Auton Agent Multi-Agent Syst (2013) 26:202–244
DOI 10.1007/s10458-011-9189-y

Observer effect from stateful resources in agent sensing

Adam Eck · Leen-Kiat Soh

Published online: 1 February 2012

Abstract 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 to overcome its knowledge limitations also requires resource use, the agent's knowledge refinement is affected due to its inability to always sense when and as accurately as needed, further leading to poor decision making. In this paper, we consider what happens when sensing actions require the use of *stateful* resources, which we define as resources whose state-dependent behavior changes over time based on usage. Current literature addressing agent sensing with limited resources primarily investigates stateless resources, such as avoiding the use of too much time or energy during sensing. However, 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 action can and will distort its own outcome (and potentially future outcomes), termed the Observer Effect (OE) after the similar phenomenon in the physical sciences. Under this effect, when deliberating about when and how to perform sensing that requires use of stateful resources, an agent faces a strategic tradeoff between satisfying the need for (1) knowledge refinement to support its reasoning, and (2) avoiding knowledge corruption due to distorted sensing outcomes. To address this tradeoff, we model sensing action selection as a partially observable Markov decision process where an agent optimizes knowledge refinement while considering the (possibly hidden) state of the resources used during sensing. In this model, the agent uses reinforcement learning to learn a controller for action selection, as well as how to predict expected knowledge refinement based on resource use 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 that produce side effects such as the OE, as opposed to simply using stateless resources with no such side effect. We evaluate our approach in a *fully*

A. Eck (✉) · L.-K. Soh
Department of Computer Science and Engineering, University of Nebraska,
256 Avery Hall, Lincoln, NE 68588-0115, USA
e-mail: aeck@cse.unl.edu

L.-K. Soh
e-mail: lksoh@cse.unl.edu

and partially observable agent mining simulation. The results demonstrate that considering resource state and the OE during sensing action selection through our approach (1) yielded better knowledge refinement, (2) appropriately balanced current and future refinement to avoid knowledge corruption, and (3) exploited the relationship (i.e., high, positive correlation) between sensing and task performance to boost task performance through improved sensing. Further, our methodology also achieved good knowledge refinement even when the OE is not present, indicating that it can improve sensing performance in a wide variety of environments. Finally, our results also provide insights into the types and configurations of learning algorithms useful for learning within our methodology.

Keywords Observer effect · Stateful resources · Active perception · Agent sensing

1 Introduction

In many real-world applications of multiagent systems, such as collaborative group work applications (e.g., [24]), personal information managers (e.g., [34,55]), wireless sensor networks (e.g., [3,36]), and robotic systems (e.g., [9,33]), agents typically suffer from *bounded rationality*: a *lack of knowledge* about their environments and a *lack of resources* (e.g., CPU cycles, memory, time) used to perform agent activities. Unfortunately, performing sensing actions to overcome knowledge limitations can also require the use of limited resources. This forces a *tradeoff between the quality and/or quantity of information gathered during sensing and the cost of resource consumption to gather that information*. For example, in an intelligent wireless sensor network (e.g., [36]), sensing the environment requires battery resources that are limited, constraining the lifetime of the network and the number of observations agents controlling the sensors can receive.

In this paper, we focus on a specific instance of this tradeoff that 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*). For example, in a user support system, an agent must perform preference elicitation to model its user and guide its recommendations [2]. Sometimes, this requires directly interrupting the user (i.e., the resource) to inquire about her preferences, which can cause frustration (i.e., resource state changes) [1] and reduce the user's goodwill and patience with the system [25], especially if she is busy [32]. This increase in frustration can then lead to fewer quality responses and inaccurate user modeling. Thus, the sensing actions used to gather information to achieve a certain task can actually corrupt the information gathered and hamper the quality of the solution for the task. We call this interesting phenomenon the **Observer Effect** (**OE**) after the similar phenomenon in the physical sciences. From the perspective of the intelligent agent, the OE creates an important tradeoff—the **OE tradeoff problem** (**OETP**)—between satisfying the need for (1) achieving knowledge refinement from sensing with stateful resources to improve reasoning, and (2) avoiding knowledge corruption due to distorted sensing outcomes caused by the OE.

Although sensing with limited resources has been considered previously in the literature (c.f., Sect. 2.1), the novelty of the OE from using stateful resources makes it difficult to reuse those prior solutions. For example, in research managing the amount of time allocated to processing raw observations into information [56,58] considered information processing to be an anytime algorithm solved through performance profiles. This approach requires sensing performance to be monotonic in resource use. However, sensing

performance can be non-monotonic due to the OE. For example, additional resource use could push the resource into a bad state, resulting in worse sensing performance, thus non-monotonicity. Further, frequency adaptation solutions to sensing resource use such as those by [36] rely on accurate observation prediction using regressions. However, simple modeling techniques such as regressions will not capture resource behavior under the OE without including a good understanding of the relationship between resource state and behavior. Thus, adding resource state modeling is important to solving the OETP.

To properly address the OETP, we adopt the *active perception* perspective (e.g., [50]) to sensing where agents actively choose which sensing actions to perform, as opposed to reactively collecting whatever information is provided by the environment to the agent's sensors during its task-oriented actions. This perspective provides a vehicle for making decisions about which sensing actions to perform, given their need for resources and the consequences of resource use. In this paper, we consider active perception from a metareasoning perspective as a separate reasoning process [11], where meta-level decisions are made to control sensing (e.g., choosing actions, deciding when to stop sensing) after analyzing the agent's knowledge in order to support the agent's task-level reasoning. Using a separate reasoning process is advantageous because it allows our methodology to integrate with any task-level reasoning as a separate component, rather than modifying the original task-level reasoning used by the agent. Metareasoning is also a popular approach to bounded rationality [57].

Within this methodology, we propose a decision-theoretic solution to the OETP which models the problem of selecting sensing actions that require stateful resources as a partially observable Markov decision process (POMDP), called the **OE POMDP**. This follows previous uses of Markov decision processes (MDPs) in metareasoning (e.g., [39]). Using this model, we develop a controller for selecting sensing actions capable of reasoning about and mitigating the OE by maximizing the expected knowledge refinement performed by the agent's sensing. This controller models the relationship between resource state, observations about resource state, the possible sensing actions, the current knowledge of the agent, and the value of knowledge refinement produced by sensing. Using this model, the agent selects sensing actions that provide a maximal amount of expected knowledge refinement given the current states of resources, thereby limiting 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 frequent changes in the dynamic environment), we use partially observable reinforcement learning (PORL) to learn such a controller online as the agent interacts with its environment.

Finally, to both explore the OE in agent-based sensing and evaluate our solution, we conduct experiments in MineralMiner, a Tileworld [38] variant where agents must use stateful resources during sensing in order to refine their knowledge to support task-level decision making. Here, we consider both fully and partially observable resource states to simulate a wide range of environments. Through these experiments, we discover that our approach (1) learns to improve the knowledge refinement provided by sensing, (2) balances current versus future refinement to improve long-term behavior, and (3) exploits the relationship (i.e., high, positive correlation) between sensing and task performance to boost task performance through improved sensing. Furthermore, our solution still achieves good knowledge refinement even when the OE is not present, demonstrating that it improves sensing even outside of the OETP. Additionally, we also compare various reinforcement learning (RL) and PORL algorithms to gain insights into their advantages and disadvantages for solving the OE POMDP, finding that discrete algorithms performed better overall than continuous in the fully observable environments, whereas model-free generally did better than model-based regardless of observability.

The rest of this paper is organized as follows. We first provide background necessary for understanding the paper and discuss related work in Sect. 2. In Sect. 3, we introduce and formalize the OETP, followed by a description of our proposed OE POMDP solution methodology in Sect. 4. Section 5 describes our experimental setup and solution instantiation within the MineralMiner simulation environment. Section 6 presents the results of those experiments, including a discussion of the lessons learned from our research. In Sect. 7, we conclude by summarizing the ideas presented in this paper and describe the future work we intend to perform.

2 Background and related work

In this section, we provide the necessary background for understanding the problem and proposed solution studied in this paper, along with relevant related work from the artificial intelligence and multiagent systems literature. We first discuss the problem of using limited resources during sensing, which provides the background for our problem formalized in Sect. 3. Next, we introduce active perception and describe prior work using POMDPs to manage sensing action selection, which is the basis of our solution in Sect. 4.

2.1 Sensing with limited resources

Commonly, agents suffer from *limited knowledge* about their environments. This requires the agent to improve its knowledge by interacting with its environment so it can properly act to achieve its goals and tasks. Sensing, also called information gathering or perception, is the process through which an agent performs actions that produce observations, which in turn provide the agent with new information about the current state of the environment. With these observations, the agent can then build a more complete or more up-to-date model of the world around itself, overcoming its initial knowledge deficiency. Finally, with refined knowledge, the agent can then make better decisions and better accomplish its tasks and goals.

However, in many environments, resources used to produce such observations during sensing are also *limited*. For example, an agent might possess a limited number of sensors to cover the environment (e.g., [28]), each sensor might have a limited battery life and thus can only produce a limited number of observations (e.g., [36]), or processing the raw observations to refine knowledge could require costly computational and time resources (e.g., [56]). Thus, not only do limited resources affect the ability of an agent to reason and accomplish its tasks, but limited resources can also affect an agent's sensing actions.

Previously, research from the artificial intelligence and multiagent systems communities have looked at managing agent sensing with limited resources where resource behavior is assumed to be independent of the use of the resource. For example, [56,58] studied the use of computational resources (e.g., time) to process raw observations into information useful during reasoning. Here, information processing was treated as an anytime algorithm and increased time spent on processing led to no worse (and usually better) information. Further, [16,17,31] considered the use of money and time in the collection of information from sources distributed across the internet that were queried to provide information to a human decision maker. Additionally, [36] studied sensing frequency adaptation in sensor networks to balance the quality of information gathered through the individual sensors with the lifetime of the network that was reduced by energy consumption through sensing. Finally, [26–29] looked at problems such as covering an environment with a limited number of sensors to optimize one or more objective functions (e.g., threat detection).

In each of these works, while the *result* of sensing (e.g., increased information quality or quantity) could improve with additional resource use, the *behavior* of the underlying resources (e.g., time, money, energy) was constant for each use of the resource. For example, the same amount of data could be transmitted across the internet or the same number of CPU cycles could be performed while processing information during each time unit, each unit of money had the same value, each unit of energy could power the same amount of sensor action, and each sensor was capable of performing the same amount of sensing. In our research, however, we study *resources whose behavior* does *change with its use during sensing*. For example, consider an intelligent user support application. Here, the user is a resource used by the agent to gather information about her preferences. During sensing, the system essentially interrupts the human user from what she is doing. As a result, her frustration with the system increases or decreases which can change her behavior [25]. We require different solutions for sensing action selection than those proposed by the prior literature in order to account for this changing behavior of resources, as well as the impact of resource behavior on observations during sensing. We provide more details on this need in Sect. 3.3.

2.2 Active perception with POMDPs

In general, making decisions about what sensing actions to perform constitutes **active perception** (also called **active sensing**) in intelligent agents. Specifically, active perception is the process through which an agent *actively* manages its sensing behavior by choosing actions to perform based on their observational benefit (e.g., accuracy, uncertainty reduction) or cost, rather than *passively* relying on whatever observations happen to be produced by the environment [50]. Addressing agent sensing from this perspective enables the agent to explicitly consider the benefits and costs of sensing actions, allowing it to *proactively* aim to maximize its sensing performance, as opposed to *reactively* rely on (potentially) suboptimal observations. All of the approaches considered in Sect. 2.1 above for managing resource use during sensing qualify as active perception approaches. Traditionally, active perception has been modeled as a sequential decision problem and solved using classical techniques such as the POMDP. We take this approach in our solution, described in more detail in Sect. 4. Next, we introduce the POMDP to provide background for our solution methodology, then describe the prior use of POMDPs in active perception.

2.2.1 POMDP

We begin by introducing the (fully observable) **MDP** [23]. Formally, a MDP 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 actions available to the agent, and $T(s,\ a,\ s') \in [0,\ 1]$ is a probabilistic transition function representing the likelihood that the environment changes from state s to s' if the agent takes an action a, that is $T(s,\ a,\ s') = P(s'|s,\ a)$, and $R(s,\ a) \in \mathbb{R}$ is a function modeling the reward of taking an action 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 actions that optimize the rewards received by the agent for its actions. The values of states used to compute policies are represented by a set of Bellman equations optimizing discounted, expected future rewards:

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

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.

Solving for an optimal policy then requires solving the corresponding set of equations:

$$\pi(s) = \text{argmax}_{a \in A} R(s, a) + \gamma \sum_{s' \in S} T\left(s, a, s'\right) V^*(s'), \quad \forall s \in S \tag{2}$$

using exact (e.g., dynamic programming) or approximation (e.g., value iteration) techniques.

When the state of the environment is *not* directly observable (i.e., is hidden from the agent), a more appropriate model is a ***POMDP*** [23], which augments a MDP to form a six-tuple $\langle S, A, \Omega, T, O, R\rangle$ where S, A, T, and R are as in the MDP discussed above, $\Omega = \{o\}$ is a set of observations produced when an action is taken, and $\mathrm{O}(s', a, o) \in [0, 1]$ is the probabilistic observation function representing the likelihood that o is observed after taking action a leading to (hidden) state s', that is $\mathrm{O}(s', a, o) = P(o|s', a)$. In this decision process, the hidden state of the process must be estimated based on the observations produced by actions. 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 action a:

$$b'\left(s'\right) = \frac{1}{Z}\mathrm{O}(s', a, o) \sum_{s \in S} T(s, a, s')b(s) \tag{3}$$

where Z is a normalization factor insuring new belief values remain in $[0, 1]$ and sum to 1 (since this vector represents a probability distribution over states), and b' is the new belief state.

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 developing one for a MDP because solving the Bellman equations (Eqs. 1–2) for a POMDP would require creating a policy over an infinite number of belief states. Thus, various techniques for creating POMDP policies rely on approximations, such as point-based value iteration (PBVI) (e.g., [12,37]), or limited search, such as using short depth decision trees which exactly solve the POMDP [41].

2.2.2 *Use of POMDPs in active perception*

Now that we have introduced the POMDP, we are ready to describe various prior research using POMDPs to control active perception, which is similar to our OE POMDP approach described in Sect. 4. First, [18] studied the problem of classification from the perspective of an agent identifying a feature of its environment using information from observations collected through sensing. Here, the agent represents the possible classification labels as the states of the POMDP, observations produced through sensing actions reveal information about the true classification, and the agent chooses actions to minimize costs associated with those actions until it is confident in the true label based on its previous observations.

Additionally, POMDPs have been used to manage agent sensing behavior when tasked with supporting human users. In the Preference Elicitation POMDP [7,12], an agent chooses sensing actions which best determine a user's preference over a set of choices, items, or goals, then acts in some way to support the user based on her preferences. Here, the state space covers the possible user preferences, for example represented as a utility function over choices [7] or individual goals [12]. Similarly, [54] studied the problem of determining user goals in a dialog management system. In these POMDPs, observations from sensing refine the agent's belief state, representing a probability distribution indicating the likelihood that any of the states (e.g., user's utility function or goal) is the correct one. Rewards in these POMDPs are the expected utility of the agent supporting the user based on the current beliefs about the user's utility function [7], including fixed costs penalizing sensing actions and wrong intelligent support while highly rewarding correct types of support provided [12,54].

Further, [45] extended single agent active perception using a POMDP to coordinating active perception between multiple agents. Specifically, Spaan considered a multiagent system composed of fixed position video cameras and mobile robots responsible for monitoring a public space for events like fires or people entering a specific space. Like in the Preference Elicitation POMDP, these agents first manage their sensing to build an accurate world model, then act in response to detected events. For example, the agents can tilt camera viewing angles or get an up-close look through the mobile robots, then react to detected fires or assist humans wanting to find locations of interest.

Of note, in each of these prior works, the states in the POMDP represented the possible information values the agent was trying to discern through sensing: classification labels [18], user preference functions [7], user goals [12,54], and the presence of events in an area of interest [45]. Thus, the belief state in the POMDP, which is improved through observations, naturally reflects the likelihood that any of the particular information values is the correct one. Using this belief state and rewards for acting within tasks on sensed information, the agent can determine which actions to take to revise its beliefs, or whether to stop sensing and simply complete its task. However, since the state space only includes possible information values targeted through sensing, this approach neglects additional factors that might affect the quality of observations produced during sensing, such as the state of resources used during sensing that can produce the OE. This is similar to the handling of resources in non-sensing problems, such as hand washing assistance for the elderly [20] where user attitude, awareness, and responsiveness that affect assisted user behavior are included in the state space. Thus, in this paper we adapt the popular POMDP-based active perception approach by extending the notion of POMDP states to include the state of resources used during sensing to directly account for the effect of resource state on observations. We cover this adaptation in more detail in Sect. 4.

Finally, in some applications of multiagent systems, the only task of the agent is to continually monitor its environment—a task that has no rewards aside from refining the agent's knowledge. Thus, traditional task rewards (such as those employed in the previously mentioned research by [7,12,45,54]) are not available to guide the sensing actions of the agent. To handle this problem, [4] developed an extension of the POMDP called the ρ POMDP that uses changes in the belief state (i.e., agent knowledge) as the rewards received for sensing actions, as opposed to non-existent task rewards. This allows the agent to focus solely on the improvement in its knowledge based on the sensing actions chosen through active perception. We use a similar reward formulation in our approach (c.f., Sect. 4), but we add reinforcement learning to learn the dependence of knowledge refinement on the state of the underlying resources used during sensing.

Table 1 OETP definitions

Symbol	Definition
K	The agent's current knowledge
$D = (d_i)$	Task-level decision sequence faced by the agent
$AC = \{ac_j\}$	Set of sensing activities
$S = \{s_k\}$	Set of information sources
$C = \{ac_j, s_k\}$	Set of sensing actions (i.e., valid activity/source pairs)
$R = \{r_l\}$	Set of stateful resources
$RN(ac_j, s_k)$	Set of resources needed by ac_j, s_k
St	Set of all possible resource states
St_{r_l}	Set of possible states of r_l
$\sigma(r_l)$	Current state of r_l
$\delta(r_l, \sigma, ac_j, s_k)$	State transition function for r_l
Obs	Set of all possible observations about resource state
Obs_{r_l}	Set of possible observations about the state of r_l
$\phi(r_l)$	Most recent observation about the state of r_l
$\theta(r_l, \sigma, ac_j, s_k)$	Observation of the state of r_l from ac_j, s_k
$info(ac_j, s_k, RN, \sigma)$	State-dependent set of information provided by ac_j, s_k
$\otimes$	Knowledge refinement operator
K'	Refined knowledge from information provided by sensing
$KR(ac_j, s_k, RN, \sigma, K, d_i)$	State-dependent value of knowledge refinement from ac_j, s_k with respect to d_i
$V(K, d_i)$	Value of knowledge with respect to d_i
$infoRequired(K, d_i)$	Determines whether or not the information required for decision d_i is in K

3 OE tradeoff problem

In this section, we describe and formalize the focus of this paper: the OETP. We begin by providing a brief overview of the context of our problem by representing sensing with limited resources as a sequential decision problem. Then, we focus on a specific subset of possible resources used during sensing: *stateful* resources. Next, we present a consequence of using stateful resources during sensing: the OE. Finally, we define the OETP, which arises out of the OE. We provide a summary of the notation used in this section in Table 1.

To begin, we consider the general active perception problem where an agent over time must make task-level decisions from a sequence $D = (d_i)$. Each decision d_i requires information from the environment. Such information is available either (1) in the agent's internal knowledge base[1] 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 actions (i.e., activity/source pairs) form a set of two-tuples $C = \{\langle ac_j, s_k \rangle\}$. Thus, for each decision d_i, the agent must also sequentially choose which sensing actions to perform in order to gather the necessary information missing from its current knowledge. Specifically, the agent performs a series of sensing

[1] Here, K represents the knowledge held by the agent from an abstract perspective. The specific representation used by the agent, such as a POMDP belief state (c.f., Sect. 2.2.1) or a set of probabilistic beliefs (c.f., Sect. 5.1), depends on the implementation of the agent.

actions until it has the information necessary to support its task-level decision, then it acts on its task. In the following, we refine this problem to include the OETP.

3.1 Stateful resources

To perform sensing actions, the agent must use limited resources. In this paper, we are focused on what happens when the agent uses a specific category of resources, which we call *stateful* resources. We define **stateful resources** as resources whose behavior depends on some notion of internal state for the resource that changes over time based on resource usage. This is in contrast to *stateless* resources, which always behave the same way regardless of past usage.

Stateful resources exist in a variety of multiagent applications and environments. For example, a network shared by multiple agents can be a stateful resource whose behavior depends on its bandwidth (resource state). Here, sensing actions by agents called active monitoring [30] that monitor network state actually introduce additional traffic into the network, thereby changing the state (e.g., bandwidth) of the resource. Depending on the network protocols used within the application (e.g., whether or not they delay packet delivery based on earlier packet loss), different network states can create different behaviors (e.g., latencies), limiting the usefulness of the network to the agents. Another example of a stateful resource is a human user whose behavior depends on her current frustration (again resource state). Here, interrupting the human user to gather information about her preferences and activities, such as in a recommender system (e.g., [2]) or personal information management (e.g., [10,34,55]) application, can distract the user's cognitive processes [32] leading to higher frustration levels [1,32] and affecting future interactions with the user.

However, while such examples of stateful resources exist, previous research involving agent sensing (c.f., Sect. 2.1) has generally assumed resources used during sensing were stateless. As a result, agents would reason about or perform sensing without considering the impact of sensing actions on the states of the very resources used during sensing. Potential negative consequences of this type of sensing behavior are described in Sect. 3.2 as we formalize the OE.

In our problem formulation, stateful resources are represented by the set $R = \{r_l\}$ which can take on possible states $St = St_{r_1} \times St_{r_2} \times \cdots \times St_{r_{|R|}}$, 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 $\sigma : R \rightarrow St_{r_l}$. 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:

$$\sigma'(r_l) = \delta\left(r_l, \sigma, ac_j, s_k\right) \tag{4}$$

with $\delta : R \times St_{r_l} \times AC \times S \rightarrow St_{r_l}$ and could be deterministic (i.e., always return the same state for each combination of inputs) or stochastic (i.e., return different states according to an internal probability distribution).

However, the state of the resource might be hidden from the agent (e.g., user frustration), so state must be estimated using information contained in observations produced by sensing. These possible state observations are represented by $Obs = Obs_{r_1} \times Obs_{r_2} \times \cdots \times Obs_{r_{|R|}}$, where Obs_{r_l} is the set of the possible observations about the state of resource r_l. The most recent observation of a resource is denoted by $\phi(r_l)$ with $\phi : R \rightarrow Obs_{r_l}$. The specific observations observed depend on the state of the resource and the sensing action performed, which is represented by:

$$\phi(r_l) = \theta\left(r_l,\ \sigma,\ ac_j,\ s_k\right) \tag{5}$$

with $\theta : R \times St_{r_l} \times AC \times S \rightarrow Obs_{r_l}$ which could also be deterministic or stochastic, as with the state transition function (Eq. 4).

Please note these observations used to estimate resource state may or may not be the primary targeted information of the agent's action to refine the agent's knowledge. For example, an intelligent user interface agent chooses sensing actions to learn about the user's preferences, but these actions also produce secondary information about the user's frustration. On the other hand, the primary targeted information of sensing might also be used to predict resource state. For example, in a network monitoring application the agent's sensing measures network bandwidth, which is also the state of the resource.

Further, we have assumed in the above formalism that the behavior of each resource is independent of every other resource. That is, the state transitions and observations produced by a resource depend only on that resource. However, this formalism could be easily extended such that resources are dependent on one another, in which case the transition and observation functions would represent the *joint* probabilities of the resources changing state and producing observations.

3.2 Observer effect

As seen from the examples of stateful resources provided in the previous section, in general using stateful resources requires the agents to consider the impact of the resource usage on the resource's behavior. This is because sensing actions change the state of the resource used during sensing, and the resulting state change can lead to a different behavior of the resource, thus affecting the outcome of the sensing action. 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 OE after a similar phenomenon in the physical sciences.

In our problem formalism, performing a sensing activity ac_j on an information source s_k produces $info(ac_j,\ s_k,\ RN,\ \sigma)$. This information is used to refine the knowledge of the agent through the domain dependent knowledge operator $\otimes$ (e.g., belief state updates from Sect. 2.2.1):

$$K' = K \otimes info(ac_j,\ s_k,\ RN,\ \sigma) \tag{6}$$

Here, we include the resources and their states because the actual information produced can be distorted through the OE. Thus, considering the state of resources used during sensing is important to the agent's sequential (sensing) decision process to address such knowledge distortion.

Such distortion from the OE can occur for several reasons, depending on the influence of resource behavior on sensing outcomes. One example of the OE occurs when sensing accuracy depends on the behavior of the resource, resulting in a situation where *a sensing action reduces its own accuracy.* In our earlier example of the stateful network resource, the additional traffic produced by active monitoring reduces bandwidth [30], which increases congestion and latency [14]. As a result, observations produced do not reflect the true state of the network when sensing is not performed. Thus, using stateful resources and OE can cause a *cost with respect to information quality* (e.g., reduced accuracy).

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., [5,43]), 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., [5]), 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.

To clarify, in this latter example, it appears at first that a *stateless* resource (sensor energy level) *directly* causes the OE (outdated information). This would be in contrast to our earlier claim that the OE is the result of *stateful* resources. However, this example actually contains a stateful resource—the wireless sensor network, whose behavior depends on the energy of the individual nodes. Specifically, it is the behavior of the stateful wireless sensor network that changes based on sensing actions, not the static stateless energy behavior (whose units always power the individual sensors in the same way). Thus, it is possible for a stateless resource to *indirectly* lead to the OE, if that resource is actually the state of the stateful resource used by the agent.

A final example of the OE occurs when the *quantity* of information provided by sensing depends on the behavior of the resource. For instance, in our earlier user preference elicitation example, prompting the user to elicit her preference is an interruption that affects her feelings towards the system [25], which can lead to less willingness to provide responses, resulting in fewer responses. 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 unable to transmit as much information back to the agent. Thus, using stateful resources and the OE can also cause a *cost with respect to information quantity* (i.e., reduced quantity).

3.3 Observer effect tradeoff problem

Considering all of these examples, we see that the OE 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 that is used to produce good decisions during reasoning, the OE leads to the following difficult problem:

> **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 satisfying the need for (1) knowledge refinement from sensing with stateful resources, and (2) avoiding knowledge corruption due to distorted sensing outcomes caused by the Observer Effect.

That is, as an agent chooses to perform more sensing actions to provide more information to support its reasoning, the benefits might be offset by a decrease in sensing performance caused by the OE 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 OE 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 OE places stress on the sensing action selection of agents to collect information used to refine the agent's knowledge, necessary to properly achieve its goals.

Specifically, the agent is concerned with the *value* of a change in knowledge, which 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: $AC \times S \times 2^R \times St \times K \times D \rightarrow \mathbb{R}$:

$$KR(ac_j,\ s_k,\ RN,\ \sigma,\ K,\ d_i) = V(K',\ d_i) - V(K,\ d_i) \tag{7}$$

where $V(K,\ d_i)$ with $V : K \times D \rightarrow \mathbb{R}$ measures the domain-dependent value of the subset of agent knowledge necessary for making a given decision. For example, this value might be the confidence or uncertainty the agent has in its knowledge, changing with new information observed. We note that this measure is very similar to value of information measures (e.g., the contribution of information towards a particular decision) used elsewhere in the sensing literature, such as by [16,17] and [7] (c.f., Sect. 2). However, our value (1) is calculated over agent knowledge refined after sensing which might be used for any arbitrary collection of decisions, and (2) includes the impact on knowledge of the OE. Further, this equation also represents an objective function for sensing, similar to the OSP studied by [26–29].

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

$$\text{Given } K,\ d_i,\ RN,\ \sigma,\ C$$

$$\text{Choose } (ac_j,\ s_k) \in C \quad \text{according to } KR(ac_j,\ s_k,\ RN,\ \sigma,\ K,\ d_i) \tag{8}$$

$$\text{Until } infoRequired(K,\ d_i) \text{ is satisfied} \tag{9}$$

Here, the goal of the agent is to select sensing actions (i.e., activity/source pairs) based on the value of refinement in agent knowledge with respect to its current reasoning decision. This occurs until the agent has the knowledge it needs to successfully make a decision in order to achieve its goals, as defined by a domain-specific $infoRequired(K,\ d_i)$ function. For example, the agent could stop when it has collected the necessary information to make a decision: $info(d_i) \subseteq K$, or the agent's knowledge confidence is high enough to make a decision: $confidence(K) \geq confidence(d_i)$.

The choice mechanism used in Eq. 8 might depend on the application and/or the decisions facing the agent. For example, it could myopically choose the sensing action maximizing knowledge refinement:

$$\text{argmax}_{(ac_j,\ s_k) \in C}\ KR(ac_j,\ s_k,\ RN,\ \sigma,\ K,\ d_i) \tag{10}$$

Additionally, if the agent knows what decisions it will be facing in advance, it can also non-myopically choose sensing actions which maximize the knowledge refinement for all known decisions, for example by using the following choice mechanism[2]:

$$\text{argmax}_{(ac_j,\ s_k) \in C} \sum_{t=i}^{n} KR(ac_j,\ s_k,\ RN,\ \sigma,\ K,\ d_t) \tag{11}$$

for the next n known decisions. However, the agent should not *over* sense and use resources too much, increasing the likelihood of pushing resources into bad states and corrupting knowledge. Thus, the agent should only sense until it has the required information for its decision(s), handled by the constraint in Eq. 9. Overall, by maximizing knowledge refinement without over-sensing, the agent effectively balances the OE Tradeoff between the need for knowledge refinement to make decisions and the need to avoid knowledge corruption due to the OE.

We would like to note that other constraints might be added to the selection process, depending on the application and environment. For instance, in applications where specific

[2] This mechanism could easily be modified to maximize expected future decisions by including a likelihood of the agent facing each decision if the decisions instead are not known for sure in advance. Additionally, a discount factor (c.f., Sect. 2.2.1) could be included for uncertain future decisions to avoid gathering information which might not be necessary by de-emphasizing future refinements that become more uncertain the farther out the decision.

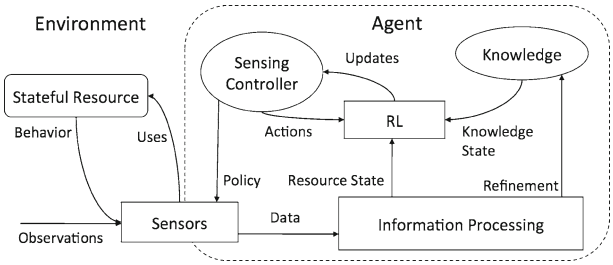

Fig. 1 Methodology overview

states of a resource should be avoided, where these states can be predicted as those which provide expected knowledge corruption, an additional constraint can be included. Here, the agent also stops sensing when no expected refinement is possible, modeled as:

$$KR\left(ac_j,\ s_k,\ RN,\ \sigma,\ K,\ d_i\right) \leq 0 \tag{12}$$

For example, this constraint might be useful in our user preference elicitation example. Here, over-frustrating the user should be avoided or the user might lose faith in the system [25] and quit wanting to use the system altogether.

4 OE POMDP

In this section, we present our proposed decision theoretic solution to the OETP. We begin by mapping the OETP formalization defined in Sect. 3 into a POMDP, which we brand the OE POMDP. This approach follows a popular approach to managing sensing action selection in active perception (c.f., Sect. 2.2). Afterwards, we detail how a sensing action selection controller following the OE POMDP can be learned through reinforcement learning in order to solve the OETP when the relationship between resource state and knowledge refinement must be learned through experience.

The overall solution methodology is summarized in Fig. 1. In short, an agent uses sensors to collect information through observations from the environment, which is then processed and used to revise its limited knowledge. This requires the use of stateful resources whose behavior influences the observations collected. To control the agent's sensors and use of stateful resources, the agent uses active perception, specifically by modeling sensing action selection as an OE POMDP and using reinforcement learning to build and revise a sensing controller. This sensing controller is a metareasoner that decides how to gather information to support the agent's task-level reasoning. After describing the details of this methodology in the following subsections, we will summarize this approach from a high level algorithmic, step-wise perspective.

4.1 OE POMDP mapping

Recall (c.f., Sect. 2.2.1) that a POMDP models a sequential decision process as a tuple $S,\ A,\ \Omega,\ T,\ O,\ R$, where an agent chooses actions maximizing a reward function based on the hidden state of the environment estimated through observations. Here, we demonstrate how to solve the OETP by modeling it as a special **OE POMDP**, summarized in Table 2.

Table 2 Transformation from OETP to OE POMDP

OE POMDP	OETP transformation	Description
$s = \langle R_S,\ K_S\rangle \in S$	$R_S = \langle \sigma(r_1),\ \sigma(r_2), \ldots, \sigma(r_{\lvert R\rvert})\rangle$ $K_S = \text{state}(K)$	The **sensing states** are combinations of resource and knowledge states
$a \in A$	$a = \langle ac_j,\ s_k\rangle \in C$	The **sensing actions** are the valid sensing activity/source pairs
$T(s,\ a,\ s')$	$\delta(r_l,\ \sigma,\ ac_j,\ s_k)$ $K \otimes info(ac_j,\ s_k,\ RN,\ \sigma)$	**Sensing state changes** depend on the changes in resource and knowledge states due to a chosen sensing action
$o = R_O,\ K_O \in \Omega$	$R_O = \langle \phi(r_1),\ \phi(r_2), \ldots, \phi(r_{\lvert R\rvert})\rangle$ $K_O = state(K)$	The **observations** are combinations of observations about resource states and fully observable knowledge state
$O(s',\ a,\ o)$	$\theta(r_l,\ \sigma,\ ac_j,\ s_k)$ $K \otimes info(ac_j,\ s_k,\ RN,\ \sigma)$	The **observation function** depends on the observations of resource state and the changes in knowledge state
$R(s,\ a)$	$KR(ac_j,\ s_k,\ RN,\ \sigma,\ K,\ d_i)$	The **reward** for making choices given the current sensing state is the value of knowledge refinement as the result of sensing

We define the current *sensing state* $s \in S$ as a tuple $\langle R_S,\ K_S\rangle$ consisting of the current state R_S of all resources and the current state of the agent's knowledge K_S, with

$$R_S = \langle \sigma(r_1),\ \sigma(r_2), \ldots, \sigma(r_{|R|})\rangle \tag{13}$$

$$K_S = \text{state}(K) \tag{14}$$

where state(K) is an implementation-specific description of the current knowledge of the agent (e.g., quantity of information, top belief state value). 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 refinement (Eq. 7): resource state through the OE and knowledge state through the improvement in knowledge.

We define the *active perception actions* $a \in A$ in the POMDP to be the set of possible sensing activity/source pairs $\langle ac_j,\ s_k\rangle \in C$ in the OETP. In our example, the actions represent the different combinations of sensing activities the agent can perform to gather information about the user's preference (e.g., asking the user directly, observing her behavior in a task) with the different sources of such information (e.g., the user herself, her peers).

Furthermore, we define the transition function $T(s,\ a,\ s')$ in the POMDP as a probability measure over the changes to resource state $\sigma(r_l)$ and knowledge state state(K') by a sensing action $\langle ac_j,\ s_k\rangle$ determined by Eqs. 4 and 6. In our example, interrupting the user to ask about her preference increases frustration [1,32], changing her state. Based on the user, her frustration could increase by different amounts at different times for the same interruption dependent on her current frustration. Also, the information gathered through the interruption revises the agent's knowledge, changing the knowledge state.

In many environments, the true state of the resource might be hidden from the agent, thus it must be estimated from observations[3] that result from taking actions using the resource. However, we generally assume that the knowledge state of the agent is not hidden. Since the overall sensing state is a tuple with two parts, we also represent the observations $o \in \Omega$ in the POMDP as a tuple $\langle R_O, K_O \rangle$, with:

$$R_O = \langle \phi(r_1), \phi(r_2), \ldots, \phi(r_{|R|}) \rangle \tag{15}$$

$$K_O = state(K) = K_S \tag{16}$$

For example, the speed of the user's responses to interruptions, audible expressions, or her heart rate could provide evidence indicating her current frustration level. Note that since we assume full observability of knowledge state, there is a one-to-one correspondence between observations about knowledge state and actual knowledge states.

The specific observations about sensing state made by an agent are determined by the observation function $O(s', a, o)$ in the POMDP, defined as the probability of the agent making observation $o \in \Omega$ about sensing state when action a produces state s'. For example, once a user becomes more frustrated after a recent sensing interruption, she might respond faster to return to her original task. However, she might also respond slower (with a different probability) because she delayed answering until she was finished with what she was already working on. At the same time, the agent's knowledge is revised based on the targeted information collected during sensing. Since knowledge state is fully observable, the correctly corresponding observation/state values in the observation function have probability 1 and all other values have probability 0.

Finally, we define the $R(s, a)$ reward function in the POMDP as the value of knowledge refinement optimized in the OETP: $KR(ac_j, s_k, RN, \sigma, K, d_i)$ (Eq. 7). For our example, this could be the increase in the possibility [21] the agent ascribes to the user's correct preference.

Using this approach, we see that building a policy (using any of the approaches briefly introduced in Sect. 2.2.1) for the POMDP optimizes the reward function based on making choices given the estimated sensing state. Since our problem mapping assigns the reward function to be the value of knowledge refinement, following the POMDPs policy thus optimizes the value of knowledge refinement function $KR(ac_j, s_k, RN, \sigma, K, d_i)$ given a current reasoning-level decision d_i from the OETP. Thus, the policy created by the POMDP provides a controller for choosing sensing actions to solve the OETP, which can be either myopic for current refinement or non-myopic for long-term, discounted refinement, depending on the POMDP solver used. This behavior is made possible due to the fact that the sensing state includes resource state, so the reward function explicitly considers the current state of the resource and the action chosen, which is necessary for calculating the expected value of knowledge refinement due to the OE. However, since 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 from Eq. 12 is included, c.f. Sect. 3.3) or following any other optional constraints. Each of these "safeguards" is in place in order to avoid knowledge corruption and unnecessary resource usage. Otherwise, the controller should only follow the policy created by the POMDP.

[3] Please note that these observations might be secondary to the task-level observations used to refine the agent's knowledge that are also produced by the sensing action chosen. Here, in the context of POMDPs, these observations refer to the information gathered by the agent used to estimate hidden state in the POMDP [23].

Of note, an initial belief state of the OE POMDP must be specified. For example, in environments where information about resource state are known a priori (e.g., the resource always starts in a given state), then the belief state can incorporate such information by centering its mass based on what is known about resource state (e.g., starting with a belief of 1.0 on the sensing state with the known initial resource and knowledge states). However, if no such a priori information is known about the resource state, then the belief state can be a uniform prior over all sensing states with the initial (known) knowledge state and all possible resource states.

Finally, we note that for some environments (e.g., energy aware wireless sensor networks, c.f., Sect. 3.2), resource state (e.g., sensor energy levels) might be fully observable to agents and thus a simpler MDP formulation can be used for the OE POMDP. Here, the model remains the same, except the agent does not require state estimation through observations. For the remainder of this section, we assume that resource state is hidden and a POMDP is required.

4.2 Learning a sensing activity controller

When an explicit, parameterized OE POMDP model of the active perception decision process is not provided by the agent designer, an agent must instead *learn* to make sensing action choices. This lack of an a priori model might occur due to insufficient prior knowledge about the domain or because the underlying environment is inherently dynamic and the POMDP model's parameters change over time. For such learning, we turn to the field of RL in general, and partially observable RL PORL algorithms in particular.

Briefly, **RL** (e.g., [22,46]) is a process through which agents learn how to act through feedback from the environment during exploration. RL commonly comes in two forms: (1) **model-based RL**, where the agent first learns an explicit model of the environment (often an MDP or POMDP), then uses that model to determine an optimal action policy, and (2) **model-free RL**, where the agent instead only learns the outcomes of actions dependent on environment states, then chooses actions maximizing its learned rewards. Below we discuss the relative advantages and disadvantages of these two types of approaches for our methodology. Further, when the environment is partially observable, specialized **PORL** algorithms must be used which learn how to estimate environment state based on observations in addition to RLs learning.

Given our assumption that an explicit, parameterized model of the OE POMDP is *not* provided to agents that use stateful resources during sensing, agents can use PORL to learn a controller for choosing sensing actions. For such learning, the agent can either (1) use model-based PORL to actually learn the entire parameterized model for the OE POMDP (e.g., the transition probabilities between resource states and the likelihood of observations about each resource state) then solve the POMDP to generate its controller, or (2) use model-free PORL to learn the controller directly by only learning which actions maximize (immediate or long-term) knowledge revision based on estimates about resource state. For instance, in our intelligent user support application example, model-based PORL would entail learning the likelihoods of user frustration changes and the relationship between user frustration and observations about frustration. It would also include learning how knowledge about the user's preferences is refined based on different levels of user frustration and sensing actions. Model-free learning, on the other hand, would only focus on learning this latter information.

We conjecture that either type of PORL algorithm is acceptable (c.f., Sects. 5 and 6 for experiments evaluating this conjecture), but note that one type might be more appropriate depending on the specific domain and application to which the agent is deployed.

Table 3 OE POMDP solution

1.	Map the OETP to a corresponding OE POMDP which defines resource state and OE behavior
2.	Use a RL/PORL algorithm to learn a controller to solve the POMDP
3.	Build a policy for sensing action selection using the learned controller
4.	Follow the policy to choose sensing actions to gather information
5.	Repeat steps 2-4 until the agent is ready to make a task-level decision

For example, if the environment is very dynamic, using a model-free PORL algorithm might be more appropriate than a model-based PORL algorithm since the parameters learned by the latter might become outdated as the environment changes, decreasing the usefulness of the learned model and potentially leading to improper decisions by the controller. Here, model-free algorithms should better adapt to the dynamic environment since they do not need to "unlearn" as much outdated information. If the environment is static, however, leveraging the additional environment model learned by a model-based PORL algorithm could result in better proactive behavior through considering the probability of each state transition before taking an action and thus achieve potentially higher long term rewards.

However, although the two types of algorithms differ in whether or not they learn a model of the OE POMDP parameters, one requirement is that whatever PORL algorithm is chosen, it *must* learn the reward function $R(s, a)$ which represents the expected value of knowledge refinement of a given sensing action and sensing state. Without learning this reward function, the agent will not be capable of considering the OE (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 PORL algorithm chosen. Concrete examples of how this reinforcement learning process works in the OE POMDP for various PORL algorithms is provided with the descriptions of the experimental setup used to evaluate our solution approach in Sect. 5.2.

4.3 Summary

Now that we have presented the details of our solution, we summarize by looking at it again from a high-level perspective, as we did in Fig. 1. Specifically, we outline the steps required in our solution as an abstract algorithm in Table 3, using the individual components of the methodology described in Sects. 4.1 and 4.2 as configurable pieces of the overall solution.

In the following, we will give an example of Step 1 by describing a mapping from a specific instance of the OETP into an instance of our OE POMDP. Then, we will discuss several RL/PORL algorithms used to learn a controller for the POMDP that are used in Step 2. Finally, we will conduct experiments using the mapping used in Step 1 and chosen RL/PORL algorithms in order to study the OETP and validate our approach.

5 Implementation and experimental setup

In this section, we detail the experimental setup used to investigate the OE and validate our OE POMDP solution methodology for solving the OETP. Specifically, we consider the MineralMiner simulation environment implemented using the Repast Agent Simulation Toolkit [35]. This environment features (1) parameterized stateful resource behavior for investigating a range of OE influence on agent sensing, and (2) both *fully* and *partially observable*

resource state for evaluating our methodology with respect to the agent's ability to perceive the resource state that governs the OE. In the following, we first describe the environment and then detail the instantiation of our solution. Finally, we present the objectives of our experiments, along with the specific parameters defining our experimental setup.

5.1 MineralMiner: a robotic mining simulation

For our experiments, we utilize *MineralMiner*, a modified Tileworld [38] similar to PacketWorld [51] and RockSample [44]. In this environment, an intelligent agent is randomly placed in a 2D grid consisting of mines of various minerals (gold, silver, uranium) and is tasked with sequentially collecting minerals that must be returned to a base according to firm time deadlines. To accomplish these collection tasks, the agent must first find mines containing the appropriate type of mineral requested in its tasks, then drill to collect enough minerals to complete each task. Drilling requires a different action for each mineral type, and choosing the wrong type destroys the mine. Thus, there is a strong need for correctly determining the type of each mine. If an agent fails to accomplish a task before its deadline, that task is discarded and any minerals collected are saved for later tasks.

An agent begins with a map of the location of all mines and the base, but with no a priori knowledge about the contents of those mines. Thus, it must perform sensing activities with a stateful resource (called an *electronic microscope*) to determine mine contents:

(1) *advanced mine test*, which uses a large amount of energy but is highly accurate,
(2) *basic mine test*, which uses a low amount of energy but is not very accurate, and
(3) *wait*, which produces no observations but allows the microscope energy to recharge.

The specific amount of energy required for each test is random up to a maximum value (c.f., Table 7 in Sect. 5.3), representing an uncertain amount of work required to perform a test. Furthermore, the accuracy of each test is affected by (1) the current *energy* level of the microscope, where the microscope's accuracy decreases with energy usage, and (2) the test's *sensitivity* to microscope energy level, where the advanced test is more sensitive to the microscope's status.

Agent knowledge is represented by evidence-based opinions [21] for each possible mineral type in each mine. Each observation from the microscope yields evidence in favor of one mineral type and against the others for the mine tested. Knowledge is refined by counting this evidence for each opinion per mine. To support its drilling decisions for each task, the agent senses a mine until (1) the expectation (Exp) of the top opinion for the mine is above a confidence threshold, indicating the agent has sufficient information for its decision (Eq. 9), or (2) the agent no longer believes it can converge to a confident opinion (explained in the following paragraph)—an additional constraint to the OETP. Here, the Exp of an opinion is:

$$Exp = \frac{b + w * u}{b + d + u} \tag{17}$$

where b and d are the amounts of evidence in favor of and against the opinion, respectively, u is the amount of uncertain evidence (fixed to 1 in our simulations), and w is a weighting factor for u (fixed to 1/3 in our simulations for the three possible mineral types, called relative atomicity in the original paper by [21]). We use a confidence threshold of 0.7 in our experiments to require several agreeing observations before an agent is confident in a mine's contents.

Since the observations produced through sensing are uncertain and noisy and this Exp formula (Eq. 17) suffers from diminishing returns as b and d increase, it is possible that

Table 4 MineralMiner OE POMDP

OE POMDP	MineralMiner characteristics
Stateful resource	Electronic microscope
Sensing states $S = \{\langle R_S, K_S \rangle\}$	R_S = Microscope energy, K_S = # of Previous observations
Sensing actions A	Advanced/basic mine test, wait
Observations Ω	Noisy microscope energy readings
Transition probabilities $T(s, a, s')$	Change in energy from a test or self-recharge during wait, increased number of observations
Observation probabilities $O(s', a, o)$	Probability of noisy energy readings based on actual energy level
Knowledge refinement reward $R(s, a)$	Increase or decrease in the believed possibility of the true mineral in the supply

an agent will never actually converge on a confident opinion about the mineral type inside any particular mine (i.e., never have sufficient information for its drilling decision). Thus, the agent periodically estimates whether or not it believes it can still reach the confidence threshold at its current mine by comparing the knowledge refinement produced in the top opinion for that mine over a previous window of observations (*Sensing Window*) with the amount of refinement still required to reach the confidence threshold. It continues sensing if the previous refinement is more than the amount still necessary, else it goes to an unsensed mine. Agents also move on to another mine if its best action (in terms of expected knowledge refinement) is the wait action and it believes the microscope is fully charged[4]. Otherwise, the agent would be stuck indefinitely since waiting will not change the sensing state, thus the agent will choose to wait repeatedly.

5.2 OE POMDP 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 through the OETP the need for knowledge refinement necessary for accomplishing its current tasks against knowledge corruption due to sensing distortion from the OE. To handle this tradeoff, we model the process of selecting microscope-based sensing actions using our OE POMDP solution methodology, summarized in Table 4.

Within MineralMiner, we consider two types of observability of the microscope energy (resource state): (1) fully observable and (2) partially observable. In the fully observable case, the agent always accurately reads the current amount of energy in the microscope and simply uses an MDP representation of the OE MDP. In the partially observable case, on the other hand, the energy reading is noisy and provides observed values near the true value based on:

$$\phi(r) = \sigma'(r) + \text{uniform}(-dist, dist) \tag{18}$$

where $\text{uniform}(x, y)$ returns a uniformly sampled value in the range $[x, y]$ and $dist$ is a distortion parameter (c.f., Table 6) determining the amount of additive noise in the observation.

[4] However, for the RL and PORL agents described in Sect. 5.2, it is possible that the agent's learning will cause it to choose a different action at the same mine when its energy is full. Thus, after a learning update, we allow these agents to return to such mines.

For knowledge state, on the other hand, we use the number of previous observations for the mine being tested, equal to $b + d$ in Eq. 17 since these combine with the constant prior $u = 1$ to be a weighting factor on the influence of the next update. Thus, they determine how much (or little) knowledge refinement is possible from a sensing action.

Further, we represent the value of knowledge refinement (i.e., reward) as the change in the agent's Exp (Eq. 17) for the correct mineral type in each supply, known after drilling regardless of whether the mine is collected from or destroyed. Thus, knowledge refinement rewards are calculated and agent learning can occur only after each drilling action since the agent must wait for ground truth before calculating knowledge refinement.

Finally, we assume no *a priori* knowledge about the state of the microscope resource. Thus, we start with an initial belief state containing a uniform prior over all possible resource states.

To learn a controller for the OE MDP in the *fully* observable case, we consider three RL algorithms: (1) RMax, (2) Q-Learning, and (3) REINFORCE. For the *partially* observable case, we use two PORL algorithms to learn a controller for the OE POMDP: (1) Bayes-adaptive POMDPs (BAPOMDP), and (2) recurrent policy gradients (RPG). Note that the RL algorithms were chosen due to (1) their similarity to the chosen PORL algorithms, or (2) their popularity. The two PORL algorithms, on the other hand, were chosen because they represent the state-of-the-art in model-based and model-free PORL, respectively. A comparison of the algorithms and the parameters used are provided in Table 5. Next, we only briefly introduce the RL and PORL algorithms and point the reader to their original references for more details.

Of note, for the discrete RL and PORL algorithms (RMax, Q-Learning, and BAPOMDP), we discretize the R_S values into discrete bins of equal size across their range $[0, 100]$, and K_S values are limited to $0, 1, \ldots, |K_S| - 1$ in order to create a finite set, with all values greater than or equal to the final count mapped to the same state[5]. Observations about resource state are discretized in the same manner as R_S. The specific number of states and observations used is given with the RL algorithm parameters in Table 5, with $|S| = |R_S| \cdot |K_S|$ and $|R_S| = |K_S|$. For the continuous RL and PORL algorithms (REINFORCE and RPG), on the other hand, such discretization is not necessary.

5.2.1 Reinforcement learning algorithms

RMax [8] is a popular, polynomial-time, probably approximately correct model-based RL algorithm that uses evidence counting to learn the parameters of the underlying MDP. Specifically, it counts the number of observed transitions between states and fills in part of the state transition function using these counts once their sum from any state exceeds a threshold. To learn rewards, it first assumes a maximal value for all state/action pairs to encourage exploration, then updates values based on the first reward received for the state/action pair. However, since knowledge refinement might not be deterministic for state/action pairs due to the stochastic accuracy of the microscope, we extend the reward update in RMax to follow a similar counting-based learning strategy as learning state transition probabilities in order to learn a stochastic reward function. Our controller learned using RMax solves the MDP using exhaustive search on the Bellman equations (Eqs. 1–2, c.f., Sect. 2.2.1) with a finite horizon (of 5 to provide some look ahead without expanding too far into the future).

Second, Q-Learning [48] is a popular model-free RL algorithm which learns an approximation of the underlying MDPs utility function in tabular form based on rewards received from the environment. Specifically, it maintains a $Q(s, a)$ entry for every state/action pair

[5] Chosen because of diminishing returns on knowledge refinement as the number of observations increases.

Table 5 Reinforcement learning algorithms

Algorithm	RL versus PORL	Model-based (MB) versus model-free (MF)	Description	Parameters
RMax [8]	RL	MB	Tabular count-based learning Learns state transition and reward function	$R_{Max} = 1$ $\|S\| = 100$ $\|R(s, a)\| = 10$ Horizon = 5
Q-Learning [48]	RL	MF	Tabular geometric averaging-based learning Learns reward function	$\alpha = \frac{1}{n}$ $\gamma = 0.3$ $\epsilon = 0.1$ $\|S\| = 100$
REINFORCE [53]	RL	MF	Neural network-based learning Learns stochastic controller and reward function	$\alpha = 0.3$ Hidden nodes = 10
BAPOMDP [40]	PORL	MB	Dirichlet count-based learning Learns probabilities over POMDP models	$\|S\| = 25$ $\|\Omega\| = 5$ $\|R(s, a)\| = 10$
RPG [52]	PORL	MF	LSTM RNN-based learning Learns stochastic controller and reward function	$\alpha = 0.3$ Hidden nodes = 10

representing an approximation of the utility of taking action a in state s. These values are updated whenever an explicit reward is received from the environment for a state/action pair by mixing the old value with the new reward. For the mixture, we use a *discounted learning rate* $1/n$ where n tracks the number of previous updates to the cell in the Q table being updated, which encourages faster convergence of the agent's learning. Our controller uses an ε-greedy strategy, where the agent explores a suboptimal action with probability ε and exploits the action for the current state with the highest $Q(s, a)$ value with probability $1 - \varepsilon$.

Finally, REINFORCE [53] is a model-free RL algorithm using neural networks to simultaneously learn (1) a reward function $R(s, a)$, and (2) a stochastic controller determining the probability the agent should select each action given the state of the environment. The former is learned using traditional backpropagation [42], whereas the latter is learned through eligibility backpropagation [53] based on the received rewards.

5.2.2 PORL algorithms

BAPOMDP [40] is a state-of-the-art model-based PORL approach which is similar to RMax [8] but for partially observable environments. Specifically, this algorithm relies on experience tracking (in Dirichlet distributions) to model the conditional transition and observation probability functions. However, because the resource state is hidden, the entries in these

vectors cannot be counted deterministically. Instead, the agent considers the possibility that a number of possible counts are correct. Here, the states in the BAPOMDP are augmented with the Dirichlet distributions themselves. Thus, the new (larger) POMDP contains multiple possible state/learned model pairs as its states, allowing the agent to consider multiple 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. Our agent learns a controller by first learning the corresponding BAPOMDP model, then solves the POMDP model for each complex state and chooses actions online using 1-step decision trees (c.f., Sect. 2.2.1). Of note, to learn the reward function for use with BAPOMDP, we use a similar experience counting approach as that with RMax (c.f., Sect. 5.2.1). However, since the state of the environment is hidden and estimated by the belief state, we actually increment fractional counts based on the amount of belief for each state, rather than a full 1.0 count for a known state in fully-observable environments.

Additionally, we also employ a state-of-the-art model-free PORL approach called the RPG algorithm [52] which extends REINFORCE [53] to partially observable environments. Specifically, for its neural networks, it trains long short-term memory (LSTM) [15,19] recurrent neural networks (RNNs). These networks can implicitly discover and consider patterns of environment behavior based on observing the results of previous actions, allowing the agent to implicitly estimate hidden state through observations. Training the action controller and reward function neural networks in RPG follows backpropagation-through-time [49] for LSTMs [15,19].

5.3 Experimental setup

5.3.1 Objectives

Within the MineralMiner simulation environment, we conduct experiments to evaluate the use of the OE MDP and POMDP for controlling agent sensing with stateful resources. Specifically, we have two objectives:

> *Sensing performance objective*: Evaluate the **sensing performance** of agents under the OETP.

Specifically, we evaluate the sensing performance of agents suffering from the OETP in both fully and partially observable MineralMiner. This evaluation entails comparing the previously mentioned three RL and two PORL algorithms (Sect. 5.2) for learning to choose sensing actions within the OE MDP and POMDP to solve the OETP against baseline agents which follow sensing policies that do not consider resource state or the OE. These baseline agents are: (1) ***Advanced*** and (2) ***Basic***, where the agent always chooses the advanced and basic mine test (ABT) sensing actions, respectively, and (3) ***Random***, where the agent randomly chooses one of the three sensing actions, including wait. Further, we consider two more elaborate baselines (4) ***OE MDP*** and (5) ***OE POMDP***, which contain a priori models of the OE MDP and POMDP, respectively. OE MDP solves its model in the same manner as RMax (c.f., Sect. 5.2.1), whereas the OE POMDP solves its model in the same manner as BAPOMDP (c.f., Sect. 5.2.2). Further, both use the following structure parameters, similar[6] to the learning approaches (c.f., Table 5): $|S| = 100$ ($|R_S| = |K_S| = 10$), $|\Omega| = 10$ for OE POMDP and $|S| = 400$ ($|R_S| = |K_S| = 20$) for OE MDP. Together, these baselines serve

[6] BAPOMDP only uses $|S| = 25$, $|\Omega| = 5$ rather than 100, 10 due to the high computational complexity of this approach, whereas OE MDP uses twice as many resource and knowledge states due to its low complexity.

as an upper-bound for agent learning and demonstrate the performance of our solution if the application were such that learning would not be required.

To evaluate these approaches, we measure sensing performance as the *average knowledge refinement per sensing activity*. This measure was chosen because it is knowledge refinement that agents aim to optimize in the OE POMDP in order to balance the OETP. In MineralMiner, average knowledge refinement is defined as the average change in the Exp 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, even if the mine is destroyed).

Task performance objective: Evaluate the **task performance** of agents under the OETP.

Second, we evaluate the impact of the OETP on the task performance of agents given its impact of sensing performance in both fully observable and partially observable MineralMiner. This evaluation is motivated by the Performance Hypothesis:

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 motivates 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, especially for bounded rational agents. Without such improvement, there is no need for high quality sensing. Instead, an agent could minimize the amount of sensing performed to minimize costs. While we assume this hypothesis holds in many MAS environments, it might 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 sufficient a priori knowledge about the environment.

To evaluate the Performance Hypothesis in both fully and partially observable MineralMiner, we consider the task performance of agents in each simulation. If sensing performance does lead to improved task performance (confirming the Performance Hypothesis), we should expect to see higher levels of task performance when sensing performance increases. In MineralMiner, task performance is measured as the *total number of tasks completed* by the agent. We chose this measurement because the primary goal of the agent is to complete as many tasks as possible. If sensing performance does lead to improved task performance, we should expect to see more tasks accomplished through drilling more properly identified mines.

5.3.2 Environments

In order to understand the impact of the OE on agent behavior, we consider the following environments in our experiments. Specifically, we use six environments with varying levels of OE in the microscopes, using different amounts of state-dependent accuracy error by varying the noise factor (*NF*):

$$error = \frac{(100 - energy)}{100} * NF * Sen \tag{19}$$

where Sen is the test's sensitivity to the OE and a larger NF produces more error and OE. Thus, using the microscope for sensing decreases its energy, which increases error and reduces sensing accuracy. Here, accuracy is then the test's fully charged accuracy (given in Table 7) minus $error$. For each of the *NF* values considered in our experiments, we present

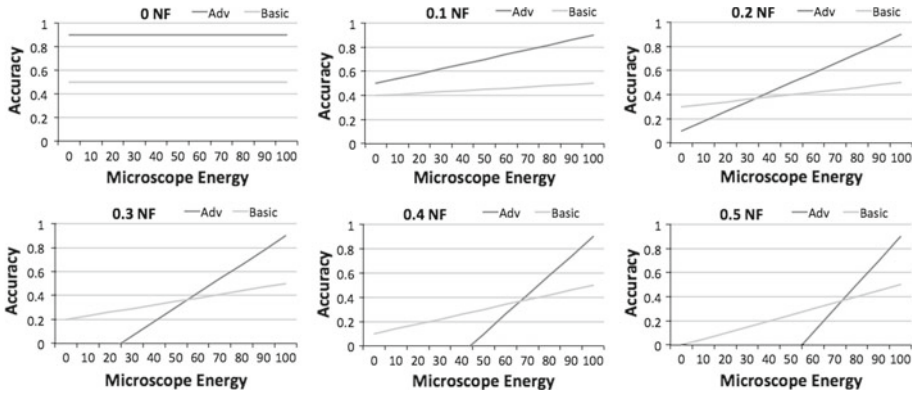

Fig. 2 Microscope test accuracy for various levels of OE

Table 6 MineralMiner experiment parameters

Parameter	Value
Grid size	40×40
#Supplies	600
# Tasks	600
Microscope recharge rate	10%/time unit
PO noise dist	10%
Sensing window	5
NF	0, 0.1, 0.2, 0.3, 0.4, 0.5

Table 7 MineralMiner sensing action parameters

Sensing activity	Max energy drain (%)	Accuracy	Sensitivity
Advanced	40	0.9	4
Basic	20	0.5	1

the accuracy-energy relationship curve due to this error in Fig. 2. These curves show that increasing NF not only produces lower accuracies in the two tests, but also shifts the ranges of energy values where each test is the optimal choice due to their different sensitivities to the OE. For example, for the lowest values of *NF* environments, advanced dominates, whereas for the highest levels, basic is more accurate for most energy values. Therefore, depending on the *NF* value, agents have reason to choose different sensing activities to minimize OEs impact on knowledge refinement.

5.3.3 Parameters

The parameter values to the fully and partially observable MineralMiner simulations important to our experiments are presented in Tables 6, 7. To reduce the variance of the results, we average the results over 30 runs (each with a different random seed).

We chose these parameters for the following reasons. First, we wanted enough learning opportunities to demonstrate the effects of RL/PORL, so we chose 600 mines and tasks (200 mines each of 3 types of minerals). We also wanted sufficient difference in the effect

of the OE on sensing and to provide incentive for agents to choose various sensing activities dependent on microscope energy. Choosing the parameters from Table 7 gave us this behavior, as shown in Fig. 2. We chose a $dist$ value of 10%, which distorts energy observations for partial observability, based on the size of the discretized R_S state space (Table 5) to allow observations above and below the correct state to occur. Finally, we picked a microscope recharge rate near the cost rate of the other sensing actions, and we chose a *Sensing Window* of 5 (c.f., Sect. 5.1) to allow the agent to sense a mine enough times to gather the necessary information but not be stuck forever.

6 Results

In this section, we analyze the results of the experiments outlined in the previous section, used to evaluate our OE POMDP solution in solving the OETP. We begin by evaluating the results of the experiments for the *sensing performance* objective, followed by the results of experiments for the *task performance* objective. For both objectives, we consider the results in both fully observable as well as partially observable MineralMiner. Finally, we conclude the section with a discussion of the results across all experiments, including common trends, differences, and significant discoveries.

6.1 Sensing performance objective

We begin our results analysis by considering the sensing performance of agents suffering from the OETP in both fully and partially observable environments. Recall that these experiments were conducted to determine (1) how the OE impacts the sensing performance of agents using various sensing action selection strategies, and (2) validate that our OE POMDP methodology can handle the OETP and improve agent sensing performance by considering the relationship between OE and resource state during sensing.

To perform our evaluation, we present the sensing performance results of our experiments in Figs. 3 and 4 for the fully and partially observable experiments, respectively. Recall that we measure sensing performance as the average value of knowledge refinement for all observations during agent sensing (including during training for the RL/PORL agents). For both

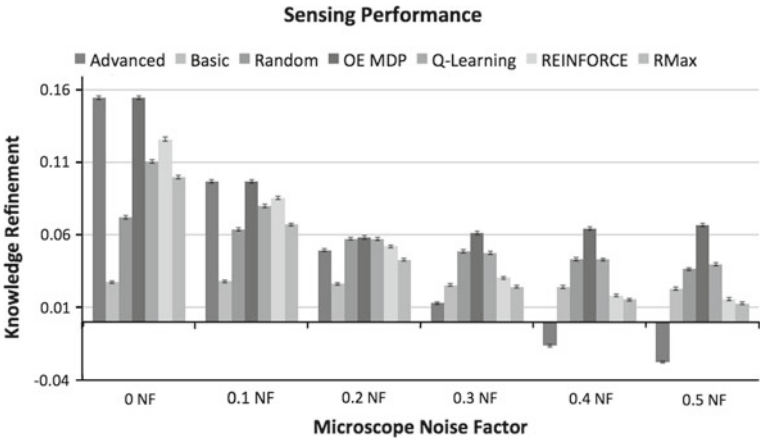

Fig. 3 Sensing performance in fully observable MineralMiner

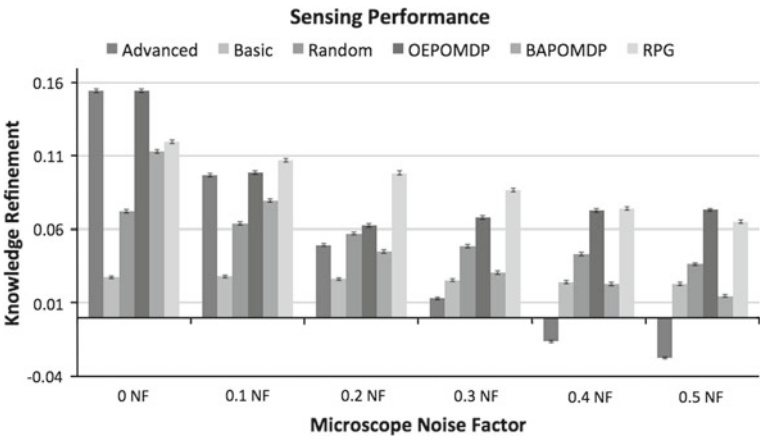

Fig. 4 Sensing performance in partially observable MineralMiner

figures, we also provide error bars representing 95% confidence intervals. From the sensing performance results in Figs. 3 and 4, we observe several important trends.

6.1.1 Impact of OE on sensing performance

First, we note that in both Figs. 3 and 4, as OE (i.e., NF) increased, almost all agents generally achieved lower sensing performance. This demonstrates that the decreased accuracy representing the OE is affecting agent sensing performance as expected. That is, as sensing actions changed the state of the stateful microscope resource used for sensing, sensing accuracy was affected, which in turn reduced the average knowledge refinement produced by noisy information gathered during sensing. Therefore, the OE did impact agent sensing to produce the OETP.

However, we also observe that this trend did not hold for the OE MDP (Fig. 3) and OE POMDP (Fig. 4) approaches. Recall these two agents started with a priori models of the fully and partially observable versions of the OE POMDP methodology, respectively. Here, we note that these approaches actually *improved* their performance from 0.2 NF to 0.5 NF, in spite of the increase in the OE. Upon further investigation, this phenomenon occurred because once the OE was large enough (as reflected in the agent's known reward function), the two a priori model approaches increased the frequency with which they chose the wait action to allow the microscope to recharge before sensing (i.e., improve resource state). This behavior limited the impact of the OE, which only distorted observations when the microscope energy was not full. Looking closer at this behavior, we observe that at lowest levels of OE (NF < 0.2), these approaches did not wait at all since the OE was small enough (0.0, 0.1 NF). This result is shown as the proportion of wait actions Figs. 5b and 6b. Then, for higher levels of OE (NF $= 0.2$–0.5), the OE MDP and POMDP approaches not only started to choose wait, but did so at a progressive rate in response to the increasing difficulty of agent sensing. From these results, we conclude that our OE POMDP methodology properly adapted its sensing behavior depending on the amount of OE in the environment. Specifically, when the OE was minimal, the agent did not worry much about resource state and sensed frequently. Then, when the OE had a larger affect on agent sensing, the agent became more concerned with resource state and acted to improve resource state at the sacrifice of immediate sensing in order to minimize the OEs affect on knowledge refinement.

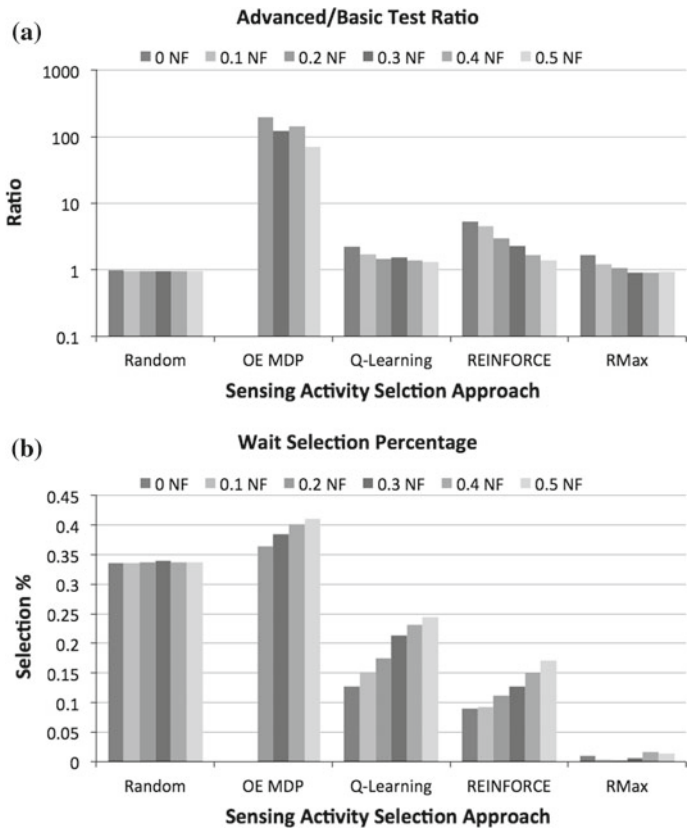

Fig. 5 Sensing activities selected in fully observable MineralMiner **a** ABTs ratio. OE MDP has missing values in **a** because it only chose advanced for 0 and 0.1 NF. **b** Wait selection percentage

Further, we note that how long the agent waited was determined by the discount factor in the expected knowledge refinement calculations (Eq. 1) when building a sensing policy. Thus, for lower levels of OE, sensing before the microscope had been fully charged had higher expected reward than waiting now and receiving higher but discounted future rewards from more accurate sensing. If we were to increase the discount factor to its maximum of 1, we would expect the approach to always wait until the microscope energy is maximal, then perform an advanced test to achieve maximum accuracy and no OE distortion. However, given the task's real-time constraints (i.e., firm deadlines), this could result in too slow of sensing and less task accomplishment, thus smaller discount factors are useful. Overall, this observation shows that while the OE POMDP methodology was capable of adapting to the OE when necessary, it could be adjusted to further adapt sensing behavior to its environment. We intend to investigate such improvements as future work.

6.1.2 Sensing action selection approach comparison

Now that we have established that the OE impacted sensing performance as predicted by the OETP and our OE POMDP methodology was capable of adapting to the OE, we next compare the agents employing our methodology versus the baseline agents. Specifically, we

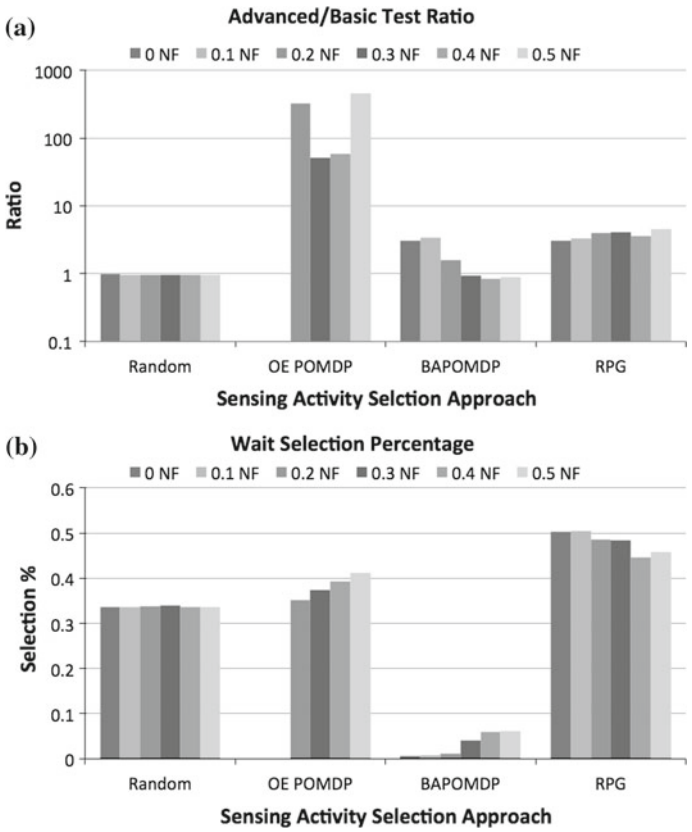

Fig. 6 Sensing activities selected in partially observable MineralMiner **a** ABTs ratio. OE POMDP has missing values in **a** because it only chose advanced for 0 and 0.1 NF. **b** Wait selection percentage

observe that agents using some variant of our methodology had the highest performance in all environments. These include OE MDP and Q-Learning in the fully observable experiments, as well as OE POMDP and RPG in the partially observable environments. This demonstrates that not only was our methodology capable of adapting to the OE to solve the OETP, it also leveraged this adaptation to outperform baseline sensing action selection strategies which do not consider resource state. Therefore, considering resource state was not only important to adapt to the OE, but also to improve sensing performance under the OETP. Further, we observe that agents using our methodology gracefully handled the case where the OE was not present as they still performed well for 0 NF, indicating that our methodology was useful for sensing action selection in environments without the OETP, as well.

Further, we also note some important relationships in the sensing performances of the various approaches. First, the OE MDP and POMDP approaches performed almost identically to the Advanced approach for lowest levels of OE (NF = 0, 0.1) and to the Random approach for NF = 0.2 in terms of sensing performance (Figs. 3, 4). This occurred because the ideal strategy for the lowest OE environments was to simply choose the advanced test because of minimal impact on accuracy from microscope energy levels (Fig. 2), which was followed by both the Advanced approach (by design) and by the OE MDP and POMDP approaches (by maximizing expected knowledge refinement). Further, in the NF = 0.2 environment, the

advanced and basic actions produced very similar accuracy levels for all microscope energy levels (Fig. 2), so randomly choosing between them yielded roughly similar performance as always selecting the most ideal one. Since the Random approach followed such a random selection strategy, along with randomly choosing the wait action to recharge the microscope, Random performed very similarly to the ideal strategy followed by OE MDP and POMDP, which chose the optimal action and began to choose to wait to recharge the microscope to improve accuracy (Figs. 5b, 6b). Then, as the impact of the OE (i.e., NF) increased, both (1) the ABTs became easier to distinguish between again, and (2) the need for more waiting increased, so the OE MDP and POMDP improved their performances, as discussed previously, whereas all other approaches decreased in performance. Finally, we observe that the OE MDP and POMDP approaches performed very similarly to one another for all environments in terms of both behavior and performance, in spite of the differences in their models (e.g., fully vs. partially observable).

Similarly, we note that the Q-Learning and RPG RL/PORL approaches were able to learn how to outperform the non-ideal baselines (Advanced, Basic, and Random) for environments with more OE (NF$=0.2, 0.5$ and >0, respectively). Thus, reinforcement learning did provide a means for learning to improve agent sensing when necessary, even when the agent started with no knowledge about the effects of OE on knowledge refinement. Unexpectedly, the RPG PORL approach also outperformed (on NF$=0$–0.4) the related OE POMDP baseline that started with a priori knowledge about the impact of the OE on agent sensing and knowledge refinement. In the following, we will investigate these results further.

6.1.3 Learning a controller for the OE POMDP

Effects of learning on sensing action selection

Next we consider how learning affected the sensing performance of agents using RL and PORL to solve our OE POMDP methodology. First, we consider the impact of learning on sensing action selection, which in turn leads to various levels of sensing performance. We observe that in the environments with the lowest levels of OE (NF$=0.0, 0.1$), the RL/PORL agents learned to favor the advanced test, as indicated by an ABT ratio (based on the number of times each is selected) of greater than one in Figs. 5a and 6a. This was the appropriate choice because the advanced test achieves a higher accuracy than the basic test for these OE levels (Fig. 2), making the advanced test the optimal choice in all sensing states (also chosen by the OE MDP and POMDP agents). Further, the RL/PORL agents still favored the advanced test for the highest levels of OE (NF$=0.4, 0.5$), but only after learning to wait more frequently (again as done by the a priori OE MDP and POMDP approaches), as shown in Figs. 5a–6b. Thus, the general behavior of the RL/PORL agents was to learn to behave similar to the agents starting with a priori of the OE POMDP, indicating that they learned the appropriate behaviors to handle the OETP while accounting for resource state during sensing. However, we note that the model-based RMax and BAPOMDP approaches did not learn to wait as much as the model-free approaches, which we will discuss in more detail below.

On the other hand, the RL/PORL agents still used the basic test more frequently than the OE MDP and POMDP agents, which was a direct consequence of having to explore all actions in various resource and knowledge states to learn when the advanced test was better than basic. Therefore, exploration during learning affected the quality of the RL/PORL solutions, as expected. Further, the basic test was a better choice than advanced for the highest levels of OE (NF$=0.4, 0.5$) as indicated by its higher accuracy in these environments (Fig. 2),

so the RL/PORL agents were still making good choices, but not necessarily the ideal choices (wait, then use advanced) as frequently as the a priori OE MDP and POMDP agents.

Effects of learning on sensing performance

Taking a closer look at the individual learning algorithms used to learn how to solve our methodology, we next highlight the advantages and disadvantages of each with respect to learning a controller for the OE POMDP to solve the OETP. Here, we go beyond their similarities in learning how to select sensing actions to determine why they exhibited differences in their sensing performances.

Most importantly, we observe that the RPG agent not only outperformed all of the baseline agents unaware of resource state and the OE (Advanced, Basic, and Random), it also learned to achieve higher sensing performance than the OE POMDP approach with an a priori model for all but the highest level of OE (NF = 0–0.4). We believe that this highest level of performance was caused by the implicit non-myopic reasoning of the RPG agent, since RPG trained over *sequences* of observations when learning a controller to choose sensing actions. Thus, the agent considered not only current rewards for sensing, but also future rewards for later sensing actions dependent on the change in resource state from the currently chosen sensing action. Recall from Sect. 5.3 that we restricted OE POMDP to build 1-step decision trees to form a policy for choosing sensing actions. Thus, OE POMDP was myopic. This was done to keep the comparison fair with BAPOMDP, which also solved a (learned) POMDP model using 1-step decision trees—an approach chosen to reduce computational complexity because BAPOMDP must actually solve multiple possible POMDPs (one for each complex state, c.f., Sect. 5.2.2) to choose an action. We postulate that that OE POMDP would have also done better had it used a non-myopic approach for solving the POMDP model, such as larger depth decision trees or approximation techniques such as PBVI [37]. Therefore, we believe that non-myopic solutions could be very important for solving the OE POMDP with or without learning, even if the agent does not *explicitly* consider non-myopic rewards. We intend to further investigate this issue in the future.

However, when the OE was highest (NF = 0.5), we note that the a priori OE POMDP approach achieved the best sensing performance. With so much effect from resource state on agent sensing performance, it appears that optimal performance was more difficult to learn and better knowledge about the OE was more important than non-myopic reasoning.

Further, RPGs performance was a drastic improvement over the fully-observable variant REINFORCE from which the RPG algorithm was extended [52], which did worse than nearly all baselines for non-low levels of OE (NF $>$ 0.1). Specifically, the primary difference between REINFORCE and RPG was also myopic versus non-myopic reasoning: REINFORCE trained only on single instances of sensing outcomes, whereas RPG trained on sequences as stated previously. While this improvement to RPG was originally intended [52] to allow RPG to estimate hidden environment state (i.e., resource state) through observations (i.e., noisy energy readings), it turns out that it also allows the agent to implicitly consider the sequential benefits of action sequences, and thus implicitly consider non-myopic rewards to balance current versus future refinement. This is an advantage over the myopic REINFORCE. Therefore, we have further evidence that non-myopic reasoning is important for solving the OETP.

To better understand the differences in the other learning approaches, we next compare the RL and PORL approaches considered in our experiments by evaluating the impact of their learning on sensing performance while the agents learned how to behave in their environments. Specifically, we consider the difference in sensing performance over time

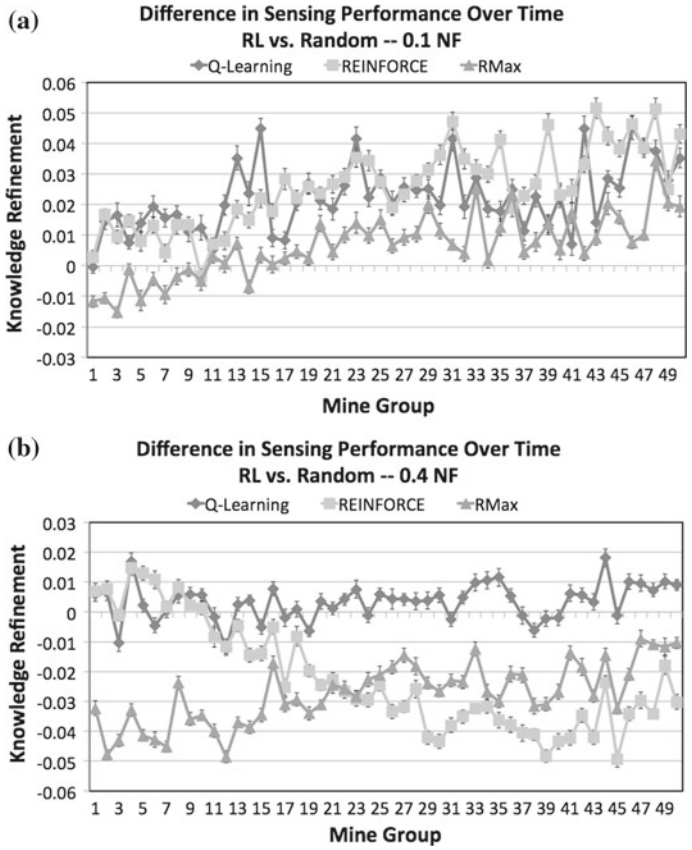

Fig. 7 Differences in sensing performance over time in fully observable MineralMiner **a** 0.1 NF,**b** 0.4 NF

(i.e., successive groups of tested mines) due to learning compared to the Random baseline, shown with 95% confidence intervals in Figs. 7 and 8 using 0.1 NF and 0.4 NF as examples. For our time series, we split observed mines into 50 successive groups of 12 mines each.

In Fig. 7b, we observe that continuous REINFORCE actually learned to sense *worse* over time. This behavior is in contrast to discrete Q-Learning and RMax, which improved over time in both Fig. 7a and b. Upon further investigation, this worsened performance for REINFORCE appears to have been primarily due to a bias towards choosing the advanced test, shown in the higher ABT ratio for REINFORCE at all levels of OE in Fig. 5a (notice this is a logarithmic scale) than the other RL algorithms. We believe this bias was caused by the manner in which REINFORCE learns its reward function. Specifically, this RL approach learned a single function approximator using a neural network that covered all possible state/action pairs. Since the advanced test did achieve higher accuracy (Fig. 2) and thus higher knowledge refinement when the resource state was high, the learned reward function appeared to become biased towards believing advanced would always do well. Q-Learning and RMax, on the other hand, tabularized the reward function based on discrete regions in the state space and thus learned each region independently. Therefore, discretizing the state space appears to have avoided bias from good performance in a subset of sensing states.

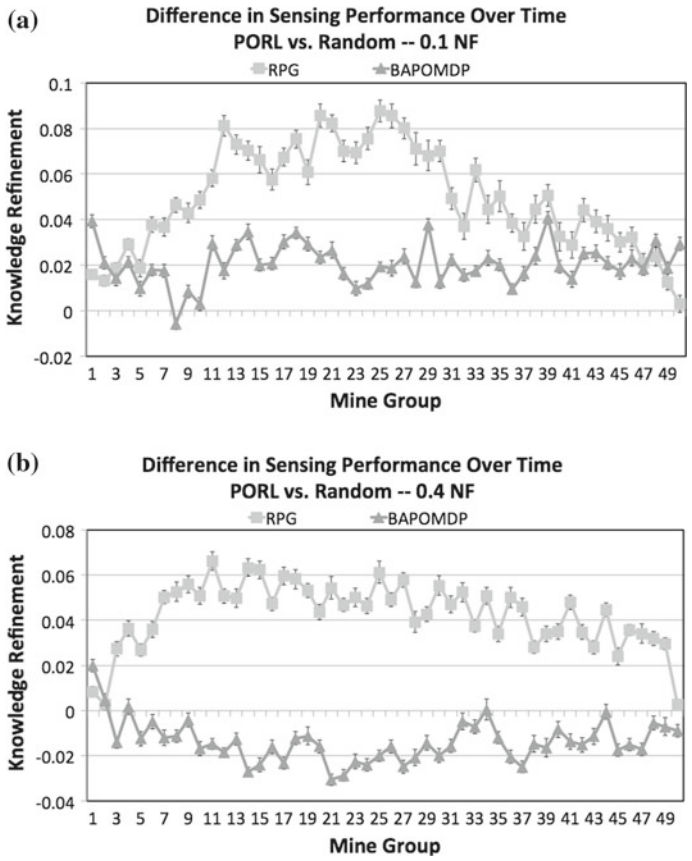

Fig. 8 Differences in sensing performance over time in partially observable MineralMiner **a** 0.1 NF, **b** 0.4 NF

On the other hand, in Fig. 7a we observe that REINFORCE was capable of improving its performance over time. However, we believe this result was a fluke because the bias towards the advanced test happens to be the correct action to choose. Overall, we believe this bias problem with REINFORCE could be overcome by tweaking its learning parameters, such as adding additional hidden nodes to better differentiate resource states to avoid bias or changing its learning rate, which we also intend to explore as future work.

Further, model-free Q-Learning always outperformed model-based RMax, both overall (Fig. 3) and over the time series (Fig. 7). The same held true for the model-free RPG over model-based BAPOMDP (Figs. 4, 8). This success was due to the faster learning performed by the model-free learning algorithms, shown in Figs. 7a–8b, since they did not require time to collect additional information to learn a full model of the environment (i.e., transitions probabilities between sensing states and observation probabilities). Further, we believe that the much lower amount of waiting to recharge the microscope's energy done by the model-based approaches was due to the additional amount of time spent learning the model parameters. Specifically, parameter learning required the agent to use the advanced and basic actions more times to learn appropriate model parameters for those actions. Overall, after comparing model-free and model-based RL/PORL, it appears that knowledge refinement

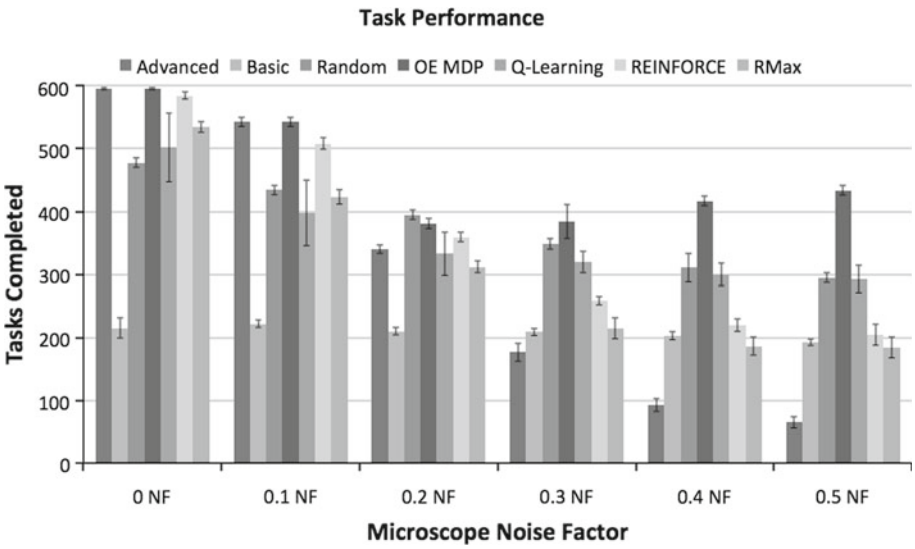

Fig. 9 Task performance in fully observable MineralMiner

reward learning alone was sufficient for improving sensing performance in the OETP and was advantageous due to faster learning.

However, in the fully observable experiments, we observe that the additional model learning performed by RMax could also be beneficial, as shown by its ultimately higher or equal performance towards the end of the simulations in Fig. 7a and continued increasing trend in Fig. 7b, compared to the performance of Q-Learning. Thus, if the application includes enough learning episodes to overcome the slower learning rate of model-based RL, the additional information learned might result in even better sensing performance and an even better solution to the OETP than model-free RL. In the partially observable experiments, on the other hand, learning a model of the (more complex) environment proved to be too difficult as BAPOMDP never outperformed RPG. Thus, the additional complexities of such learning were not worth the trouble.

Finally, we note that RPG appeared to actually peak early and then decreased in performance over time. Thus, RPG appears to have overfitted from its experience. Therefore, although RPG was one of the top performers when the OE was present, this approach could actually be improved to further boost sensing performance. This overfitting problem could be solved by either (1) reducing the learning rate used (Table 5, c.f., Sect. 5.2), or (2) simply stop learning after a fixed number of episodes. Overall, this further implies (as seen in Sect. 6.1.1) that although off-the-shelf settings did reasonably well, the OETP is sufficiently complex enough to require parameter tweaking to adapt our solution to the specific environment in order to achieve optimal sensing performance.

6.2 Task performance objective

Now that we have a good understanding of the sensing performances of the various approaches considered when the OETP is present, we next analyze the task performance results of the agents in both fully and partially observable MineralMiner. Our goal here is to shed light

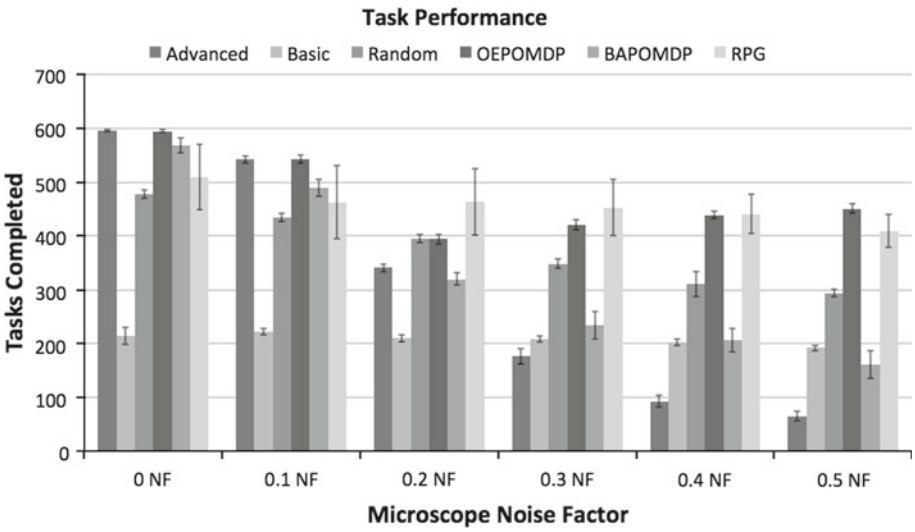

Fig. 10 Task performance in partially observable MineralMiner

on how sensing performance influences task performance. Recall that task performance is measured as the number of tasks completed by the agents. We present these task performance results for our fully and partially observable experiments in Figs. 9 and 10, again with 95% confidence intervals. From these results, we again observe several important trends.

6.2.1 Impact of the OE on task performance

Similar to what we observed with sensing performance in Figs. 3 and 4, Figs. 9 and 10 show that in almost all cases, as the OE (i.e., NF) increased, task performance decreased or remained similar. Thus, the OE also affected agent task accomplishment through its impact on agent sensing, making it *a valid concern for agent developers beyond its impact on agent sensing*. However, as with sensing performance, the task performance of the OE MDP and POMDP approaches improved from 0.2 to 0.5 NF, again due to improved sensing action selection in response to increasing OE.

6.2.2 Relationship between sensing and task performance

We further observe that task performance generally followed the same trends observed for sensing performance in Figs. 3 and 4 (c.f., Sects. 6.1.1 and 6.1.2). In almost all cases, agents can be ranked with respect to task performance as they were for sensing performance. Furthermore, the relative differences between their task performances appear to be proportional to their differences in sensing performance. Thus, for any fixed amount of OE, we know: (1) sensing and task performance both changed together in the same direction with similar magnitude, (2) the only variable changed was the sensing action selection approach used, (3) changing the approach is known to change sensing performance, and (4) the approaches only consider sensing and do not consider task performance. Therefore, the *only cause for improved task performance was improved sensing performance, validating the Performance Hypothesis*. Further, this result demonstrates that our solution methodology was valuable for improving not only the agent's sensing performance, but also its ability to accomplish tasks.

Table 8 Correlation between sensing and task performance in fully observable MineralMiner

Approach	0 NF	0.1 NF	0.2 NF	0.3 NF	0.4 NF	0.5 NF
Advanced	−0.1215	0.3255	0.6903	0.8906	0.8846	0.9380
Basic	0.8620	0.6452	0.6998	0.6154	0.5004	0.4872
Random	0.4762	0.7066	0.7115	0.7258	0.9753	0.7981
OE MDP	−0.1215	0.3255	0.5914	0.9489	0.6164	0.6307
Q-Learning	0.3343	−0.5042	−0.2865	−0.0838	0.1260	0.9705
REINFORCE	0.6373	0.5405	0.5975	0.6931	0.8011	0.6220
RMax	0.5981	0.8449	0.8524	0.9658	0.9042	0.9422
Total	0.8457	0.7971	0.7015	0.9148	0.9522	0.9639

Table 9 Correlation between sensing and task performance in partially observable MineralMiner

Approach	0 NF	0.1 NF	0.2 NF	0.3 NF	0.4 NF	0.5 NF
Advanced	−0.1215	0.3255	0.6903	0.8906	0.8846	0.9380
Basic	0.8620	0.6452	0.6998	0.6154	0.5004	0.4872
Random	0.4762	0.7066	0.7115	0.7258	0.9753	0.7981
OEPOMDP	0.0445	0.4883	0.8044	0.6157	0.5759	0.6243
BAPOMDP	0.6985	0.8225	0.8853	0.5460	0.6198	0.1759
RPG	0.7479	0.6642	0.6790	0.8218	0.8886	0.6667
Total	0.8744	0.8235	0.8235	0.9186	0.9563	0.9308

To further investigate the relationship between sensing and task performance, we look at the correlation between these two values computed over each individual simulation run (i.e., each random seed). These results are presented in Tables 8 and 9 for the fully and partially observable experiments.

Considering these results, we observe that many of the correlation values are high and positive, confirming a strong relationship between sensing and task performance in this application. Since these strong relationships were positive, we conclude that this relationship can be exploited to improve agent task performance by simply focusing on improving agent sensing. Further, this relationship was generally strongest for high levels of OE (0.4, 0.5 NF), demonstrating that when the OE was most prevalent, not only did solving the OETP become necessary to achieve good sensing, but also to accomplish the agent's tasks. Therefore, the OE is an important challenge to consider not only from the perspective of agent sensing, but also overall agent task behavior.

Further, we observe a *stronger* positive correlation between sensing and task performance in the partially observable experiments. We believe that this stronger relationship was caused by *an increased need for sensing in the partially observable environment*: because resource state was hidden, an agent had to rely on sensing actions to not only refine its knowledge, but also gather information about its resource to determine how to properly choose its next sensing action. Without such information, the agent could not properly choose sensing activities, which would have led to worse knowledge refinement and fewer tasks accomplished. Thus, proper sensing action selection was even more important when resource state is hidden from the agent, and our OE POMDP with or without PORL satisfied this need.

However, we also observe that Q-Learning did not experience the same general trends as the other agents. Instead, for low amounts of OE (0, 0.1 NF) it achieved similar or worse task performance than Random, in spite of sensing better, and the relative gap between the task performance (Fig. 9) of Q-Learning against REINFORCE and RMax was greater than the relative gap in sensing performance (Fig. 3). Further, Q-Learning commonly achieved negative correlations between sensing and task performance. We explain the cause of these problems below.

6.2.3 Effects of learning about sensing performance on task performance

Now that we have explored the general relationship between sensing and task performance, we take a look at how the individual learning algorithms used by our agents impacted task performance. Here, we compare the cumulative task performance of the various RL/PORL agents over time with 95% confidence intervals to show the impact of their learning about sensing on their ability to accomplish tasks. Similar to our earlier sensing performance analysis (c.f., Sect. 6.1.3), we split tasks into successive 50 groups of 12 tasks each for our time series. Again, we use the difference between each RL/PORL agent and the Random agent for this comparison to show the explicit impact of learning. We present the results of the 0.1 and 0.4 NF environments as examples in Figs. 11 and 12.

First, we observe that the Random and RL/PORL approaches (except Q-Learning and RPG, see below) all performed very similarly at the beginning of the experiments. Specifically, the agents performed every task in a task group at the beginning. The differences in their performance at the end of the simulations, on the other hand, were caused by how long the agent could continue collecting minerals from mines. These behaviors were due to the sensing performed by the agents. Since there was not a one-to-one correspondence between tasks and mines, agents searched the entire grid for mines until they found the minerals necessary for tasks. At the beginning of the simulations, there were a sufficient number of mines to find enough minerals to collect in order to accomplish early tasks. However, as the agent continued to sense and perform tasks, it encountered two problems. One, the agent depleted the contents of mines, or destroyed their contents due to incorrect drilling actions caused by bad sensing. Two, the agent could not converge to a confident opinion on some mines, and then moved on to other mines or gave up sensing altogether. Thus, at the end of simulations, there were no longer any mines to sense or drill to collect minerals for tasks. Therefore, the key to accomplishing tasks in MineralMiner was to be able to drill the longest without destroying mines, an ability supported by converging to the greatest number of correct, confident opinions about mine contents. From another perspective, such convergence was necessary for providing the agent with confidence that it could complete its tasks (as there could be sufficient untapped minerals to complete its tasks but it lacked confidence in its knowledge about their location). Since the number of mines was constant in our experiments, agents with the best sensing over time should have reached such convergence the most often and thus accomplished the most tasks. Therefore, our time series results demonstrate that learning that produces better sensing can also lead to longer task accomplishment, which also leads us to confirm the Performance Hypothesis.

Further, this impact from the OE on the agent's task performance actually indirectly affects the agent's *learning* as well. Specifically, recall that the agent must drill to discover the ground truth of a mine which can only occur once it has converged on an opinion about a particular mineral type at that mine with high Exp. Thus, when reduced sensing performance from the OE prevented convergence and drilling, the agent could not learn from its experiences in order to improve its sensing to overcome the OE. Therefore, such a negative impact on learn-

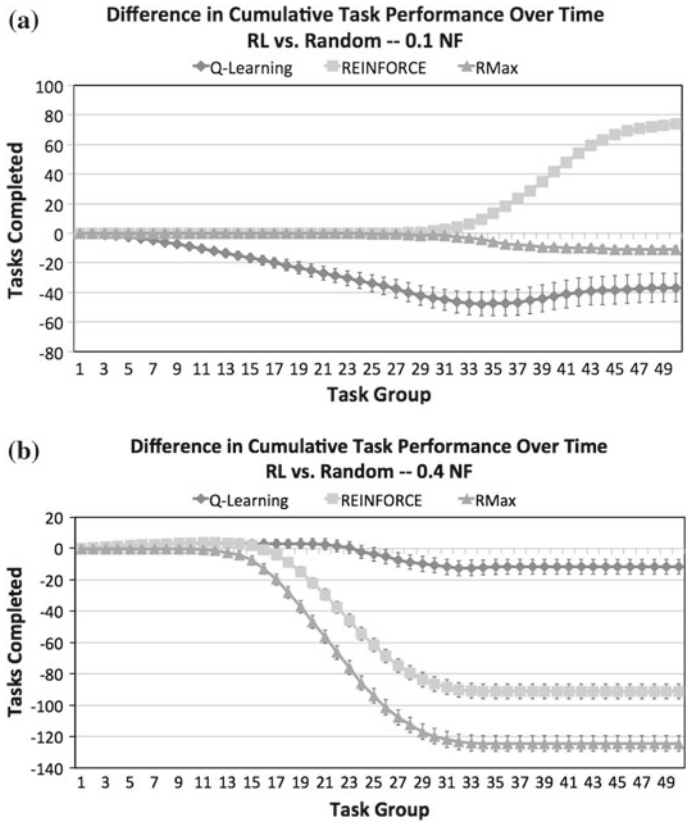

Fig. 11 Differences in task performance over time in fully observable MineralMiner **a** 0.1 NF, **b** 0.4 NF

ing from the OE could hold for any environment where the agent's learning about sensing requires feedback based on the ground truth of sensing, which is not available until a task is accomplished.

Finally we note that the Q-Learning and RPG agents did not appear to complete as many tasks at the beginning of the simulations (Figs. 11a, 12a), in spite of the fact that there were a sufficient number of mines from which to collect. Upon further investigation, this behavior for Q-Learning was caused by the agent actually learning to *stop sensing* in several experiment runs due its fast learning rate and state discretization. Specifically, a rare event described below caused the agent to believe that it could not achieve any knowledge revision. Although infrequent, this phenomenon was certainly problematic and sheds light on an interesting challenge to solving the OETP through learning.

Specifically, this unexpected Q-Learning behavior was due to the agent quickly learning the best action for sensing a mine a second time with full energy was the wait action. This learning was in response to previous inaccurate sensing outcomes for both the ABTs in this sensing state. As described in Sect. 5.1, an agent moved on to another mine if it selected the wait action when it believed that the microscope was fully charged to avoid waiting at a mine forever. Since Q-Learning discretized the state space, it considered a range of energy levels to represent a fully charged microscope. Thus, it was more likely to experience knowledge corruption (i.e., negative refinement) through inaccurate sensing during its wider range of

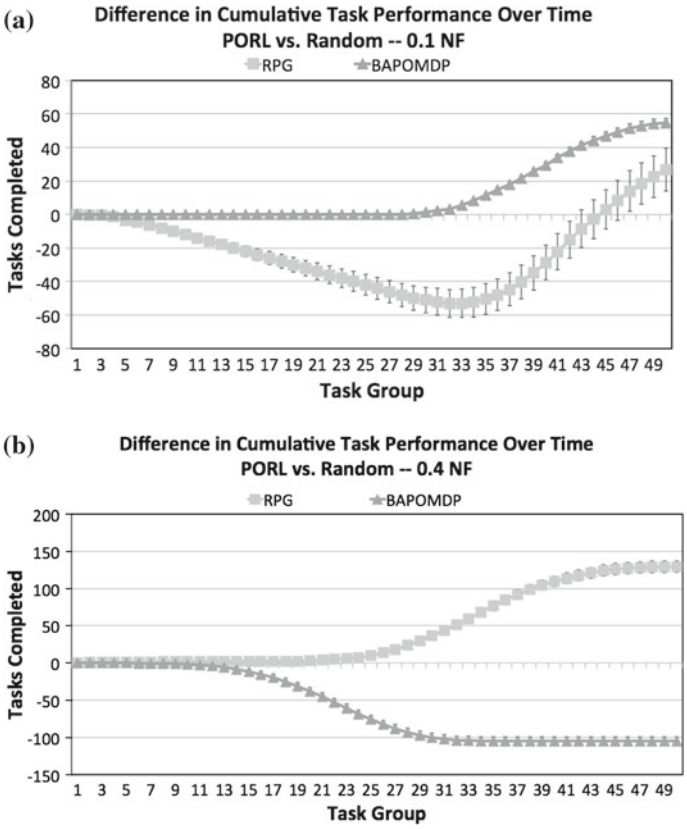

Fig. 12 Differences in task performance over time in partially observable MineralMiner **a** 0.1 NF, **b** 0.4 NF

what it believed to be a fully charged microscope. Further, its fast convergence due to the chosen discounted learning rate led it to over generalize its early experiences and assume future bad knowledge refinement from both the ABTs without additional experience in this sensing state. Thus, Q-Learning in our experiments was too quick to accept that a good sensing state (high energy) led to bad sensing. The other learning algorithms on the other hand, avoided this problem through (1) continuous sensing states, or (2) slower learning. Therefore, we believe that these problems with Q-Learning could be overcome by either (1) increasing the state space to result in smaller discretization, reducing the likelihood of bad sensing outcomes with what is believed to be full energy, or (2) reducing the learning rate to allow for additional exploration.

Like Q-Learning, the RPG agent also tended to achieve lower task performance early on than the Random and BAPOMDP agents. Since RPG also learned very rapidly (Fig. 8), this result provides further evidence that over generalizing early experiences (c.f., Sect. 6.1.3) deteriorated not only sensing performance, but also the task performance of the agent. Thus, between Q-Learning and RPG, we have further evidence that parameter tuning in the complex OETP is needed not only for improved sensing performance, but also task performance.

6.3 Discussion

Finally, we summarize our results and highlight important conclusions by comparing and contrasting the results of the fully and partially observable MineralMiner experiments across various topics of analysis. First, we evaluate the impact of the OE on agent sensing performance based on our results. In both the fully and partially observable experiments, we observed that our OE POMDP methodology using RL and PORL led to improved sensing performance by agents over time. Furthermore, as the OE increased, our Q-Learning RL and RPG PORL agents outperformed other approaches that do not consider resource state or OE during sensing. Further, the OE MDP and POMDP baseline approaches with a priori models also performed the best of all approaches (except related RPG) in all levels of OE by adapting action selection properly to the amount of OE in the environment. Therefore, overall our results provide strong evidence both (1) demonstrating that considering resource state is beneficial in accounting for the OE to solve the OETP, and (2) validating that our methodology is one successful approach to solving the OETP.

Second, we evaluate the impact of the OE on task performance based on our results. In both the fully and partially observable experiments, we observed that higher sensing performance almost always led directly to improved task performance. We also observed high, positive levels of correlation between sensing and task performance, indicating a strong relationship worth exploiting. Thus, we confirmed the Performance Hypothesis for both environments, indicating that research on improving sensing performance is valuable to overall agent research.

Third, we found many impacts of the OE on the agent and its behavior in both simulation environments. First, we observed that the OE impacted not only agent sensing performance, but also agent *knowledge* through lack of confidence in its beliefs. Furthermore, the OE impacted *task performance* not only through reduced task accomplishment, but also through reduced opportunities to accomplish tasks due to the lack of knowledge confidence. Finally, the OE also impacted the *learning* of the agent through a reduced number of learning opportunities by preventing the agent from discovering the ground truth of its prior sensing. Overall, these results demonstrate the importance of considering the OE beyond just its impact on sensing, especially in real-world environments where tasks are already difficult to achieve and learning is difficult to perform.

Fourth, we also discovered the *robustness* of our methodology. Specifically, we observed in both the fully and partially observable results that our methodology performed (nearly) as well as the best other approaches when the OE was not present, in spite of the fact that our methodology assumed its presence. Thus, our solution provides a generally good way of improving sensing.

Finally, we also uncovered many advantages and disadvantages to the various types of learning employed by our methodology. In the fully observable experiments, we observed that discrete algorithms were not biased by some successful sensing outcomes, unlike the continuous algorithm whose performance was hampered through a bias towards sensing actions that provided good knowledge refinement in only some sensing states. In the both the fully and partially observable experiments, we also observed that model-free learning generally outperformed model-based learning overall due to the additional time costs of learning additional state transition (and observation in PORL) parameters. We also observed evidence that over generalizing and overfitting can occur if learning parameters are not tuned properly due to the complexity of the OETP. Therefore, we now have a better idea of what types of learning algorithms are best for solving the OETP through the OE POMDP.

7 Conclusions and future work

In conclusion, we have introduced the OETP in this paper, a subproblem of limited resource use during sensing. This problem is the result of using *stateful* resources during agent sensing which can produce an OE. This OE represents a distortion in sensing outcomes caused by changes in resource behavior from resource usage during sensing. Specifically, the OETP requires an agent to balance satisfying the need for (1) knowledge refinement to support agent reasoning, and (2) avoiding knowledge corruption through OE distorted sensing. The stateful resource categorization, OE, and OETP are all novel in multiagent systems research.

To solve the OETP, we proposed a novel decision theoretic solution called the OE POMDP which models the active perception sequential decision process of selecting sensing actions that change resource state and produce knowledge refinement as a POMDP. This approach follows from our mathematical formulation of the OETP. Since a priori knowledge about the relationship between resource and knowledge state, sensing activities, and knowledge revision is generally not available to agent developers to embed within their agents, our methodology uses RL to solve the OE POMDP in both fully and partially observable environments.

To investigate the OE and validate our methodology, we conducted experiments in MineralMiner, a Tileworld [38] variant where an intelligent agent searches for mines containing rare minerals in an unexplored space. In our experimental results considering both fully and partially observable resource state, we discovered that our methodology (1) learned to improve the knowledge refinement provided by sensing, (2) balanced current versus future refinement to improve long-term behavior, and (3) exploited the relationship between sensing and task performance to boost task performance by improving sensing. Further, our solution still achieved good knowledge refinement even when the OE was not present, demonstrating that it can improve sensing performance in a wide variety of environments. Additionally, we also compared various RL/PORL algorithms to explore their advantages and disadvantages for solving the OE POMDP, finding that discrete algorithms performed better overall than continuous in the fully observable experiments, whereas model-free generally outperformed model-based regardless of observability. Thus, now that we have explored the OETP, we have (1) better insights into this challenging and important problem, (2) an effective solution with ideas for further improvement, and (3) a test bed environment for further research. Overall, this is key contribution to better understanding agent sensing in real-world environments with complex challenges where the OE is most likely to be present.

Based on this research, we have identified several avenues for future work. First, we plan to conduct further experiments proposed in the results analysis (Sect. 6) to improve our methodology and better understand the OE and its influence on knowledge refinement. Specifically, we are most interested in studying how the planning horizon and myopic versus non-myopic reasoning affect the ability of the agent to solve the OETP through our POMDP-based methodology since we have observed evidence that this solution characteristic appears to play an important role in the ability of the agent to manage resource state. Second, we also want to incorporate other areas of recent research into our methodology, including factoring in natural changes in resource state not influenced by agent actions, similar to work by [13], which modeled changes in resource quantity (but not behavior) using MDPs and competitive game theory in the context of resource harvesting in environmental sustainability. Next, we also want to extend our work to more multiagent cooperative sensing control, including investigating the potential use of decentralized MDP/POMDP models [6] for our methodology. Finally, we wish to move our research out of simulation and into real-world applications, including avoiding and mitigating user frustration during human-agent

interactions in an intelligent user interface supporting collaborating researchers in the Biofinity Project (http://biofinity.unl.edu).

Acknowledgments We would like to thank the anonymous reviewers for their constructive feedback and helpful suggestions, which were used to improve the presentation of this research. This research was supported by a grant from the Department of Education (grant P200A040150), an NSF grant (DBI-0743783), and a NSF Graduate Research Fellowship (grant DGE-054850). This work was completed utilizing the Holland Computing Center of the University of Nebraska.

References

1. Adamczyk, P. D., & Bailey, B. P. (2004). If not now, when? The effects of interruption at different moments within task execution. In *Proc. of CHI'04*, Vienna, Austria, April 24–29 (pp. 271–278).
2. Adomavicius, G., & 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.
3. Akyildiz, I. F., Pompili, D., & Melodia, T. (2005). Underwater acoustic sensor networks: Research challenges. *Ad Hoc Networks, 3*(3), 257–279.
4. Araya-Lopez, M., Buffet, O., Thomas, V., & Charpillet, F. (2010). A POMDP extension with belief-dependent rewards. In *Proc. of NIPS'10*.
5. Arisha, K., Youssef, M., & Younis, M. (2002). Energy-aware TDMA-based MAC for sensor networks. In R. Karri & D. Goodman (Eds.), *System-level power optimization for wireless multimedia communication* (pp. 21–40). Norwell, MA: Kluwer Academic Publishers.
6. 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.
7. Boutilier, C. (2002). A POMDP formulation of preference elicitation problems. In *Proc. of AAAI'02* (pp. 239–246).
8. Brafman, R. I., & Tennenholtz, M. (2002). R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research, 3*, 213–231.
9. Casper, J., & Murphy, R. R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on SMC Part B: Cybernetics, 33*(3), 367–385.
10. Chalupsky, H., et al. (2001). Electric Elves: Applying agent technology to support human organizations. In *Proc. of IAAI'01*, Seattle, WA, August 7–9 (pp. 51–58).
11. Cox, M. T., & Raja, A. (2011). Metareasoning: An introduction. In M. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking* (pp. 3–14). Cambridge, MA: MIT Press.
12. Doshi, F., & Roy, N. (2008). The permutable POMDP: Fast solutions to POMDPs for preference elicitation. *Proc. of AAMAS'08* (pp. 493–500).
13. Ermon, S., et al. (2010). Playing games against nature: optimal policies for renewable resource allocation. In *Proc. of UAI'10*.
14. Fowler, H. J., & Leland, W. E. (1991). Local area network traffic characteristics, with implications for broadband network congestion management. *IEEE Journal on Selected Areas of Communications, 9*(7), 1139–1149.
15. Gers, F. A., Schmidhuber, J., & Cummins, J. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), 2451–2471.
16. Grass, J., & Zilberstein, S. (1997). Value-driven information gathering. In *Proc. of AAAI workshop on building resource-bounded reasoning systems*.
17. Grass, J., & Zilberstein, S. (2000). A value-driven system for autonomous information gathering. *Journal of Intelligent Information Systems, 14*, 5–27.
18. Guo, A. (2003). Decision-theoretic active sensing for autonomous agents. In *Proc. of AAMAS'03* (pp. 1002–1003).
19. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*, 1735–1780.
20. Hoey, J., et al. (2007). Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *Proc. of ICVS'07*.
21. Josang, A. (2001). A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 9*, 279–311.
22. Kaelbling, L. P., Littman, M. L., & Moore, W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research, 4*, 237–285.

23. Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101*, 99–134.
24. Khandaker, N., Soh, L.-K., Miller, L. D., Eck, A., & Jiang, H. (2011). Lessons learned from comprehensive deployments of multiagent CSCL applications I-MINDS and ClassroomWiki. *IEEE Transactions on Learning Technologies, 4*(1), 47–58.
25. Klein, J., Moon, Y., & Picard, R. W. (2002). This computer responds to user frustration: Theory, design, and results. *Interacting with Computers, 14*, 119–140.
26. Krause, A., & Guestrin, C. (2005). Optimal nonmyopic value of information in graphical models—efficient algorithms and theoretical limits. In *Proc. of IJCAI'05* (pp. 1339–1345).
27. Krause, A., & Guestrin, C. (2007). Near-optimal observation selection using submodular functions. In *Proc. of AAAI'07*.
28. Krause, A., & Guestrin, C. (2009). Optimizing sensing: From water to the web. *IEEE Computer, 42*(8), 38–45.
29. Krause, A., et al. (2008). Robust submodular observation selection. *Journal of Machine Learning Research, 9*, 2761–2801.
30. Landeldt, B., Sookavantana, P., & Seneviratne, A. (2000). The case for a hybrid passive/active network monitoring scheme in the wireless Internet. In *Proc. of ICON'00* (pp. 139–143).
31. Lesser, V., et al. (2000). BIG: An agent for resource-bounded information gathering and decision making. *Artificial Intelligence, 118*, 197–244.
32. Mark, G., Gudith, D., & Klocke, U. (2008). The cost of interrupted work: More speed and stress. In *Proc. of CHI'08* (pp. 107–110).
33. Monostori, L., Vancza, J., & Kumara, S. R. T. (2006). Agent-based systems for manufacturing. *CIRP Annals: Manufacturing Technology, 55*(2), 697–720.
34. Myers, K. L., et al. (2007). An intelligent personal assistant for task and time management. *AI Magazine, 28*(2), 47–61.
35. North, M. J., Collier, N. T., & Vos, J. R. (2006). Experiences creating three implementations of the Repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation, 16*, 1–25.
36. Padhy, P., Dash, R. K., Martinez, K., & Jennings, N. R. (2006). A utility-based sensing and communication model for a glacial sensor network. In *Proc AAMAS'06*, Hakodate, Japan, May 8–12 (pp. 1353–1360).
37. Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. of IJCAI'03* (pp. 1025–1032).
38. Pollack, M. E., & Ringuette, M. (1990). Introducing the tileworld: Experimentally evaluating agent architectures. In *Proc. of AAAI'90* (pp. 183–189).
39. Raja, A., & Lesser, V. (2007). A framework for meta-level control in multi-agent systems. *JAAMAS, 15*, 147–196.
40. Ross, S., Chaib-draa, B., & Pineau, J. (2007). Bayes-adaptive POMDPs. In *Proc. of NIPS'07*.
41. Ross, S., Pineau, J., Paquet, S., & Chaib-draa, B. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research, 32*, 663–704.
42. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propogation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: explorations in the microstructure of cognitions* (Vol. 1, pp. 318–362). Cambridge, MA: MIT Press.
43. Shah, R. C., & Rabaey, J. M. (2002). Energy aware routing for low energy ad hoc sensor networks. In *Proc. of WCNC'02*, March 17–21 (pp. 350–355).
44. Smith, T., & Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *Proc. UAI'04* (pp. 520–527).
45. Spaan, M. T. J. (2008). Cooperative active perception using POMDPs. In *AAAI 2008 workshop on advancements in POMDP solvers*.
46. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
47. The Biofinity Project. (2010). Retrieved March 7, 2011, from http://biofinity.unl.edu.
48. Watkins, C. J. (1989). Learning from delayed rewards. PhD Thesis, Cambridge University.
49. Werbos, P. J. (1990). Backpropogation through time: What it does and how to do it. *Proceedings of the IEEE, 78*(10), 1550–1560.
50. Weyns, D., Steegmans, E., & Holvoet, T. (2004). Towards active perception in situated multi-agent systems. *Applied Artificial Intelligence, 18*, 867–883.
51. Weyns, D., Helleboogh, A., & Holvoet, T. (2005). The packet-world: A test bed for investigating situated multi-agent systems. In R. Unland, M. Klusch, & M. Calisti (Eds.), *Software agent-based applications, platforms, and development kits* (pp. 383–408).

52. Wierstra, D., Foerster, A., Peters, J., & Schmidhuber, J. (2007). Solving deep memory POMDPs with recurrent policy gradients. In *Proc. of ICANN'07* (pp. 697-706).
53. Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*, 229–256.
54. Williams, J. D., & Young, S. (2007). Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language, 21*, 393–422.
55. Yorke-Smith, N., Saddati, S., Meyers, K. L., & Morley, D. N. (2009). Like an intuitive and courteous butler: A proactive personal agent for task management. In *Proc. of AAMAS'09*, Budapest, Hungary, May 13–15 (pp. 337–344).
56. Zilberstein, S. (1996). Resource-bounded sensing and planning in autonomous systems. *Autonomous Robots, 3*, 31–48.
57. Zilberstein, S. (2011). Metareasoning and bounded rationality. In M. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking* (pp. 27–40). Cambridge, MA: MIT Press.
58. Zilberstein, S., & Russell, S. J. (1993). Anytime sensing, planning, and action: A practical model for robot control. *Proc. of IJCAI'93* (pp. 1402–1407).