach (Figures 5.13, 5.18-5.19) where we

observe that model-free RPG achieves higher overall task performance but worse sensing performance than model-based BAPOMDP.

- Finally, although the Angry environment has the highest correlations, these are not very conclusive due to the shortness of these experiments (Figure 5.13). However, they do imply that at least in the short term, sensing performance might be correlated to task performance, but compared to our other results, that correlation might be reduced over time. We cannot fully support this implication without further investigation, however. Thus, we intend to test this implication in future work by varying the number of episodes to determine how correlation varies with the number of interactions between human and agent.

Based on these results, we observe some evidence in support of the Performance Hypothesis based on the general positive correlation between sensing and task performance. However, as these correlations are small, we cannot draw any significant conclusions. Further, we note that one of the PORL approaches (BAPOMDP) had the highest positive correlation, similar to our MineralMiner results (where REINFORCE was the highest).

5.2.5. Experimental Setup

From the UserRec results presented above, we have confirmed the Observer Effect Hypothesis for *partially observable* environments where the Observer Effect is relevant (i.e., non-negligible but not overpowering). We also note that sensing performance is generally positively correlated with task performance for all but the RPG approach, thus improved sensing by considering the Observer Effect can yield better task performance, a desired emergent behavior. This result is demonstrated in the task performance results over time as BAPOMDP, which has the highest sensing performance, also has the highest task performance after going through its learning period. Therefore, we have more evidence that the Observer Effect and its

Tradeoff are challenges to sensing necessary to consider when using stateful resources and our Observer Effect POMDP is beneficial in doing so.

Comparing the two PORL approaches considered in the UserRec experiments, we have more evidence confirming our conjecture from Section 5.1.4 that model-based approaches outperform model-free approaches because our model-based BAPOMDP approach was the best in terms of sensing performance, with the caveat that they might (and did in our experiments) take longer to learn given that they are learning more from the environment. Therefore, we still hypothesize that model-based RL algorithms are most appropriate for solving the Observer Effect MDP/POMDP. However, since we have only considered a few approaches, we still intend to further investigate with a wider range of RL/PORL algorithms in both fully observable and partially observable environments.

5.3. Discussion

Considering the results from both the MineralMiner and UserRec simulations, we see several similarities. First, for both simulations, we have observed that the RL- or PORL-based approaches have higher sensing performance than non-RL/PORL approaches because the former consider the value of knowledge refinement impacted by resource state through the Observer Effect, while the latter do not. This result is somewhat to be expected due to the fact that the RL/PORL approaches directly intend to optimize sensing performance through picking sensing activities maximizing this value. However, it is important for three reasons. First, it confirms the Observer Effect Hypothesis stating that such improvement in sensing performance is possible, which is the motivation of this research. Second, it demonstrates that our Observer Effect MDP with RL solution is one way of improving sensing performance. Finally, it shows that the approach works not just in theory but also in practice, a fact not always observed for complex, real-world environments such as those simulated by our experiments (especially

UserRec). In the future, we intend to further explore this fact by applying our approach to real-world (non-simulated) environments, such as an intelligent user interface for supporting collaborative research (c.f., Chapter 7).

Next, we observed in both simulations that task performance isn't always correlated with sensing performance, so increasing sensing performance alone does not necessarily generate an improvement in task performance. This goes against our Performance Hypothesis which motivates research in agent sensing in general. However, we did observe evidence that at least for some approaches, including REINFORCE in MineralMiner and BAPOMDP in UserRec, increasing sensing performance enough above that experienced for non-RL approaches yields an additional task performance boost by creating a positive correlation between sensing and task performance. Thus, although any sensing approach can sometimes provide quality information for refining agent knowledge, boosting the quality of sensing eventually yields a boost in task performance, even if that boost isn't necessarily as large.

Third, considering the relative performance of the various RL/PORL algorithms employed in our experiments, we observed that in general, model-based approaches outperform model-free in sensing performance and sometimes task performance. Thus, in spite of our earlier conjecture (c.f., Section 3.3.2) that any type of reinforcement learning algorithm is applicable, it appears that some are better than others. As previously stated, we intend to further explore (by considering wider range of algorithms) what types of algorithms are most appropriate based on the properties of different types of environments in order to provide a set of guidelines for better solving the OETP through the Observer Effect MDP/POMDP.

Finally, we note that the same general results we observed for both sets of experiments, in spite of the fact that the two simulation environments and experimental setups differ in several key properties, including:

1) the observability of resource state (i.e., fully observable vs. partially observable),
2) the types of knowledge representation (i.e., possibility logic (Josang, 2001) vs. probabilistic belief states),
3) the specific RL/PORL algorithms used, and
4) the MAS applications represented (i.e., robotic exploration vs. intelligent user support).

Thus, we believe that our Observer Effect MDP solution can apply to a wide range of applications and real-world scenarios which rely on using stateful resources during agent sensing.

Chapter 6 Related Work

In this chapter, we describe work from the artificial intelligence and multiagent systems literature that is most closely related to ours. This includes research from the following areas: 1) anytime sensing, 2) value of information driven sensing, 3) the Observation Selection Problem, and 4) multiagent resource allocation. The relationship between these four areas and our research in summarized in Figure 6.1. We note that prior work in bounded rationality and general metareasoning in artificial intelligence are also related to our work, but these were already discussed in Chapter 2.

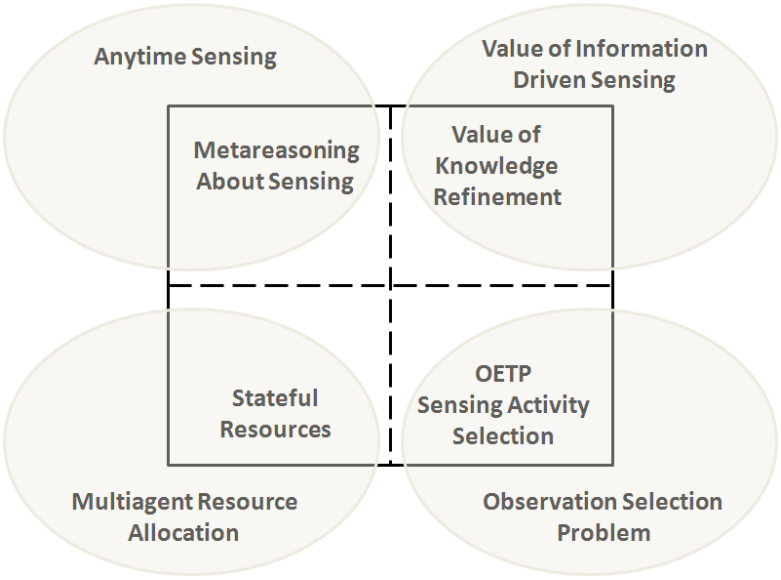

Figure 6.1 Relationship Between Our Research and Related Work

First, anytime sensing represents prior work in metareasoning about sensing. However, their research considers only computational resource usage and not stateful resource usage, and thus no Observer Effect. Second, prior work involving value of information driven sensing chooses sensing activities which try to optimize the value of information gathered during sensing, similar to our goal of optimizing the value of knowledge refinement. However, they only consider stateless resource costs for sensing. Third, the Observation Selection Problem represents prior work in selecting observations to perform in order to optimize an objective

function given various constraints (including costs), similar to our Observer Effect Tradeoff Problem. However, they work with monotonic, submodular objective functions which cannot include the non-monotonic Observer Effect. Finally, resource allocation represents a broad field covering reasoning about how to allocate resources between agents and/or tasks to accomplish agent goals. Our research extends work in this area by considering an additional resource characterization: *stateful* resources, and is closely related to recent work involving the use of MDPs to model decisions about resource allocation.

6.1. Anytime Sensing

First, one of the closest related areas of research to our own is prior work involving the control of anytime algorithms as a metareasoning strategy for controlling the use of stateful computational resources during sensing. Specifically, Zilberstein (1996; with Russell, 1993) considered the problem of how to decide how much time to allocate to sensing and planning in a robot in order to accomplish the robot's tasks (e.g., navigate to a location) as quickly as possible. Here, the robot views its environment with a camera and the quality of the information gathered through the camera depends on the amount of time spent processing the raw pixels observed. Similarly, the quality of the agent's plan for navigating through its environment depends on the amount of time spent on planning, as well as the quality of the information gathered through sensing. Balancing the time spent by the robot on sensing, planning, and movement in this scenario is challenging due to the tradeoff between 1) the need to spend time *now* on sensing and planning to create better movement plans in order save time *later* by avoiding the need to resense and replan if the robot runs into an obstacle, versus 2) spending too much time now on sensing and planning, delaying the robot from reaching its goal. To overcome this problem, Zilberstein considered both the processing of sensed pixels and planning procedures to be anytime algorithms, assuming that the quality of information

produced by processing the raw pixels and the quality of the plan will never decrease with additional resource usage (i.e., time spent). This allowed them to model the robot's ability to process sensed information with a monotonic performance profile dependent on the amount of time spent on processing. They also modeled the robot's planning with a conditional performance profile which depends on both the amount of time spent, as well as the input quality from its sensing. Using these performance profiles, generated through experience with the environment, the agent could then solve for an ideal amount of time to spend on each in order to reach its goal as fast as possible.

Comparing this work to ours, we note that theirs is appropriate for applications where additional sensing activities always produce better or no worse information than the agent already has, such as the processing of vision data. In environments where this assumption breaks down, however, sensing can no longer be represented as an anytime algorithm and using monotonic performance profiles to select an amount of resource to use during sensing will not work. Thus, their approach is applicable to sensing with stateless resources (such as time as considered in their work), but not sensing with stateful resources where sensing changes the state (and subsequent behavior) of resources, possibly distorting future sensing outcomes (i.e., the Observer Effect). Zilberstein (1996) foreshadowed this problem, stating:

> However, certain types of sensors, in particular active sensors, require a more complex treatment. The reason is the fact that active sensors may have a significant effect on the state of the environment and thus have additional influence on the planning process… The anytime sensing processes that we describe in this paper are used for information gathering only and have no effect on the state of the environment.

However, in spite of their prescience with respect to the challenge of sensing with stateful resources, to the best of our knowledge no prior work was undertaken by Zilberstein to address this problem. Our work accounts for the changes in resource state (which is part of the state of the environment) caused by sensing in the Observer Effect MDP in order to choose sensing activities to optimize the amount of knowledge refinement produced by possibly distorted sensing outcomes due to the Observer Effect.

6.2. Value of Information Driven Sensing

Second, prior work on sensing activity selection has also focused on optimizing the value of information (VOI) provided by the observations made which is very similar to our goal of selecting activities which optimize the value of knowledge revision. For example, Grass and Zilberstein (1997, 2000) created an approach called value-driven information gathering (VDIG) within a system which gathers information for user decision support from the internet under time and monetary cost limitations. Given a model of the user's decision as an inference diagram, VDIG both determines a set of queries to perform on distributed information sources to gather the information necessary to make a decision, as well as performs those queries to support the user. Specifically, it calculates the VOI for each piece of missing information necessary for the user's decision, where VOI is defined as the increase in the expected utility of the user's decision if it knows the missing information. Once the information with the highest VOI is computed, the agent determines which information sources can provide this information. Finally, the agent calculates the expected likelihood of each source returning the desired information in time (where information sources might be slow and not respond fast enough for the user's decision) as well as the monetary cost of purchasing information from the source. If the VOI of the desired information weighted by the likelihood of successful response is greater

than the monetary cost of using the source, the agent proceeds to gather the information, else it stops and has the user make her decision.

This work is similar to ours in that they determine appropriate sensing activities and execute them to gather information to support a decision process. Also, both consider the benefits versus costs of sensing when making decisions about what sensing activities to perform. However, our work differs in several ways. First, our value from sensing is based on the outcome of sensing itself (knowledge revision), not the reward for making a decision. Thus, our sensing action selection is independent of decision outcomes. Second, we consider information costs due the Observer Effect caused by stateful resource usage, instead of time and monetary costs from using those stateless resources.

Another approach to sensing driven by VOI is BIG, an agent-based approach developed by Lesser *et. al* (2000) which also automates information gathering from the internet for users. Specifically, BIG collects information requested by a user by performing multi-level planning to choose and execute sensing activities. At the top-down level, BIG creates a schedule of sensing activities to gather the desired information by considering important properties of the plan and desired information, including cost (e.g., time and money) and uncertainty. It also considers the value of information, derived from the sources which provide the information (e.g., good sources give better information). On the other hand, at the bottom-up level, BIG is opportunistic and looks for ways to collect additional low-cost information based on its current activities prescribed by the top-down level.

Like VDIG, BIG is similar to our work in that it considers both the value of sensing and costs when making decisions about what sensing to perform. Further, it is more like our approach in that its assigned value of information is based on the quality of information (through the source's history) as opposed to its relationship to decision outcomes as in VDIG.

However, again BIG only considers stateless time and monetary costs and not the Observer Effect. On the other hand, we do note that the planning component of BIG is very interesting, especially how it considers both overall goals through the top-down level, as well as *opportunistic sensing* at the bottom-up level. We are interested in extending our work to account for both *need-driven sensing* based on information needed for current decisions (as we already do), as well as *data-driven sensing* based on information useful for refining knowledge that is expected to be used in the future. Opportunistic sensing could play a key role in our intended data-driven sensing as an agent will need to balance the expected likelihood of needing the information in the future versus its current cost, so finding opportunities for low-cost sensing will increase the usefulness of data-driven sensing.

Finally, in the original Preference Elicitation POMDP work by Boutilier (2002), sensing selection is related to the value of information collected by a sensing activity. Unlike Doshi and Roy's (2008) work considered in the UserRec experiments (c.f., Section 4.2.1) where preferences are single values, Boutilier's POMDP represents the user's preference as a utility function over all possible items (which doesn't change so there are no state transitions), so each state in the POMDP represents each possible utility function. Observations from sensing help the agent refine its belief state which serves as a probability distribution indicating the likelihood that any of the utility functions is the correct one. In their work, the POMDP's reward function is defined as the expected utility of making a decision based on the current beliefs about the user's utility function. The value of information, then, is defined as the change in expected reward based on the change in the agent's belief about the user's utility function after a sensing activity outcome. Thus, like VDIG (Grass and Zilberstein, 1997; 2000), Boutilier's (2002) Preference Elicitation POMDP also uses task rewards (where the agent's task is to make a decision supporting the user) in its calculation of value of information. Our work, on the other hand, only considers the

agent's knowledge in its value calculations and not task reward. However, our reward function is more closely related to Boutilier's as theirs does consider the change in belief about the user's preference, which is what we consider to be the value of information (independent of task reward). Further, as described for Doshi and Roy's (2008) Preference Elicitiation POMDP in Section 4.2.2, our work differs from Boutilier's (2002) in that ours considers the resource's state (e.g., user frustration) which impacts resource behavior and the value of knowledge revision, while the only state they are concerned with is user preference and instead assume constant (albeit stochastic) behavior of the user in responding to sensing.

6.3. Observation Selection Problem

More recently, Krause *et. al* (2008; with Guestrin, 2005; 2007) have studied the problem of selecting sensing activities to perform in order to optimize one or more objective functions given a set of constraints, which they term the Observation Selection Problem (OSP). Specifically, this problem commonly takes the form:

$$\text{Select } \operatorname{argmax}_{A \subseteq V} F(A) \text{ subject to } C(A) \leq B \tag{21}$$

where V is the set of all possible observations, A is the chosen set of observations to perform, $F(\cdot)$ is an objective function evaluating the "goodness" of a set of observations, $C(\cdot)$ is the cost of performing a set of observations, and B is a cost budget. One popular application of this problem is the placement of sensors in an environment, such as to monitor the quality of a water supply (Krause and Guestrin, 2009). Here, one goal might be to maximize the area of coverage of the sensors according to a budget of a fixed number of sensors. Other applications include robotic patrol path planning (Singh *et. al*, 2009), experiment design (Krause *et. al*, 2008), and variance minimization (Krause *et. al*, 2008). To solve the OSP, Krause *et. al* prove that the simple greedy approach of always adding the best observation to the set selected will yield a good approximation to the optimal value (which is difficult to calculate since the problem is NP-

Hard) when the objective function optimized satisfies two key properties: 1) monotonicity, and 2) submodularity (i.e., diminishing returns from adding to the set evaluated).

As we indicated earlier, this problem can also be naturally extended to include multiple objective functions and/or multiple constraints. Often, this entails maximizing the minimum objective function value to offer some guarantee on worse case performance. In the sensor placement example (Krause and Guestrin, 2009), another goal might be to optimize the likelihood of outbreak detection based on an intruder poisoning the water supply. Given that there are multiple possible intrusion scenarios, the likelihood of detecting each can be represented by a different objective function. Thus, maximizing the minimum across these functions guarantees a worst-case bound that any intrusion is at least that likely to be detected. For this case where there are multiple objective functions, Krause *et. al* (2008) prove that no polynomial time approximation algorithm exists (unless P = NP) to solve this problem. However, they also show that by relaxing the cost constraints, an algorithm does exist (called Saturate) which can find a decent approximation to the optimal selection set.

Comparing the Observer Selection Problem to our own work, we see strong similarities to the Observer Effect Tradeoff Problem formulation. Specifically, our value of knowledge refinement function represents an objective function over sensing activities and we choose individual sensing activities to optimize this function (similar to their greedy approach to solving the OSP). However, we roll the costs of information distortion and knowledge corruption from the Observer Effect into the learned objective function because these cannot be observed directly and independently in the environment. Further, we again note that the Observer Effect results in non-monotonic sensing performance, so the objective function we consider does not satisfy the aforementioned properties required by Krause *et. al* (2008) to guarantee the quality of results. Finally, on a more positive note, our work does contribute back to the Observation

Selection Problem by adding in the consideration about resource state and side-effects from sensing based not only on the sensing activity selected, but the state of the resource in the environment. Thus, our solution could be considered a state-dependent Markov decision process solving a variant of the OSP.

6.4. Multiagent Resource Allocation

Finally, the area of multiagent resource allocation is related to our research. Specifically, both characterize resources and build solutions depending on these properties. Chevaleyre *et. al* (2006) summarize the common characterizations of resources in multiagent resource allocation as:

1) continuous vs. discrete: can the a resource *physically* have any quantity, or is it constrained to whole units?
2) divisible or not: can the resource be *allocated* in any quantity, or must it be allocated in whole units?
3) sharable or not: can multiple agents use the same resource at the same time?
4) static or not: is the resource *consumed* (i.e., depleted quantity through usage), *perishable* (i.e., depleted quantity over time), or *static* (unchanged)?
5) single-unit vs. multi-unit: is the resource *homogeneous* or *heterogenous*?
6) resources vs. tasks: task allocation is similar to resource allocation, where tasks are resources with added constraints (e.g., subtasks, task ordering, etc)

Comparing our work to that described by Chevaleyre *et. al* (2006), we note that we have added a new resource characterization: *stateful* vs. *stateless*. Similar to static resources, stateless resources do not change over time. However, the behavior of stateful resources depends on the internal state of the resource which does change over time based on its usage. This differs from both consumed and perishable resources which only consider the change in *quantity* of the

resource through usage and time. The state of a stateful resource, on the other hand, can depend on more than just the resource's quantity. For example, a human user resource in an intelligent user interface application does not have less quantity after interactions, but her frustration level does change based on interruptions (Adamcyzk and Bailey, 2004; Mark et. al, 2008).

Lastly, the use of MDPs to model decisions about resource allocation has recently been studied in the context of environment sustainability for decisions about harvesting natural resources (Ermon, 2010). Here, the state of the MDP is resource quantity which naturally grows over time but is reduced by harvesting the resource. While this MDP does not consider the behavior of resources based on state (nor the state of resources other than quantity) in its model as ours does, its modeling of resource renewal over time could be useful for refining our Observer Effect MDP when the stateful resource in question naturally changes its own state over time, which we intend to investigate in the future.

Chapter 7 Future Work

In this chapter, we describe our plans to advance and improve our research as future work. Specifically, we categorize our future work into four areas: 1) further experiments, 2) solution improvement, 3) real-world application, and 4) research extension.

First, we have identified several types of experiments we intend to conduct to further investigate both the Observer Effect Tradeoff Problem and our Observer Effect MDP. These include:

1) Determine what clues the environment and/or resource behavior provides which hints that the Observer Effect might become overpowering, useful for enabling the agent to predict this problem and avoid it (if possible) (c.f., Section 5.1.2)
2) Investigate why REINFORCE (Williams, 1992) and RMax (Brafman and Tennenholtz, 2002), an RL and PORL algorithm respectively, achieved a higher correlation between sensing and task performance, indicating that the two approaches were able to leverage the better sensing performance through considering the Observer Effect to achieve higher task performance (especially over time for RMax), while the other approaches were not (c.f., Sections 5.1.3 and 5.2.4)
3) Experiment with a wider range of fully and partially observable reinforcement learning algorithms to better understand how learning algorithm characteristics (e.g., model-based vs. model-free, continuous vs. discrete) affect the ability of agents to learn a controller for sensing activity selection within the Observer Effect MDP (c.f., Sections 5.1.4 and 5.2.5)
4) Better understand how the correlation between sensing and task performance depends on different approach and environment characteristics, such as model-based vs. model-free learning, as well as learning duration (c.f., Section 5.2.4)

Second, based on our experiment results (c.f., Chapter 5) and related work (c.f., Chapter 6), we have also determined some avenues for improving our Observer Effect MDP solution. These include:

1) Add the ability to predict (using clues from the environment and/or resources as discussed above) when the Observer Effect will overpower the agent's sensing in order to possibly mitigate this problem before it occurs, enabling the agent to maintain good sensing performance (c.f., Section 5.1.2)
2) Enhance the rate of learning, especially for the PORL algorithms such as BA-POMDP (Ross *et. al*, 2007), in order to minimize decreased performance while learning before the agent is able to exploit a good controller learned from interactions with the environment (c.f., Sections 5.2.3 and 5.2.4)
3) Improve our Observer Effect MDP model to allow agents to reason about self-state changes by the resources themselves (e.g., naturally decaying user frustration over time) similar to the application of MDPs to resource allocation in environment sustainability (Ermon *et. al*, 2010) (c.f., Section 6.4) instead of only considering changes to resource state by the agent
4) Extend the Observer Effect MDP model to allow for joint reasoning by multiple cooperative agents in order to better facilitate joint sensing and problem solving between agents in a multiagent system. Possible extensions include using a decentralized MDP or POMDP (DEC-MDP, DEC-POMDP) (Bernstein *et. al*, 2002) rather than a single-agent MDP/POMDP, as well as using multiagent reinforcement learning (Busoniu *et. al*, 2008)

5) Develop theoretical guarantees such as lower or upper bounds for solution performance based on resource or environment properties (e.g., rates of state change), as well as identify key properties we can exploit to improve performance (e.g., shape of the learned reward function)

6) Relax the Markovian state-history independence assumption in our solution for application to environments where more than just the current state effects the behavior of stateful resources

7) Consider both need- and data-driven motivations for sensing, extending our approach beyond just satisfying current knowledge refinement needs to collect information to revise knowledge it expects to need in the near future if such information can be acquired with lower expected Observer Effect now than later, (c.f., Section 6.2)

Third, we are interested in moving our research out of simulation and into real-world applications of multiagent systems. Specifically, we are currently working to include the Observer Effect MDP in an intelligent user interface agent used to support collaborating researchers in the Biofinity Project (http://biofinity.unl.edu). This agent monitors user activities such as editing a collaborative Wiki, sharing information with collaborators, running *in silico* experiments, or data management. This monitoring provides information valuable for modeling the user, necessary for providing customized support tailored to the individual needs of different users. However, at times the agent will need to interrupt the user to gather unobservable information (e.g., user goals) or clarify uncertain information in its models. To balance the need for information through such interruptions and the possible frustration increase in the user (which affects sensing outcomes (c.f., Section 4.2.1)), we plan to utilize the Observer Effect MDP. The intelligent user interface has already begun implementation and will

be initially deployed in the intelligent wiki portion of the Biofinity Project in November 2010. The addition of the Observer Effect MDP will follow shortly thereafter.

Finally, we are interested in extending aspects of our work beyond the problem of sensing with stateful resources. First, our work is grounded within the more general Limited Resource Sensing Problem (c.f., Section 2.2), so we would like to build a more generalized framework which handles reasoning about both stateful and stateless resources during sensing. Second, stateful resources can also be required during agent reasoning, such as a human user who collaboratively makes decisions with an agent in a mixed-initiative system (e.g., Ferguson and Allen, 2007). Our model of the effects of changes in the state of a resource on its behavior should be extendable into such a scenario. For example, an analog to our Observer Effect MDP might be useful for modeling the state-dependent (i.e., frustration) behavior of the human user the agent is reasoning with to choose interactions which avoid negative side-effects from user frustration (i.e., reduced cognitive ability or willingness to use the system (Klein *et. al*, 2002)). Further, the problem of choosing sensing activities under various costs is very closely related to the active learning problem in machine learning (Settles, 2010) (which is closely related to the Observer Selection Problem (Krause *et. al*, 2008) (c.f., Section 6.3)) where an agent must choose which costly data instances to purchase that will be used for training the learner. If selecting which instances to purchase effects future instances and learning, the instance provider can be seen as a stateful resource (e.g., an intelligent oracle that models the learner and offers suggestions of which instances to purchase or changes the cost to optimize its benefit) and aspects of our work could apply to this problem as well.

Chapter 8 Conclusion

In this chapter, we conclude the thesis. We begin by summarizing what we have presented in the previous chapters, followed by a highlight of the key contributions of the work.

8.1. Summary

In Chapter 1, we provided a general introduction to our research. We began by describing a common problem in current research areas of computer science in general and applications of artificial intelligence in specific: **limited resources** which constrain the intelligent agent's activities, including reasoning through bounded rationality. Next, we briefly introduced the main focus of this thesis: how the use of stateful resources during sensing also affects the bounded rationality of agents. Finally, we gave a brief overview of our Observer Effect MDP solution and highlighted the key contributions of this thesis (summarized again in the following section).

In Chapter 2, we dove deeper into the problem we address with our research: the **Observer Effect Tradeoff Problem** (OETP), a subproblem of the **Limited Resource Sensing Problem** (LRSP). First, we provided important background on bounded rationality to set the context of our work. Next, we described the LRSP which occurs when an agent must decide how to balance the need for information and knowledge revision with the costs incurred from using limited resources during sensing. Then, we introduced an important subproblem of the LRSP: the OETP which occurs when the resources used by agents during sensing are *stateful*. Specifically, we define stateful resources as those resources whose behavior depends on some notion of internal state which changes with resource usage. In the context of agent sensing, this is important because using such resources during sensing can change the state of the resources, altering their behavior and resulting in a different sensing outcome than would have occurred if the resource were in a different state. This produces a phenomenon we call the Observer Effect

(after the similar phenomenon in the physical sciences) where the *act of sensing can and will distort its own outcome (and potentially future outcomes)*. From the perspective of sensing and bounded rationality, this is an important challenge to sensing because it creates a tradeoff (the OETP) between 1) the need for sensing to refine knowledge used to guide agent decision making, and 2) the need to avoid knowledge corruption produced by distorted sensing outcomes from the Observer Effect. We concluded the chapter by formalizing the OETP in mathematical notation.

In Chapter 3, we introduced our solution to the OETP: the **Observer Effect MDP**. We began by describing how we use active perception as a vehicle for making decisions about what sensing activities to perform, a step we consider necessary for solving the LRSP and its subproblems including the OETP. Next, we describe how we model the decision process of selecting sensing activities which require the use of stateful resources as a Markov decision process (MDP), which we call the Observer Effect MDP. In this MDP, we account for the state of resources (along with the state of knowledge) in a *sensing state* and the agent makes decisions about what sensing activities to perform based on optimizing the *value of knowledge refinement* as the reward to the MDP based on the current sensing state and the agent's sensing activity choice. Here, the value of knowledge refinement captures not only expected *benefits* to knowledge refinement from sensing, but also expected *distortions* due to the current state of the resource. Because such an MDP model is difficult to provide *a priori* to the agent, we describe how an agent can use *reinforcement learning* (RL) to learn both a controller to choose actions, as well as a model of the value of knowledge refinement reward function based on its experience interacting with the stateful resources through sensing.

In Chapter 4, we described the experimental setup used to explore and validate our Observer Effect MDP solution approach to the OETP. Specifically, we considered two simulation environments, each with unique characteristics. First, we used a *fully observable* robotic mining simulation called **MineralMiner** where an agent must use a microscope to determine the mineral contents of a mine which it must collected to complete tasks. The accuracy of the microscope depends on its current energy level, which is both reduced through microscope use (depending on the type of test performed) and recharges over time when not used. Second, we used a *partially observable* user preference elicitation simulation called **UserRec**, based on prior work by Doshi and Roy (2008), where an intelligent user interface agent must determine a user's preference through interruptions which prompt the user for information. These interruptions frustrate the user, affecting both the timeliness and accuracy of her responses. Thus, in both environments, the state of the resource (energy for the microscope and frustration for the user) influence the accuracy of the sensing outcomes, producing an Observer Effect during sensing and requiring the agent to solve the OETP to make proper decisions about what sensing activities to perform. For both simulations, we described how we model the Observer Effect MDP, as well as the reinforcement learning algorithms used to control agent sensing.

In Chapter 5, we presented the results of the experimental setup described in Section 4. Specifically, we tested two hypotheses in both environments. First, we tested the **Observer Effect Hypothesis**, which states that approaches which consider the state of stateful resources and the Observer Effect during sensing (such as our Observer Effect MDP) will outperform other approaches which do not consider resource state, even though the side-effects of sensing based on resource state is difficult to model and selecting wrong actions could worsen the effect. Second, we tested the **Performance Hypothesis** which states that increased sensing performance should yield higher task performance. After demonstrating that the Observer

Effect exists in both environments as intended, we confirmed the Observer Effect Hypothesis (with statistical significance in MineralMiner), noting that the RL-based approaches tested which consider the Observer Effect achieved higher sensing performance (in the form of average value of knowledge refinement per sensing activity and thus less distortion) than ignoring this effect. Thus, considering resource state and the Observer Effect when using stateful resources is important and our Observer Effect MDP solution approach (with reinforcement learning) is capable of doing so. However, we were unable to confirm the Performance Hypothesis as the top RL algorithms achieved the highest task performance, but there was generally little to no correlation between sensing and task performance.

In Chapter 6, we described the most relevant related work from the artificial intelligence and multiagent systems literature. Specifically, we described four areas of research. First, prior work in anytime sensing has considered the use of anytime algorithms as metareasoning control of sensing activities. However, this work only applies to stateless resources with monotonic sensing performance with resource usage, an assumption violated by the Observer Effect in stateful resources. Second, prior work in value of information (VOI) driven sensing has considered how to choose sensing actions which maximize the VOI produced by sensing, which is similar to our selection based on the value of knowledge refinement. However, most of these approaches rely on decision or task rewards to calculate the VOI, while our value is independent of the outcomes of decisions, making our approach more modular. Third, prior work in the Observation Selection Problem has also considered sensing activity selection to optimize an objective function, usually given constraints, similar to our formalization of the OETP. However, their work provides algorithms with theoretical guarantees about the quality of the selection which assumes properties such as monotonicity and submodularity again not present (at least monotonicity) due to the Observer Effect. Finally, prior work in multiagent resource allocation

has defined important characteristics of resources used by agents, to which we add the stateful property. Further, very recent work has looked at the use of MDPs for modeling resource allocation in environment sustainability which considers the renewal of resource quantity over time which could be useful for improving our Observer Effect MDP model when resources also renew their states over time.

Finally, in Chapter 7, we discussed the future work we intend to explore after the conclusion of this thesis. Specifically, we are interested in four areas of future work: 1) further experimentation to discover key environment/resource properties related to the Observer Effect or further explain results discovered in this thesis, 2) further improvement to the Observer Effect MDP, including the aforementioned state renewal of resources over time, as well as considering both need- and data-driven motivations for sensing, 3) the application of our work to real-world multiagent deployments, including an intelligent user interface for supporting collaborating researchers within the Biofinity Project (http://biofinity.unl.edu), and finally 4) extending our work both to solve other problems, including the general LRSP, as well as into other areas of research, including agent reasoning in mixed-initiative systems and active machine learning.

8.2. Contributions

We conclude this thesis by re-emphasizing its key contributions. Specifically, we have provided:

1. An extension of bounded rationality as studied in artificial intelligence to the sensing activities of the agent through the Limited Resource Sensing Problem,
2. The formalization of the Observer Effect in agent sensing with stateful resources and its associated tradeoff with respect to knowledge refinement,

3. A decision theoretic solution called the Observer Effect MDP for modeling the effects of stateful resource usage during sensing and solving the OETP,
4. Simulation environments mimicking real-world scenarios and applications for studying the Observer Effect and solution approaches, and
5. A Java library offering various general artificial intelligence techniques which can be reused for other AI projects.

References

Adamczyk, P.D. and Bailey, B.P. 2004. If not now, when? The effects of interruption at different moments within task execution. *Proc. of CHI'04*. Vienna, Austria. April 24-29. 271-278.

Adomavicius, G. and Tuzhulin, A. 2005. Toward the next generation of recommender systems: A survey of state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*. 17(6). 734-749.

Akyildiz, I.F., Pompili, D., and Melodia, T. 2005. Underwater acoustic sensor networks: research challenges. *Ad hoc networks*. 3(3), 257-279.

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. 2002. Wireless sensor networks: a survey. *Computer Networks*. 38. 393-422.

Arisha, K., Youssef, M., and Younis, M. 2002. Energy-aware TDMA-based MAC for sensor networks. *System-level power optimization for wireless multimedia communication*. ed. Karri, R. and Goodman, D. Kluwer Academic Publishers: Norwell, MA. 21-40.

Bajcsy, R. 1988. Active perception. *Proceedings of the IEEE*. 76(8). 996-1005.

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.

Biofinity Project, The. 2010. Available online at http://biofinity.unl.edu

Boddy, M. and Dean, T., 1989. Solving time-dependent problems. *Proc. of IJCAI'89*. 979-984.

Boutilier, C. 2002. A POMDP formulation of preference elicitation problems. *Proc. of AAAI'02*, 239-246.

Brafman, R.I. and Tennenholtz, M. 2002. R-max – A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research.* 3. 213-231.

Bull, S. and Greer, J. 2000. Peer help for problem-based learning. *Proceedings of the 8th International Conference on Computers in Education (ICCE/ICAI '00).* 2. 1007-1015.

Burnett, G.E. and Porter, J.M. 2001. Ubiquitous computing within cars: designing controls for non-visual use. *International Journal of Human-Computer Studies.* 55(4). 521-531.

Busoniu, L., Babuska, R, and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*. 32(2). 156-172.

Casper, J. and 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.

Chalupsky, H. et al. 2001. Electric Elves: Applying agent technology to support human organizations. *Proc. of IAAI '01.* Seattle, WA. Aug. 7-9, 2001. 51-58.

Chevaleyre, Y. et. al. 2006. Issues in multiagent resource allocation. *Informatica*. 30. 3-31.

Conlisk, J. 1996. Why bounded rationality? *Journal of Economic Literature*. 34. June 1996. 669-700.

desJardins, M.E. et. al. 1999. A survey of research in distributed, continual planning. *AI Magazine*. 20(4). 13-22.

Doshi, F. and Roy, N. 2008. The permutable POMDP: fast solutions to POMDPs for preference elicitation. *Proc. of AAMAS'08*, 493-500.

Dowling, J. *et. al* 2005. Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing, *IEEE Transactions on SMC*, *Part A*. 35(3). 360-372.

Ermon, S. et. al. 2010. Playing games against nature: optimal policies for renewable resource allocation. *Proc. of UAI'10*.

Ferguson, G. and Allen, J. 2007. Mixed-initiative systems for collaborative problem solving. *AI Magazine*. 28(2). 23-32.

Fikes, R.E. and Nilsson, N.J. 1971. Application of theorem proving to problem solving. *Artificial Intelligence*. 2(3-4). 189-208.

Floreano, D. and Mondada, F. 1994. Active perception, navigation, homing, and grasping: An autonomous perspective. *Proc. of the Perception to Action Conference*. 122-133.

Fowler, H.J. and Leland, W.E. 1991. Local area network traffic characteristics, with implications for broadband network congestion management. *IEEE Journal on Selected Areas of Comm.*, 9(7). 1139-1149.

Ganek, A.G., and Corbi, T.A., 2003. The dawning of the autonomic computing era. *IBM Systems Journal*. 42(1). 5-18.

Gers, F.A., Schmidhuber, J., and Cummins, J. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation*. 12(10). 2451-2471.

Gigerenzer, G. and Goldstein, D.G. 1996. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*. 103(4). 650-669.

Gigerenzer, G. and Todd, P.M. 1999. Fast and frugal heuristics – The adaptive toolbox. *Simple heuristics that make us smart*. Oxford University Press: New York. 3-34.

Grass, J. and Zilberstein, S. 1997. Value-driven information gathering. *Proc. of AAAI Workshop on Building Resource-Bounded Reasoning Systems*.

Grass, J. and Zilberstein, S. 2000. A value-driven system for autonomous information gathering. *Journal of Intelligent Information Systems*. 14. 5-27.

Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*. 9. 1735-1780.

Horvitz, E. 1987. Reasoning about beliefs and actions under computational resource constraints. *Proc. of UAI'87*. 429-444.

Jaeger, H. 2002. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. *GMD Report 159*. German National Research Center for Information Technology.

Josang, A. 2001. A logic for uncertain probabilities, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*. *9*. 279-311.

Kaelbling, L.P., Littman, M.L., and Cassandra, A.R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*. 101. 99-134.

Kaelbling, L.P., Littman, M.L., and Moore, W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*. 4. 237-285.

Kephart, J.O. and Chess, D.M. 2003. The vision of autonomic computing. *IEEE Computer*. 36(1). 41-50.

Khandaker, N., Soh, L.-K., Miller, L.D., Eck, A., & Jiang, H. (to appear). Lessons Learned from Deploying I-MINDS and ClassroomWiki – Collaborative Learning and Writing Applications. to appear in *IEEE Transactions on Learning Technologies* (Special Issue on Intelligent Support Systems for CSCL).

Kidd, C.D. et. al. 1999. The aware home: A living laboratory for ubiquitous computing research. *Cooperative Buildings: Integrating Information, Organizations, and Architecture*. Springer. 191-198.

Klein, J., Moon, Y., and Picard, R.W. 2002. This computer responds to user frustration: theory, design, and results. *Interacting with Computers*. 14. 119-140.

Krause, A. and Guestrin, C. 2005. Optimal nonmyopic value of information in graphical models – efficient algorithms and theoretical limits. *Proc. of IJCAI'05*. 1339-1345.

Krause, A. and Guestrin, C. 2007. Near-optimal observation selection using submodular functions. *Proc. of AAAI'07*.

Krause, A. et. al. 2008. Robust submodular observation selection. *Journal of Machine Learning Research*. 9. 2761-2801.

Krause, A. and Guestrin, C. 2009. Optimizing sensing: From water to the web. *IEEE Computer.* 42(8). 38-45.

Landeldt, B., Sookavantana, P., and Seneviratne, A. 2000. The case for a hybrid passive/active network monitoring scheme in the wireless internet. *Proc. ICON'00*. 139-143.

Lee, E.A. 2008. Cyber-physical systems: Design challenges. *Technical Report UCB/EECS-2008-8*. University of California at Berkeley. January 23.

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., and Klocke, U. 2008. The cost of interrupted work: more speed and stress. *Proc. of CHI'08*. 107-110.

Monostori, L., Vancza, J., and Kumara, S.R.T. 2006. Agent-based systems for manufacturing. *CIRP Annals – Manufacturing Technology*. 55(2). 697-720.

Myers, K.L. et al. 2007. An intelligent personal assistant for task and time management. *AI Magazine*. 28(2). 47-61.

North, M.J., Collier, N.T., and Vos, J.R. 2006. Experiences creating three implementations of the Repast agent modeling toolkit, *ACM Transactions on Modeling & Computer Simulation.* 16. 1-25.

Padhy, P., Dash, R.K., Martinez, K., and Jennings, N.R. 2006. A utility-based sensing and communication model for a glacial sensor network. *Proc AAMAS'06*. Hakodate, Japan, May 8-12. 1353-1360.

Pineau, J., Gordon, G., and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. *Proc. of IJCAI'03*. 1025-1032.

Pollack, M.E. and Ringuette, M. 1990. Introducing the tileworld: experimentally evaluating agent architectures. *Proc. of AAAI'90*. 183-189.

Raja, A. and Lesser, V. 2007. A framework for meta-level control in multi-agent systems. *JAAMAS*. *15*. 147–196.

Ross, S. and Chaib-draa, B. 2007. Aems: An anytime online search algorithm for approximate policy refinement in large POMDPs. *Proc. of IJCAI '*07. 2592-2598.

Ross, S., Chaib-draa, B., and Pineau, J. 2007. Bayes-adaptive POMDPs. *Proc. of NIPS'07*.

Ross, S., Pineau, J., Paquet, S., and Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*. 32. 663-704.

Rubinstein, A. 1998. *Modeling Bounded Rationality*. MIT Press: Cambridge, MA.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. Learning internal representations by error propogation. *Parallel distributed processing: explorations in the microstructure of cognitions*. MIT Press:Cambridge, MA. 1. 318-362.

Russell, S. 1995. Rationality and intelligence. *Proc. IJCAI '95*. Montreal, Quebec, Canada. 950-957.

Saha, D. and Mukherjee, A. 2003. Pervasive computing: A paradigm for the 21st century. *Computer*. 36(3). 25-31.

Settles, B. 2010. Active learning literature survey. *Computer Sciences Technical Report 1648*. University of Madison, Wisconsin. Updated Jan. 26, 2010.

Shah, R.C. and Rabaey, J.M. 2002. Energy aware routing for low energy ad hoc sensor networks. *Proc. of WCNC'02*. March 17-21. 350-355.

Simon, H.A. 1955. A behavioral model of rational choice. *Quarterly Journal of Economics*. 69(1). 99-118.

Simon, H.A. 1956. Rational choice and the structure of the environment. *Psychological Review*. 63(2). 129-138.

Simon, H.A. 1997. *Models of bounded rationality*. 3. MIT Press: Cambridge, MA.

Singh, A. et. al. 2009. Efficient information sensing using multiple robots. *Journal of Artificial Intelligence Research*. 34. 707-755.

Smith, T. and Simmons, R. 2004. Heuristic search value iteration for POMDPs. *Proc. UAI'04*. 520–527.

So, R. and Sonenberg, L. 2009. The roles of active perception in intelligent agent systems. *PRIMA'05, LNAI 4078*. ed. Lukose, D. and Shi, Z. Springer-Verlang: Berlin, Germany. 139-152.

Sun, M., Wang, Q., and Sha, L. 2007. Building reliable MD PnP systems. *Proc. of HCMDSS-MDPnP'07*. Boston, MA. July 25-27. 104-110.

Sutton, R.S. and Barto, A.G. 1998. Reinforcement learning: an introduction. MIT Press:Cambridge, MA.

Watkins, C.J. 1989. Learning from delayed rewards. Ph.D thesis, Cambridge University, Cambridge, England.

Weiss, G. 1999. *Multiagent systems: a modern approach to distributed artificial intelligence*. The MIT Press: Cambridge, MA.

Werbos, P.J. 1990. Backpropogation through time: what it does and how to do it. *Proc. of the IEEE*. 78(10). 1550-1560.

Weyns, D., Helleboogh, A., and Holvoet, T. 2005. The packet-world: a test bed for investigating situated multi-agent systems. *Software Agent-Based Applications, Platforms, and Development Kits*. 383-408.

Weyns, D., Steegmans, E., and Holvoet, T. 2004. Towards active perception in situated multi-agent systems. *Applied Artificial Intelligence*. 18. 867-883.

Wierstra, D., Foerster, A., Peters, J., and Schmidhuber, J. 2007. Solving deep memory POMDPs with recurrent policy gradients. *Proc. of ICANN'07*. 697-706.

Williams, R.J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8, 229-256.

Yorke-Smith, N., Saddati, S., Meyers, K.L., and Morley, D.N. 2009. Like an intuitive and courteous butler: a proactive personal agent for task management. *Proc. of AAMAS'09*. Budapest, Hungary. May 13-15. 337-344.

Zilberstein, S. 1996. Resource-bounded sensing and planning in autonomous systems. *Autonomous Robots*. 3. 31-48.

Zilberstein, S. 2008. Metareasoning and bounded rationality. *Proc. of AAAI Workshop on Metareasoning: Thinking about Thinking*.

Zilberstein, S. and Russell, S.J. 1993. Anytime sensing, planning, and action: A practical model for robot control. *Proc. of IJCAI'93*. 1402-1407.

Appendix A Additional Results Figures

In this appendix, we provide all of the time series results for both the MineralMiner and UserRec experiments discussed in Chapter 5. We begin by providing the results for MineralMiner, followed by UserRec.

A.1. MineralMiner Time Series Results

A.1.1. Sensing Performance

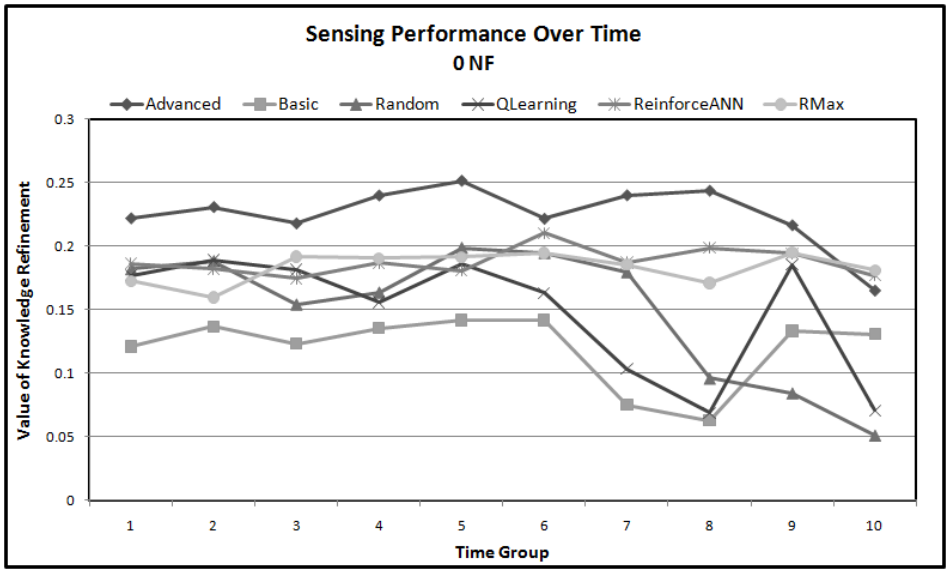

Figure A.1 Sensing Performance over Time in MineralMiner (0.0 NF)

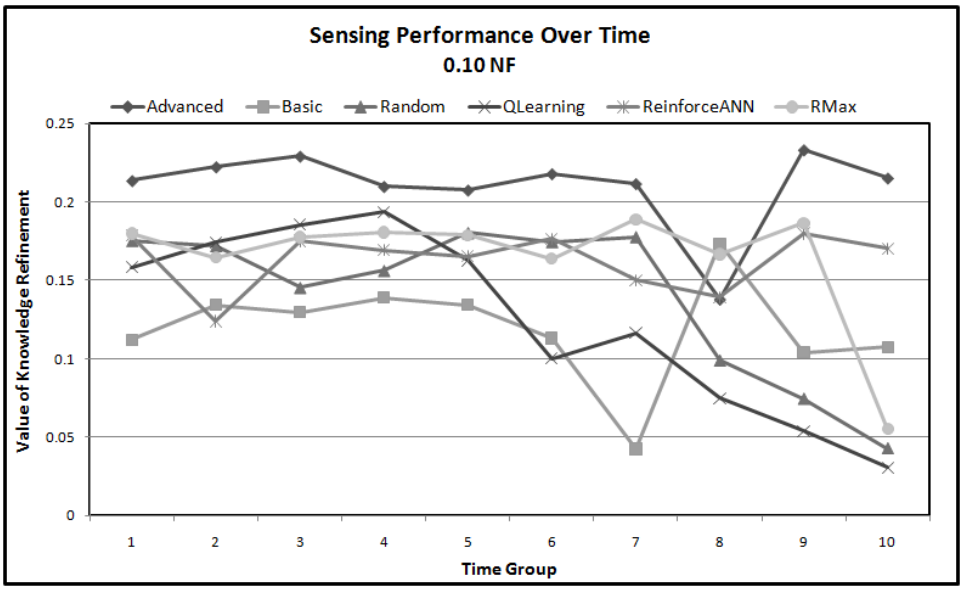

Figure A.2 Sensing Performance over Time in MineralMiner (0.1 NF)

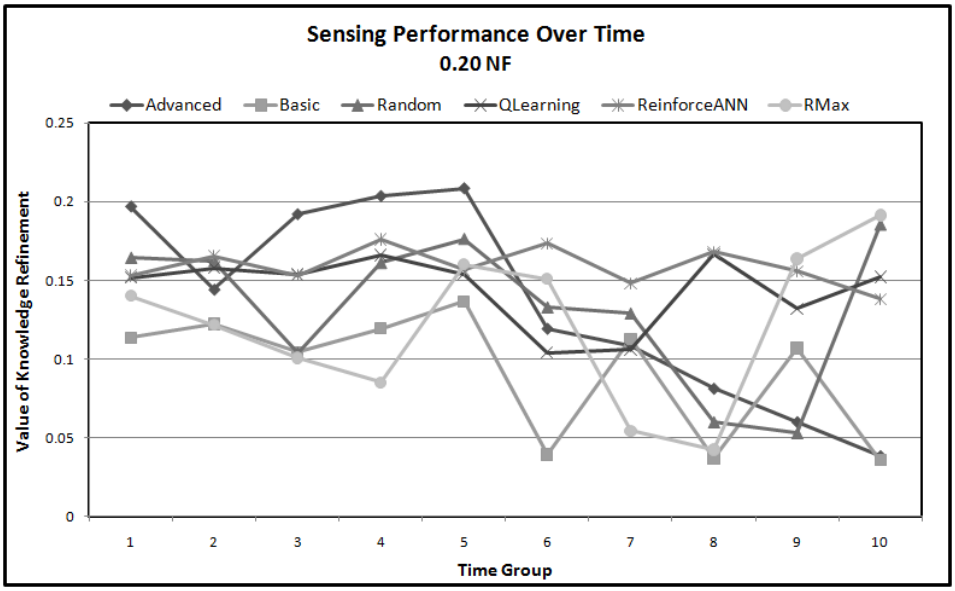

Figure A.3 Sensing Performance over Time in MineralMiner (0.2 NF)

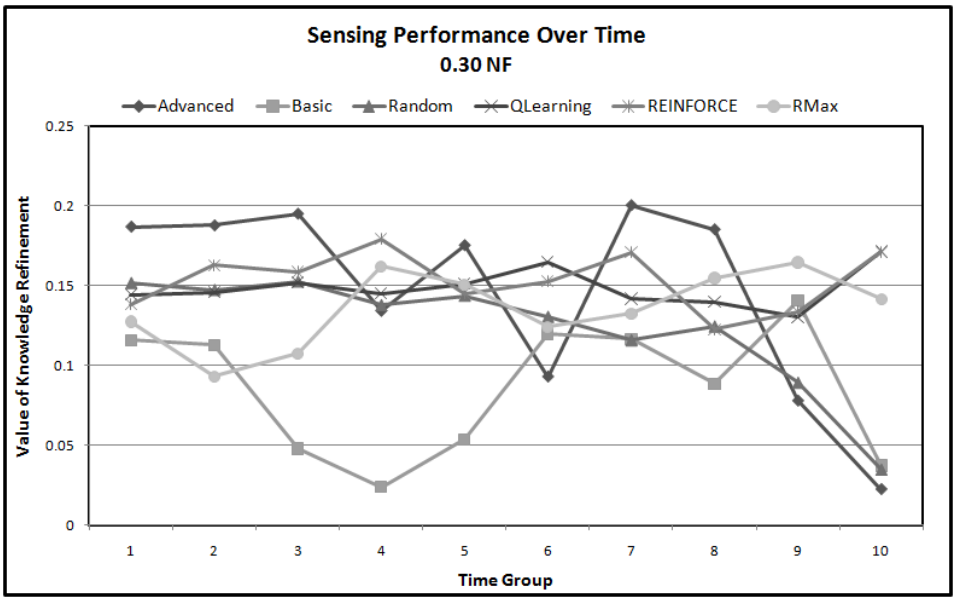

Figure A.4 Sensing Performance over Time in MineralMiner (0.3 NF)

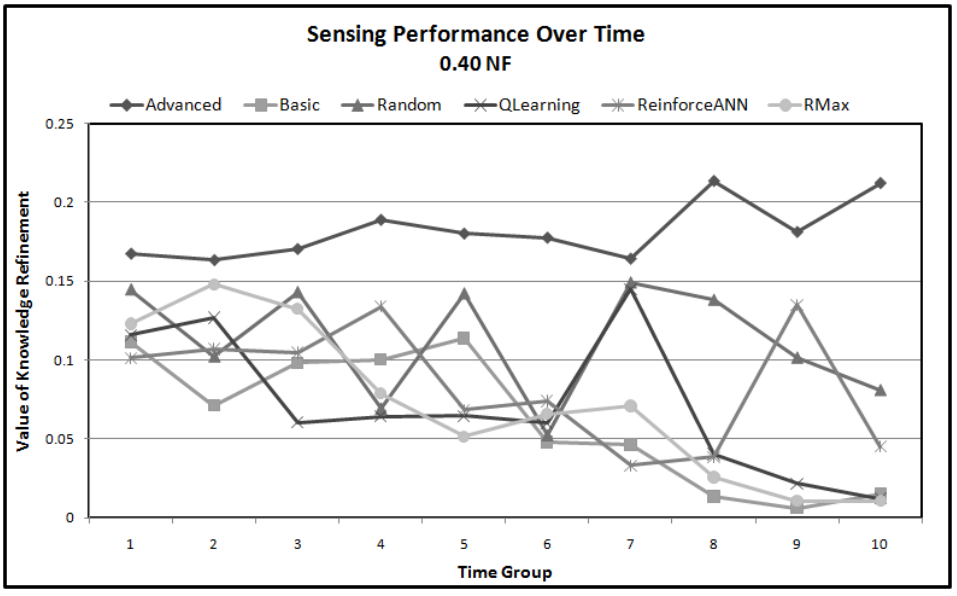

Figure A.5 Sensing Performance over Time in MineralMiner (0.4 NF)

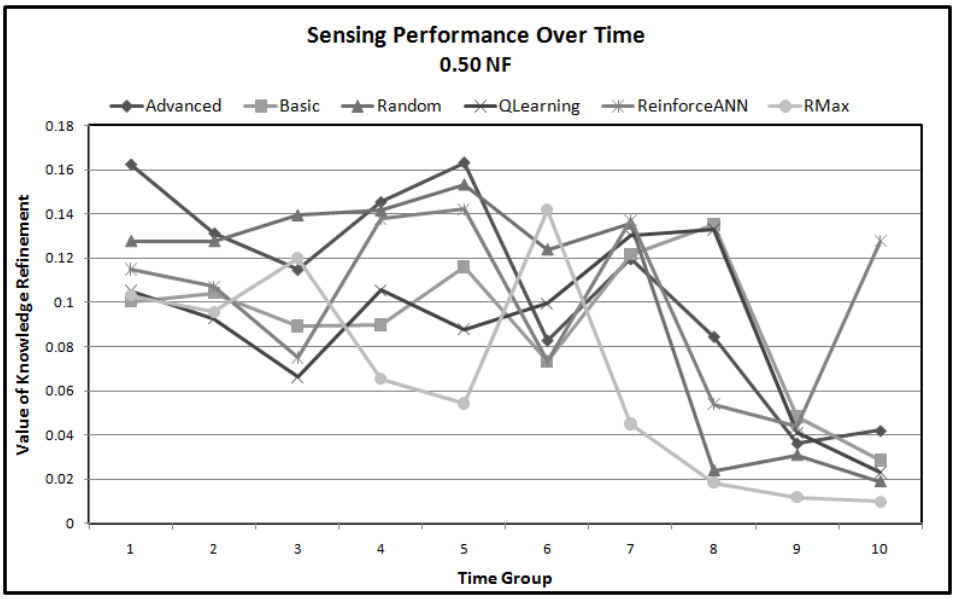

Figure A.6 Sensing Performance over Time in MineralMiner (0.5 NF)

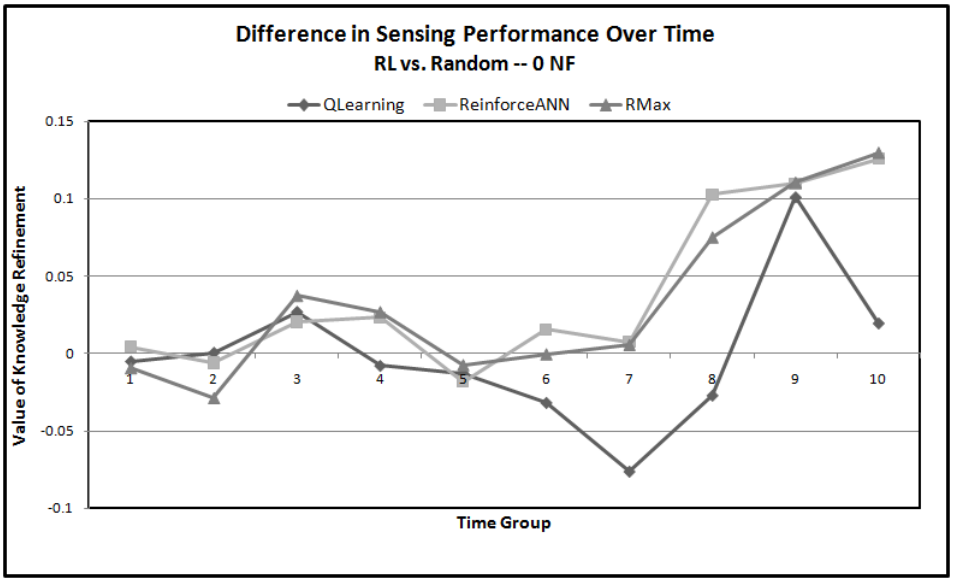

Figure A.7 Sensing Performance of RL vs. Random over Time in MineralMiner (0.0 NF)

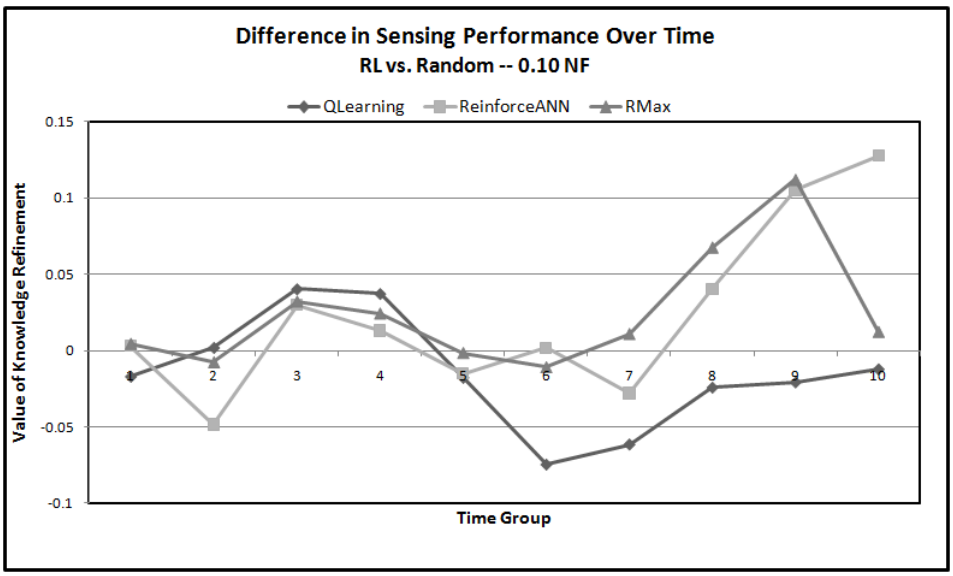

Figure A.8 Sensing Performance of RL vs. Random over Time in MineralMiner (0.1 NF)

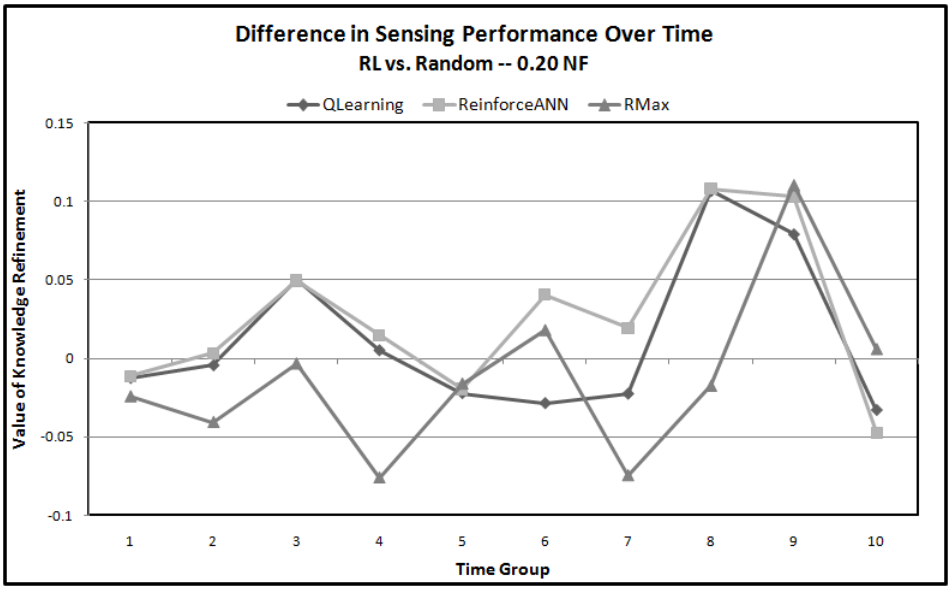

Figure A.9 Sensing Performance of RL vs. Random over Time in MineralMiner (0.2 NF)

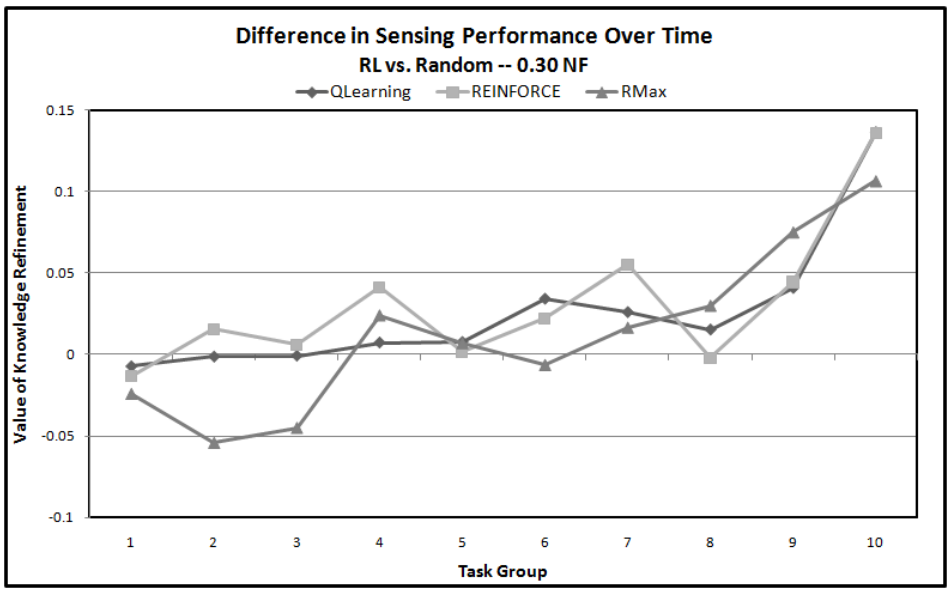

Figure A.10 Sensing Performance of RL vs. Random over Time in MineralMiner (0.3 NF)

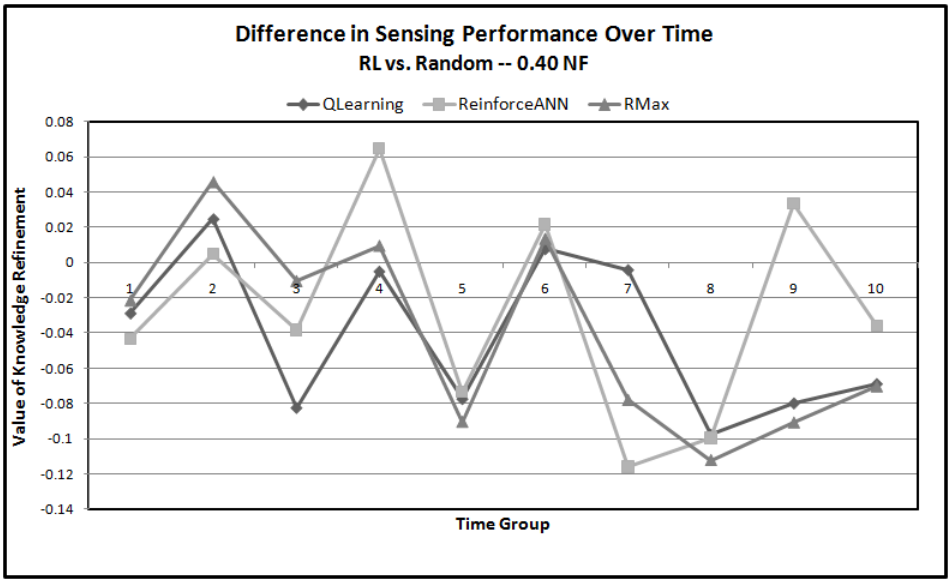

Figure A.11 Sensing Performance of RL vs. Random over Time in MineralMiner (0.4 NF)

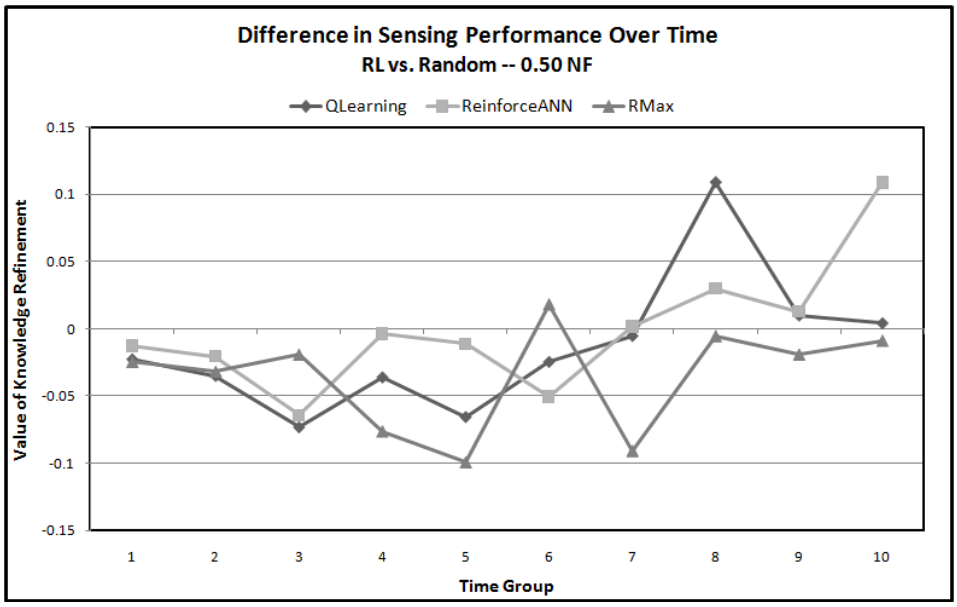

Figure A.12 Sensing Performance of RL vs. Random over Time in MineralMiner (0.5 NF)

A.1.2. Task Performance

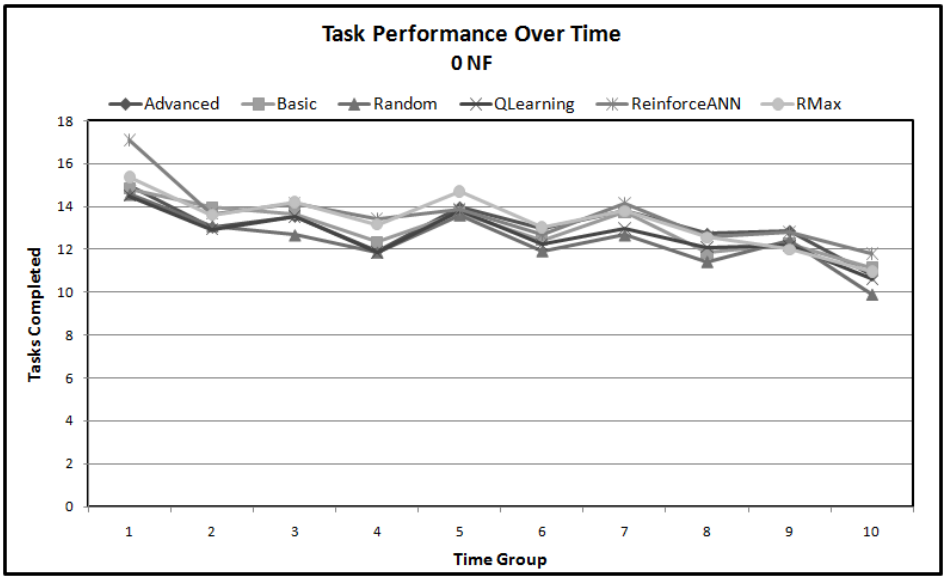

Figure A.13 Task Performance over Time in MineralMiner (0.0 NF)

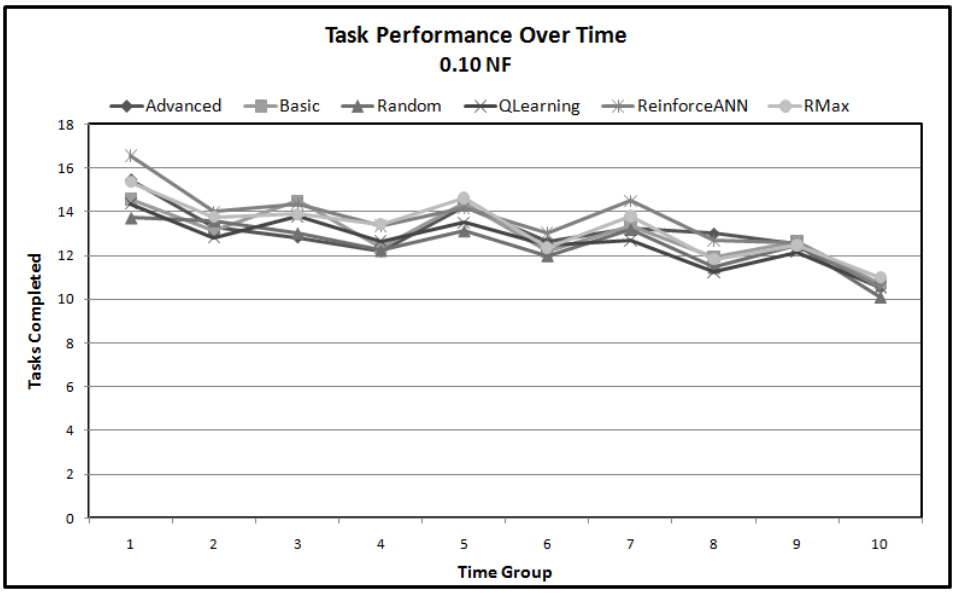

Figure A.14 Task Performance over Time in MineralMiner (0.1 NF)

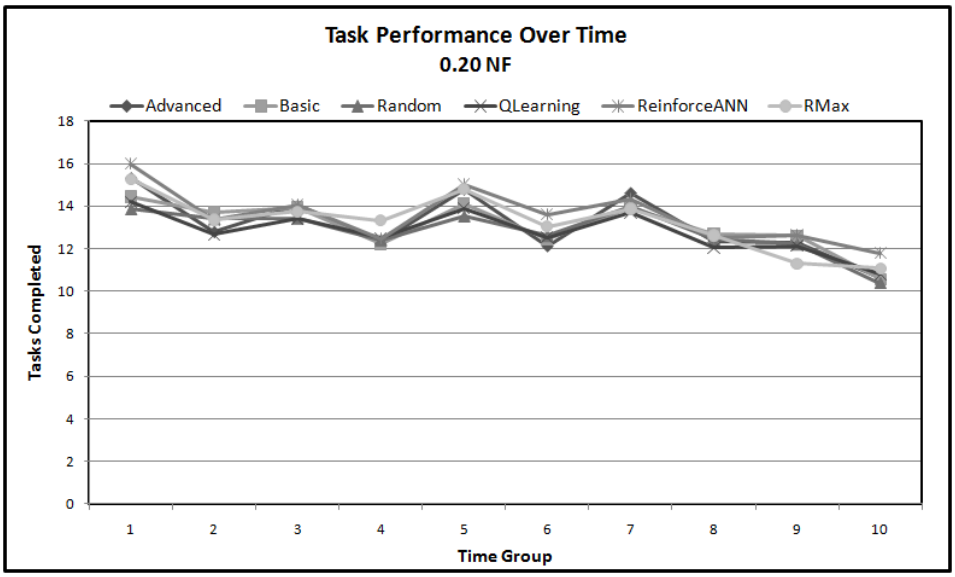

Figure A.15 Task Performance over Time in MineralMiner (0.2 NF)

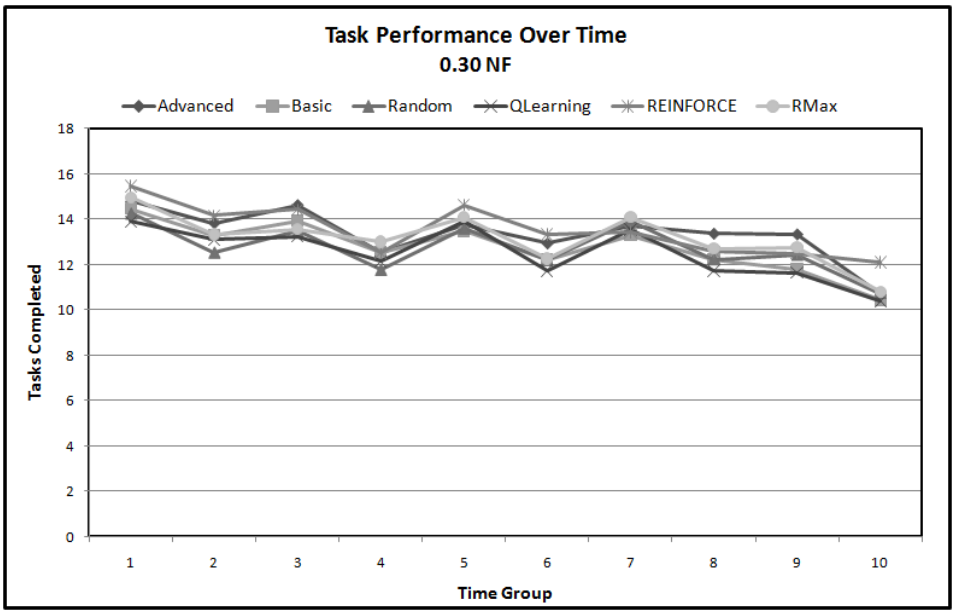

Figure A.16 Task Performance over Time in MineralMiner (0.3 NF)

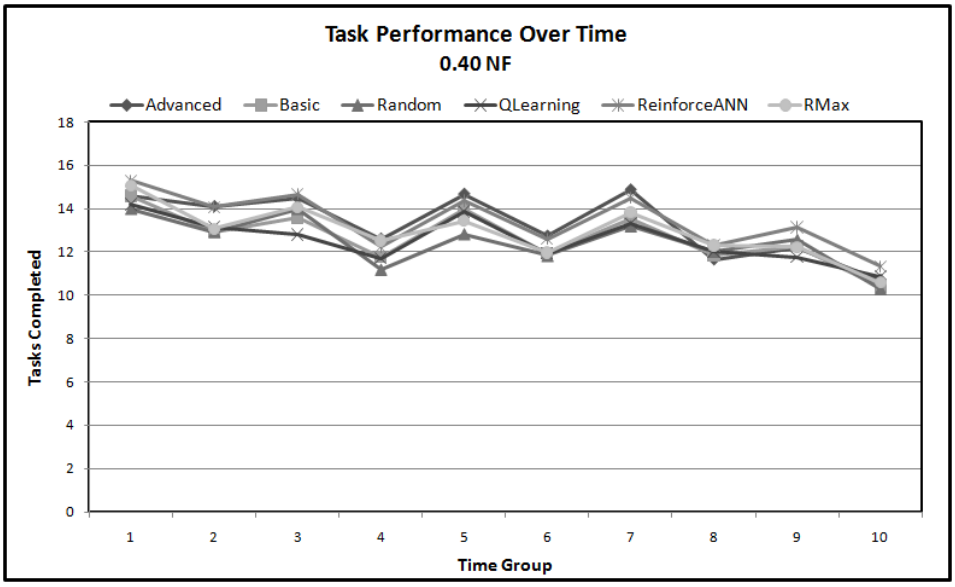

Figure A.17 Task Performance over Time in MineralMiner (0.4 NF)

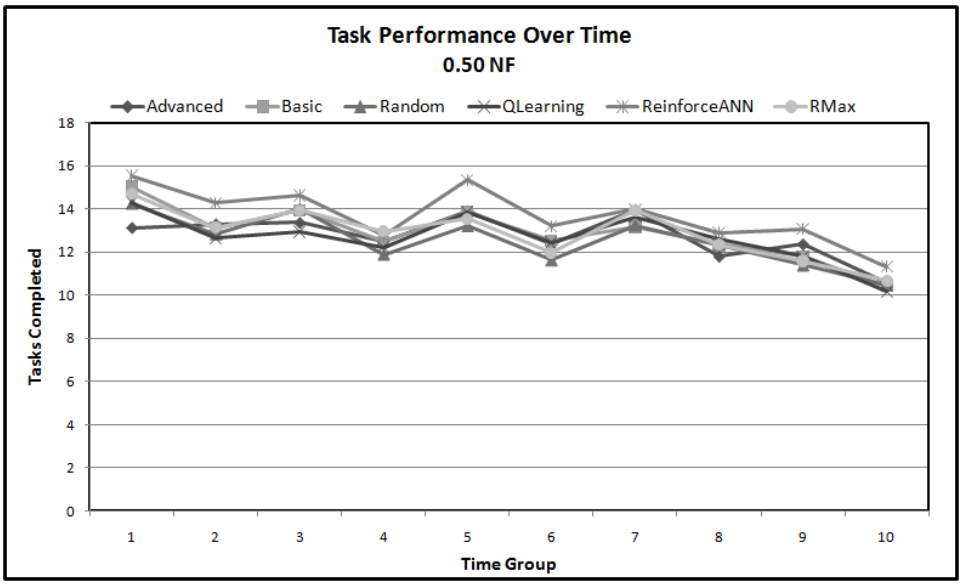

Figure A.18 Task Performance over Time in MineralMiner (0.5 NF)

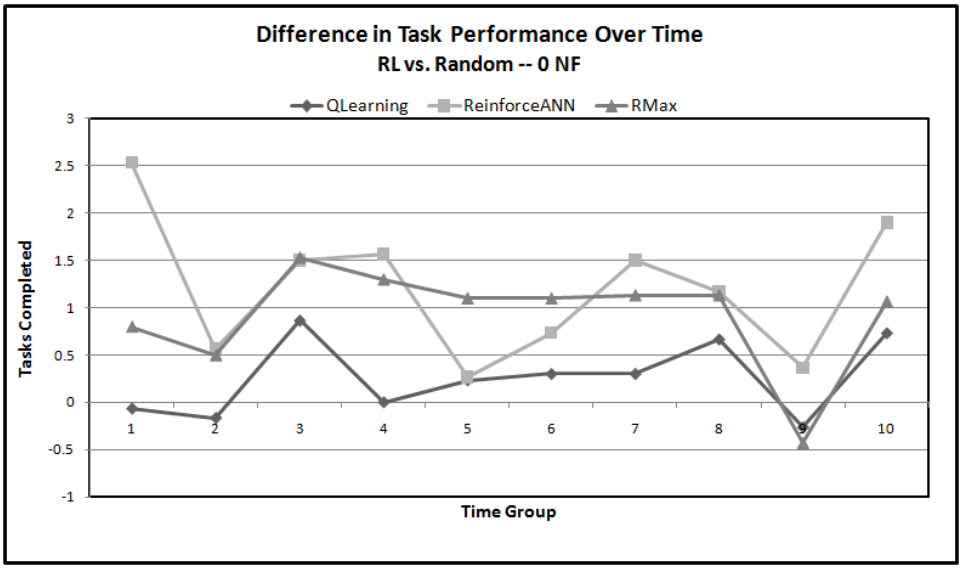

Figure A.19 Task Performance of RL vs. Random over Time in MineralMiner (0.0 NF)

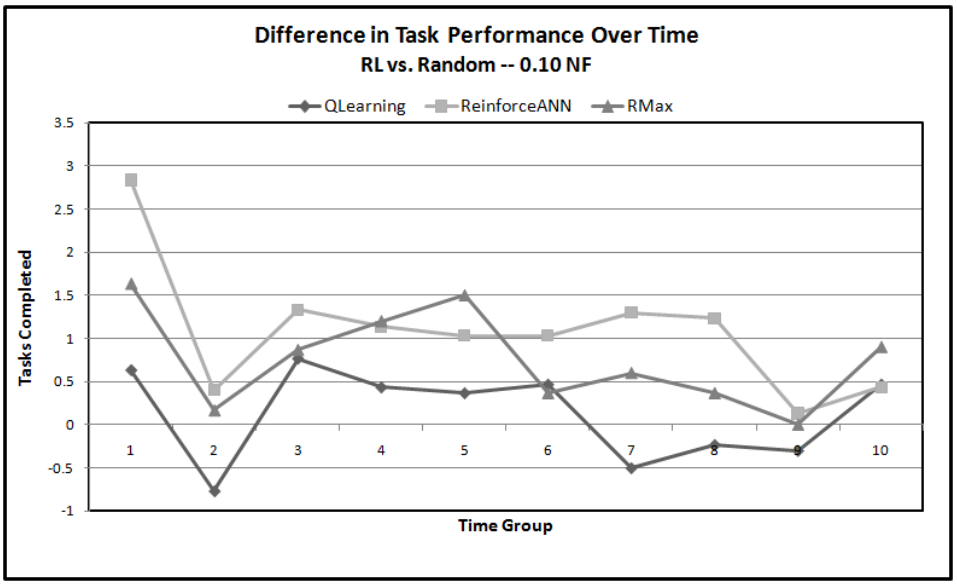

Figure A.20 Task Performance of RL vs. Random over Time in MineralMiner (0.1 NF)

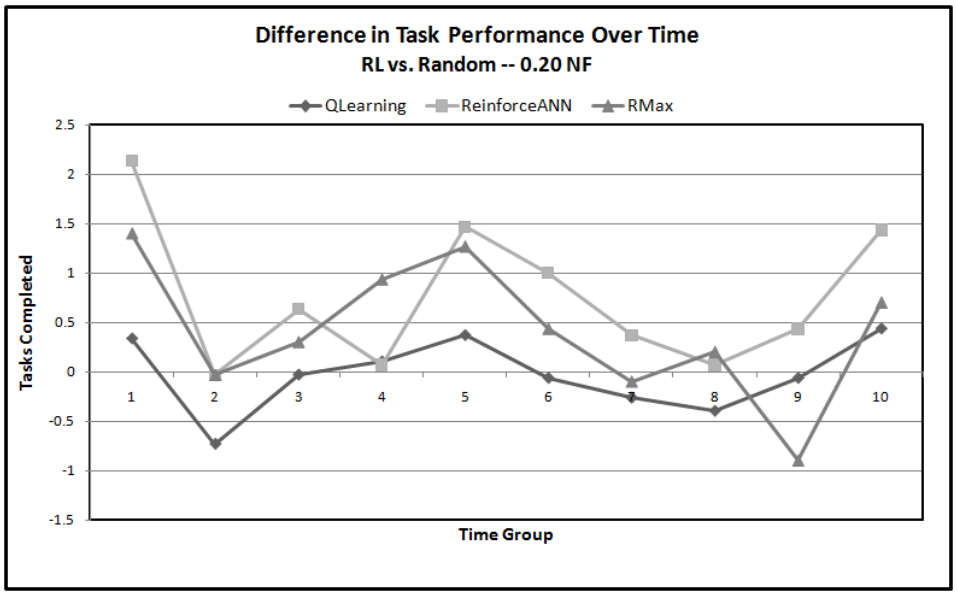

Figure A.21 Task Performance of RL vs. Random over Time in MineralMiner (0.2 NF)

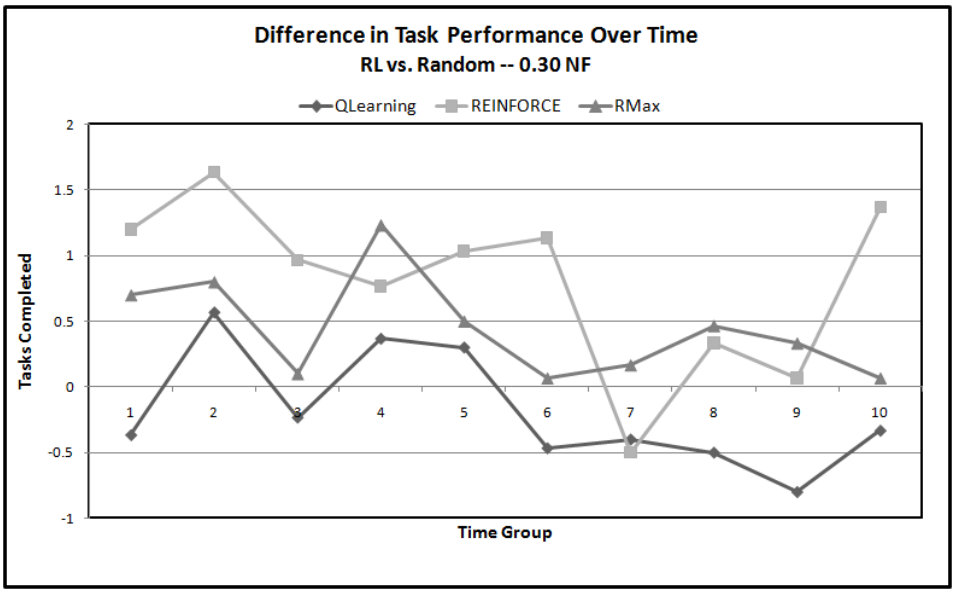

Figure A.22 Task Performance of RL vs. Random over Time in MineralMiner (0.3 NF)

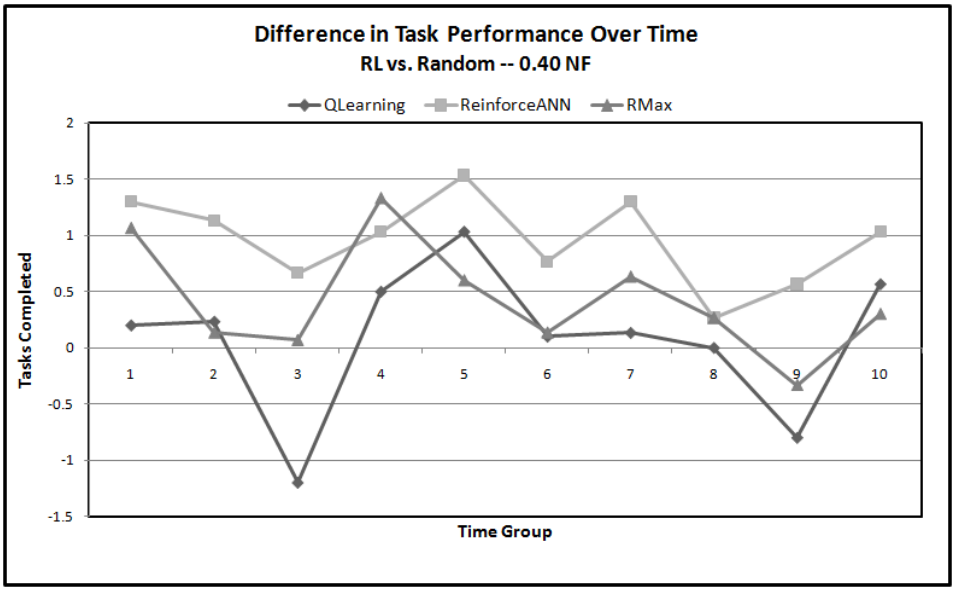

Figure A.23 Task Performance of RL vs. Random over Time in MineralMiner (0.4 NF)

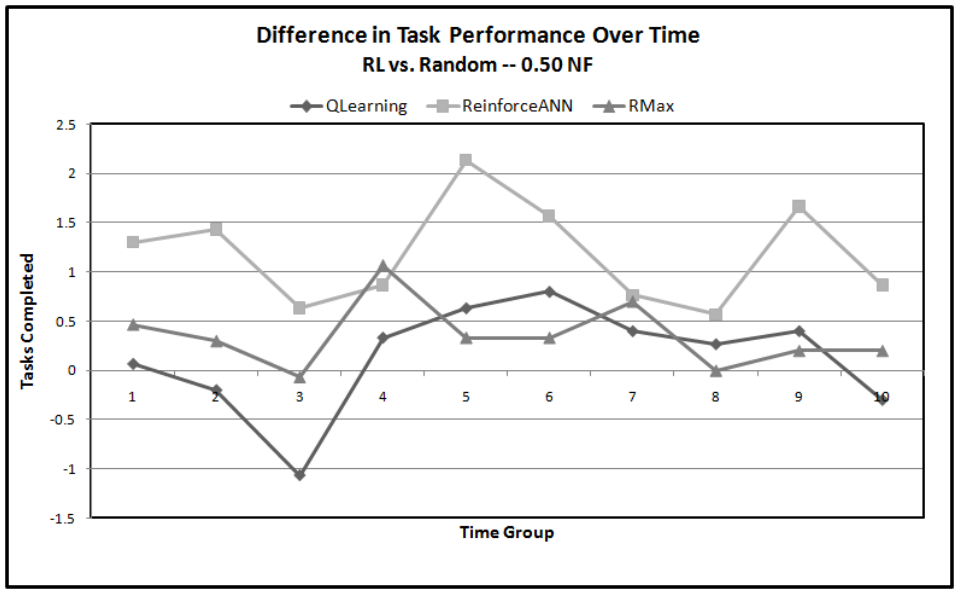

Figure A.24 Task Performance of RL vs. Random over Time in MineralMiner (0.5 NF)

A.2. UserRec Time Series Results

A.2.1. Sensing Performance

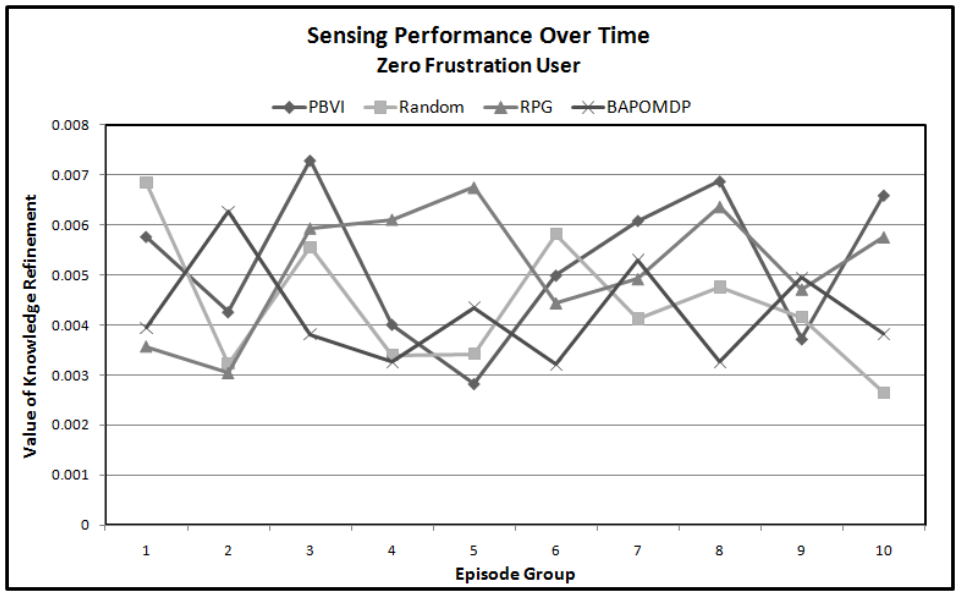

Figure A.25 Sensing Performance over Time in UserRec (Zero Frustration User)

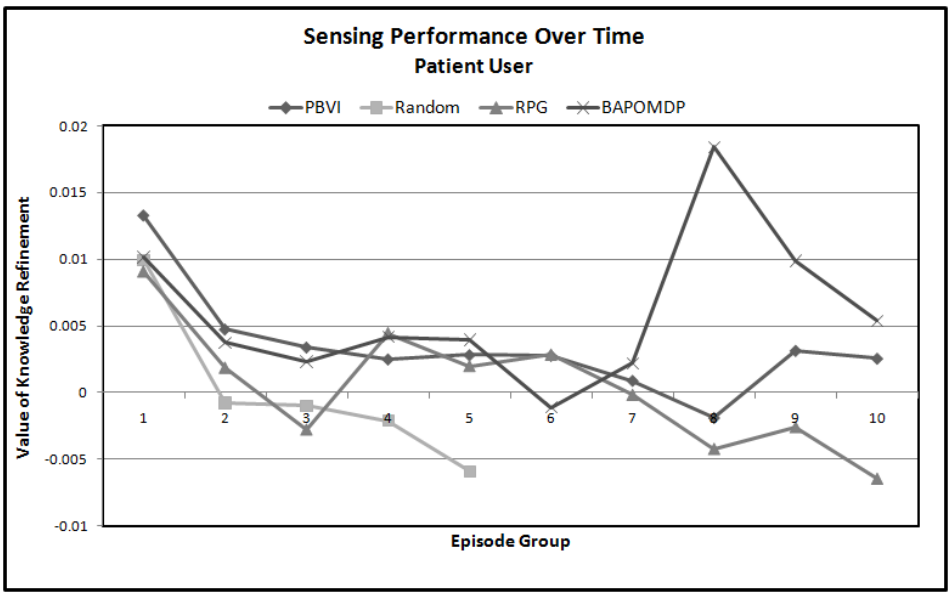

Figure A.26 Sensing Performance over Time in UserRec (Patient User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

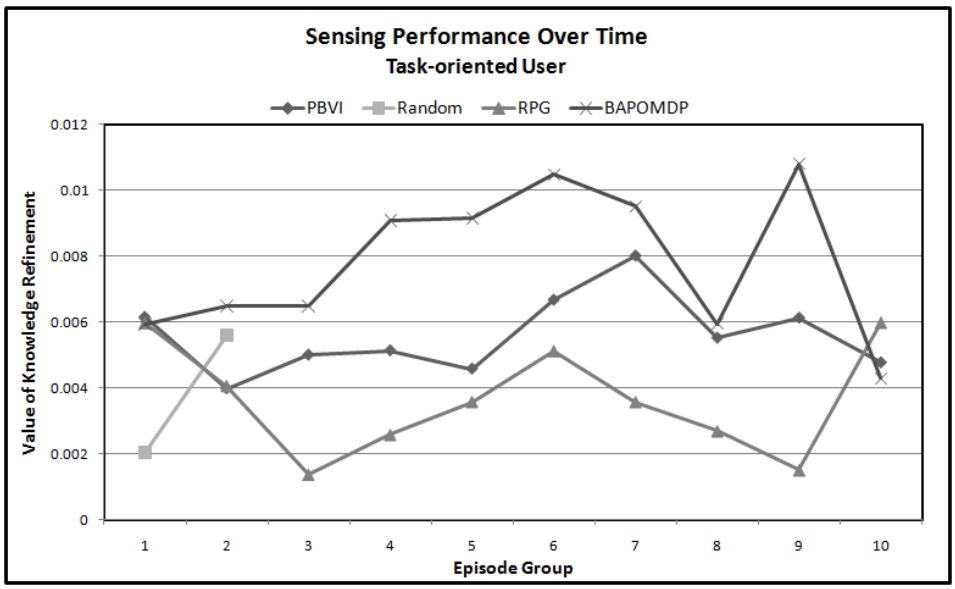

Figure A.27 Sensing Performance over Time in UserRec (Task-oriented User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

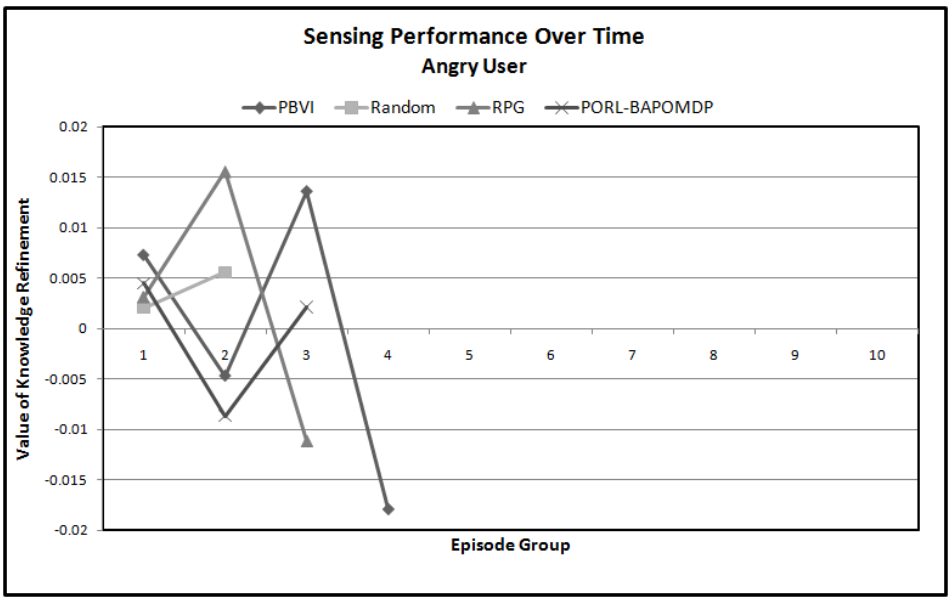

Figure A.28 Sensing Performance over Time in UserRec (Angry User)
Note: Each approach has incomplete data points because each of their simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

A.2.2. Task Performance

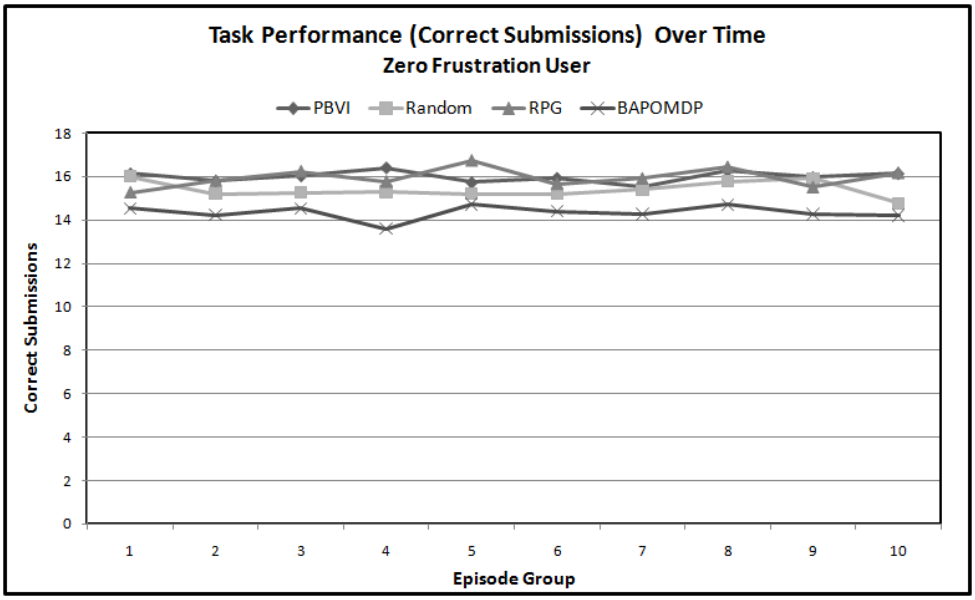

Figure A.29 Task Performance over Time in UserRec (Correct Submissions, Zero Frustration User)

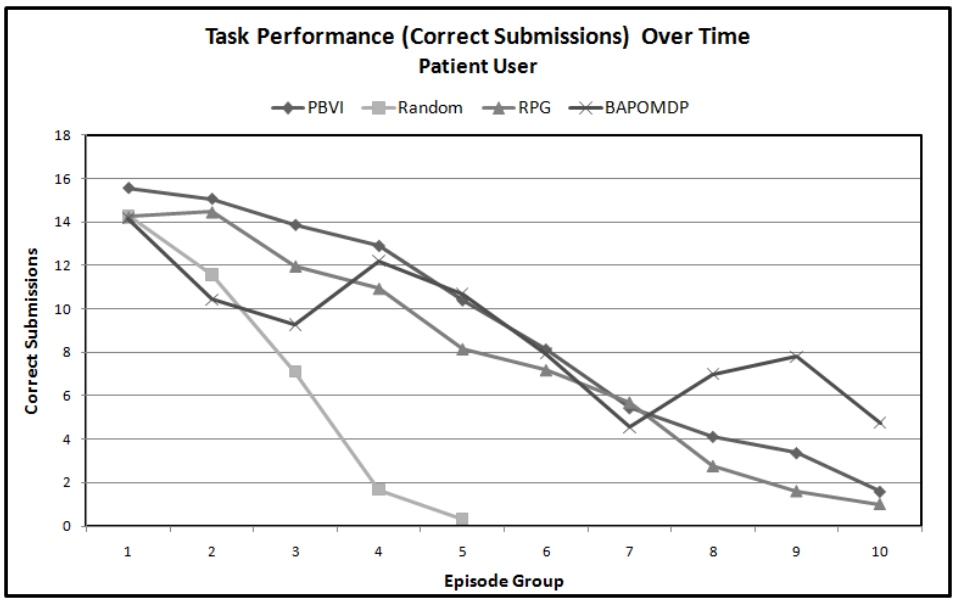

Figure A.30 Task Performance over Time in UserRec (Correct Submissions, Patient User)

Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

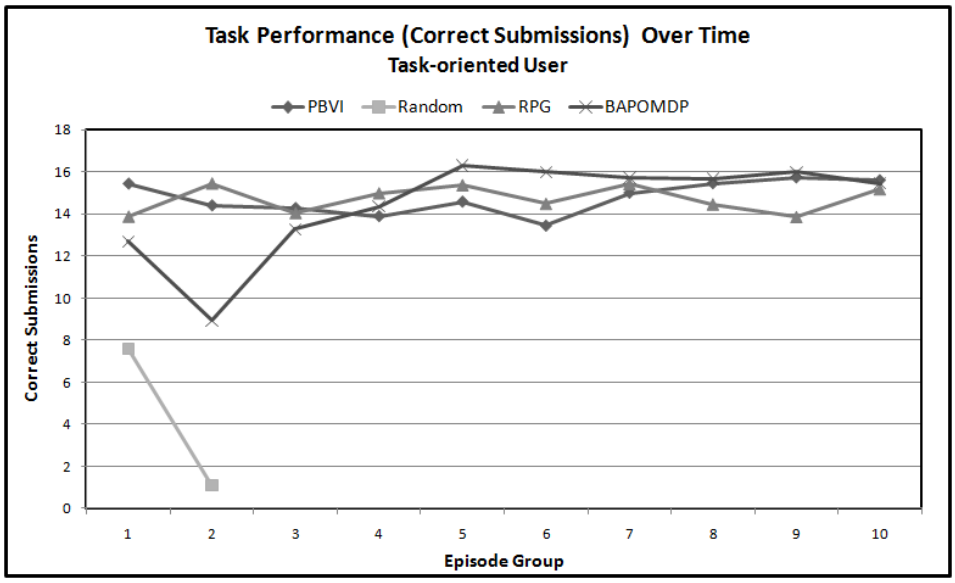

Figure A.31 Task Performance over Time in UserRec (Correct Submissions, Task-oriented User)

Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

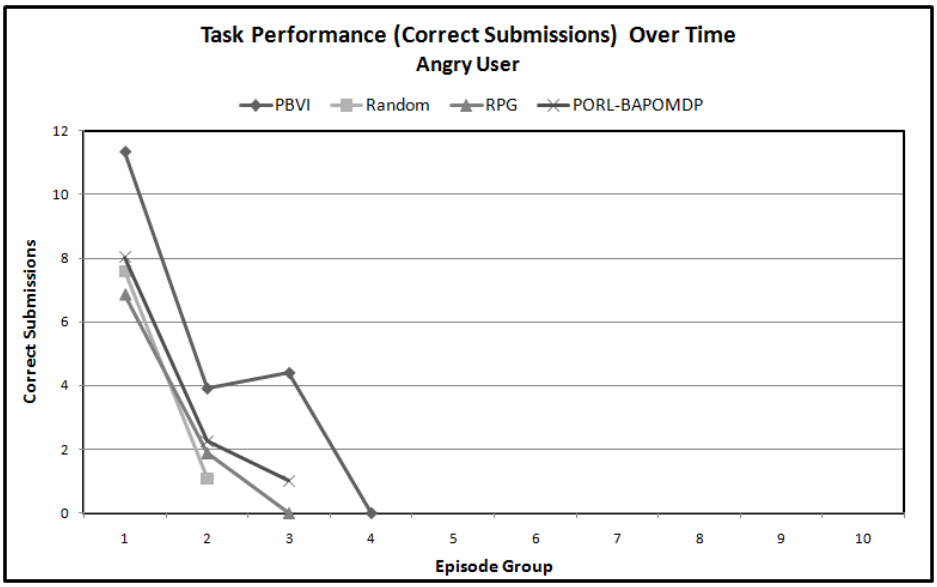

Figure A.32 Task Performance over Time in UserRec (Correct Submissions, Angry User)

Note: Each approach has incomplete data points because each of their simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

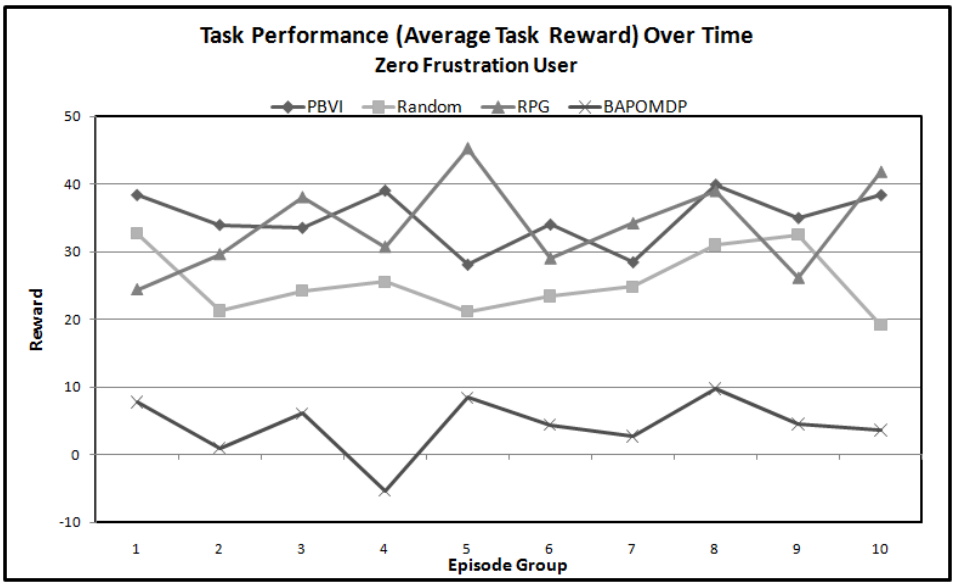

Figure A.33 Task Performance over Time in UserRec (Average Task Reward, Zero Frustration User)

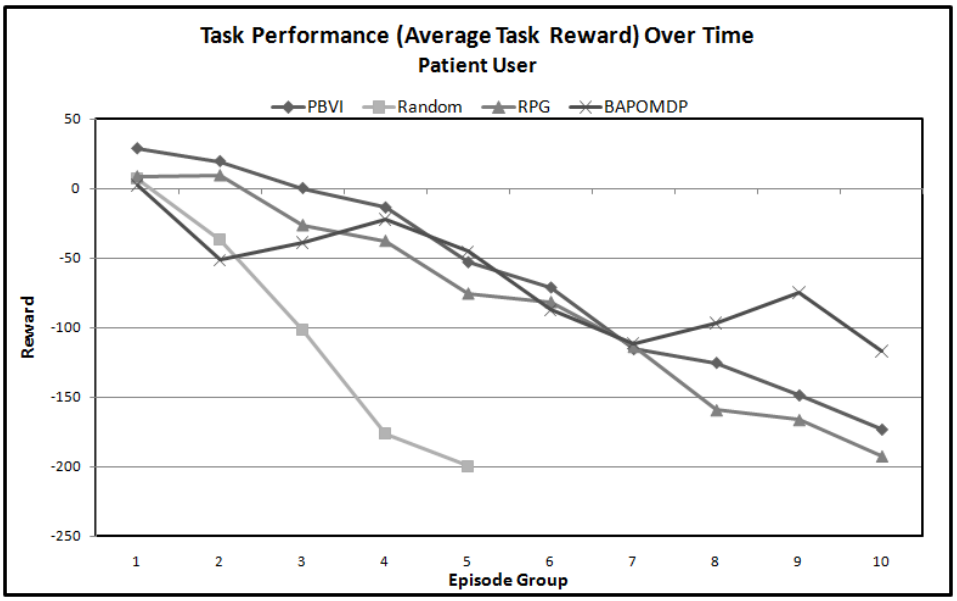

Figure A.34 Task Performance over Time in UserRec (Average Task Reward, Patient User)

Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

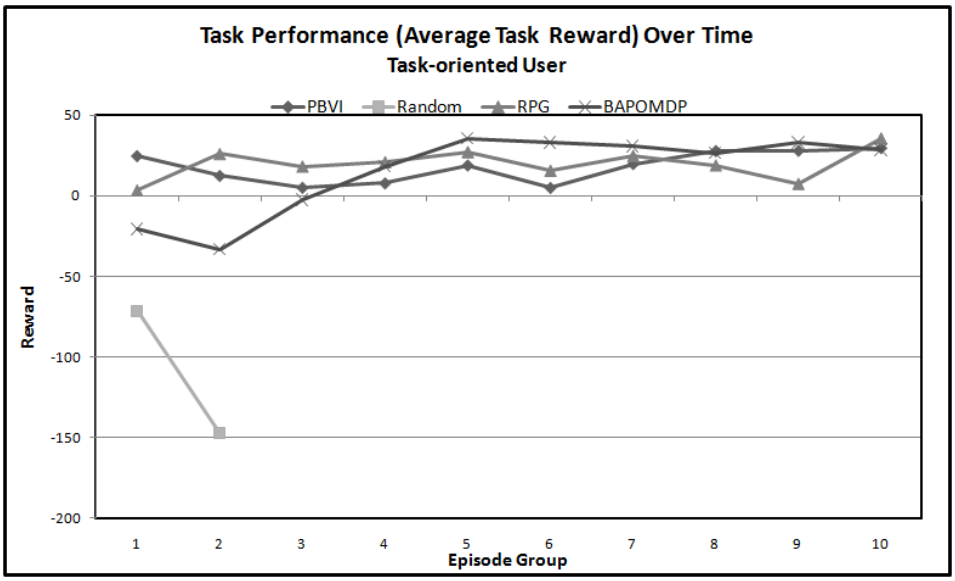

Figure A.35 Task Performance over Time in UserRec (Average Task Reward, Task-oriented User)

Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

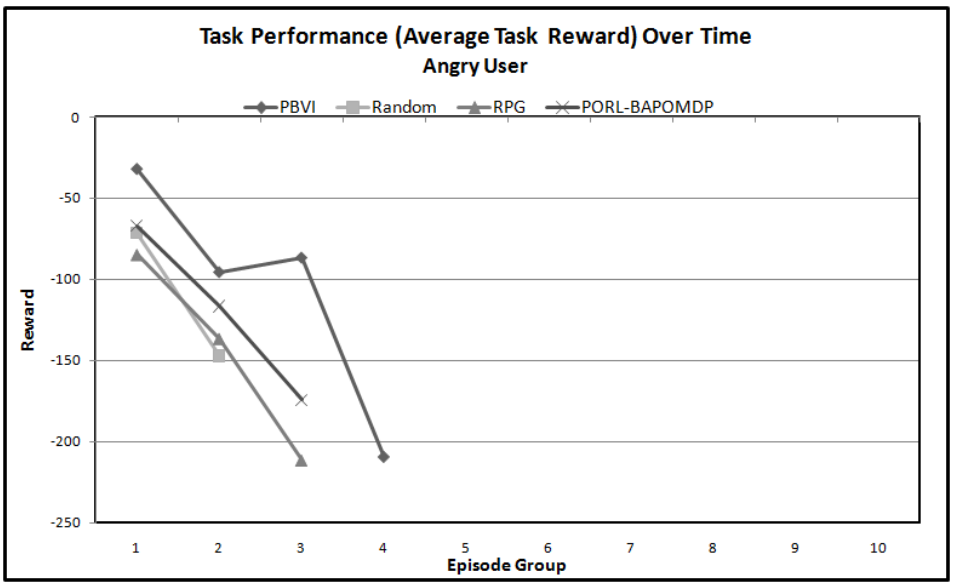

Figure A.36 Task Performance over Time in UserRec (Average Task Reward, Angry User)

Note: Each approach has incomplete data points because each of their simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

A.2.3. User Frustration

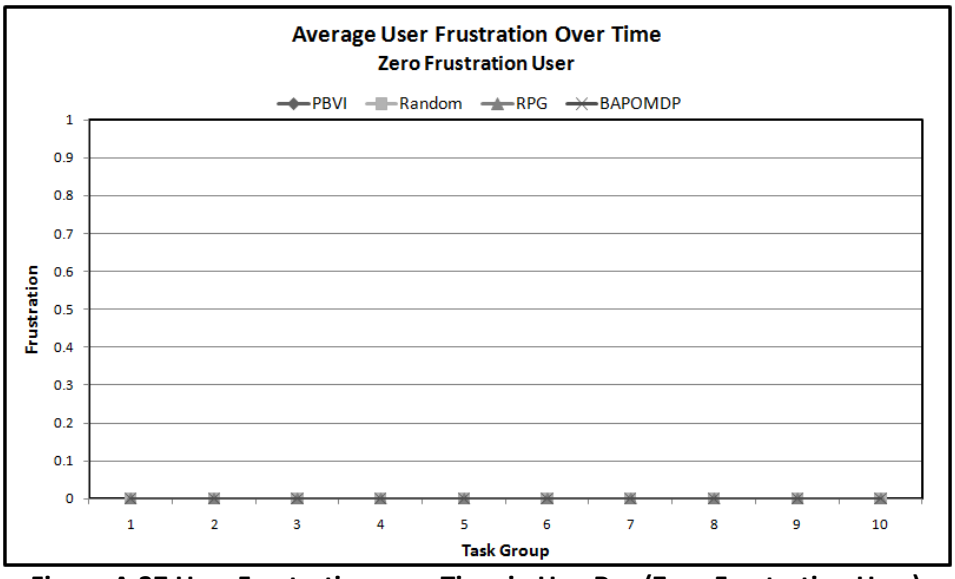

Figure A.37 User Frustration over Time in UserRec (Zero Frustration User)

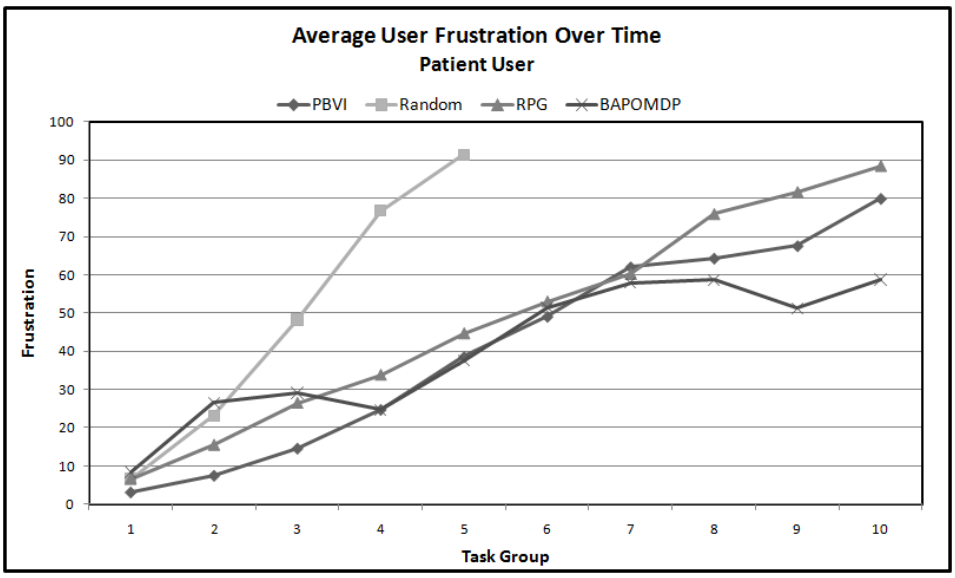

Figure A.38 User Frustration over Time in UserRec (Patient User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

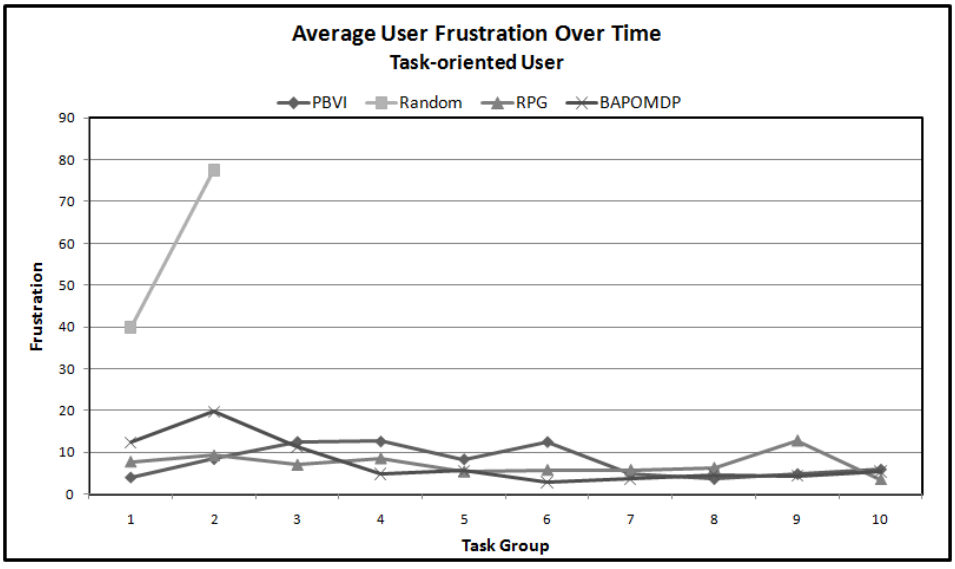

Figure A.39 User Frustration over Time in UserRec (Task-oriented User)
Note: Random has fewer data points because each of its simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.

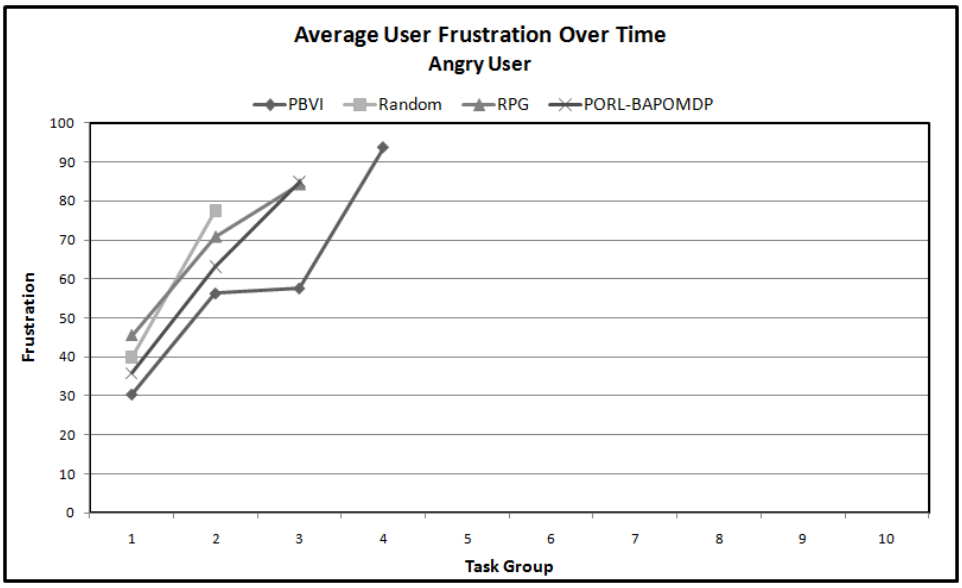

Figure A.40 User Frustration over Time in UserRec (Angry User)
Note: Each approach has incomplete data points because each of their simulation runs exceeds the user's frustration boiling point and does not survive all 200 episodes.