Rover Science Autonomy: Probabilistic Planning for Science-Aware Exploration

Thesis Proposal
Trey Smith
Carnegie Mellon University

Abstract

Next-generation Mars rovers will have the ability to autonomously navigate for distances of kilometers. At these scales, a day's traverse takes the rover over its local horizon, into regions where only low-resolution orbital data is available. This improved navigation provides both an opportunity and a challenge: we need new techniques for performing effective science while over the horizon and out of contact.

This work deals with *science autonomy* (SA), the ability of the rover to reason about science goals and the science data it collects in order to make more effective exploration decisions. The proposed work has two major components.

First, we will define a set of rover SA operational modes, and assist in developing and field testing a system that implements these modes. We will create an overall architecture for the SA system, and develop the planning component in that architecture. Research questions include: How do we define the SA operational modes? How useful is each mode, and under what circumstances should it be used? What kind of planner is most appropriate for an SA system?

Second, we will extend partially observable Markov decision process (POMDP) planning algorithms in ways that bridge the gap between the current state-of-art and the planning requirements of the SA domain. POMDP planners can generate high-quality plans that take into account action and sensing uncertainty, but realistic problems in the SA domain are far beyond the abilities of existing algorithms. Research questions include: Can heuristic search be combined with efficient representations of the POMDP value function to speed planning? Can we improve scalability when most of the world state is observable? Can we effectively use continuous planning techniques in the POMDP context?

1 Introduction

Next-generation Mars rovers will have the ability to autonomously navigate for distances of kilometers. At these scales, a day's traverse takes the rover over its local horizon, into regions where only low-resolution orbital data is available. This improved navigation provides both an opportunity and a challenge: we need new techniques for performing effective science while over the horizon and out of contact.

This work deals with *science autonomy* (SA), the ability of the rover to reason about science goals and the science data it collects in order to make more effective exploration decisions.

SA extends the set of operational modes available to the science team. Currently available operational modes are relatively simple. Examples include *directed sampling*, in which the rover visits specific local targets designated by the science team, and *periodic sampling*, in which the rover takes periodic samples while executing a traverse or coverage pattern.

But SA technology allows us to implement new modes that are more effective in over-the-horizon situations. For example, in the *science on the fly* mode, the rover watches for potential science targets during a long traverse. It uses sampling preferences provided by the science team to decide whether it is worthwhile to delay its traverse in order to perform follow-up observations, such as taking a spectrometer reading or visiting the target and using contact sensors.

In the *intelligent site survey* mode, the rover attempts to provide preliminary data about a new site to the science team. It moves around the site using an exploration strategy that balances coverage with selective sampling based on scientist preferences. This approach, taking advantage of the rover's mobility and full sensor suite, is intended to give the science team a better early understanding of the site, and allow them to plan more useful follow-up observations.

We should make it clear that the goal in SA is not to take control of the rover from the science team. Rather, we are providing them with new modes of interaction. In addition to the tightly scripted commands the science team currently has, they would gain the ability to specify more flexible commands such as, "Take microscopic images of carbonate rocks whenever you see them." The science team can begin to think of the rover as a robotic graduate student [Gulick et al., 2001] that interprets their commands intelligently, and can take some initiative when exploring new areas.

Current work in SA is for the most part focused on what we view as foundational capabilities. One capability is data understanding, such as image analysis to identify rocks and sedimentary layers or spectral analysis that characterizes mineral types [Gazis and Roush, 2001]. Another is sensor placement, for example, accurately driving to a particular rock identified from several meters away [Maimone et al., 1999], or finding a flat surface on a rock and aligning a sensor with it in order to obtain a clean reading [Pedersen et al., 2003a].

The first part of our proposed work builds on these foundational capabilities to develop (1) an overall system architecture that supports the kind of SA operational modes described earlier, and (2) the planning module for that system [Smith, 2003]. We are not the first to look at the architecture and planning module. For example, the ongoing OASIS project at JPL has done some early work connecting science data understand-

ing into a rover planning system. Their system was tested onboard a rover in the JPL Mars Yard, and was also used to post-process data from field tests [Estlin et al., 2003]. The proposed work is distinguished by its greater emphasis on long-distance traverses, using what we call a high-mobility operations strategy. We will address these research questions:

- *Problem definition:* What SA operational modes are useful to scientists, technically within reach, and applicable to our mission profile? What are the performance criteria for these operational modes? Can we develop objective and measurable performance metrics?

- *System design and characterization:* What system architecture is most appropriate? In field testing, how useful does each operational mode turn out to be? Under what mission circumstances should each mode be used?

- *Planner design and characterization:* What requirements do different system architectures place on the planning module? What aspects of planner performance are most relevant to overall system performance? How well do different planning models and algorithms perform on this problem?

The second part of our proposed work is to extend partially observable Markov decision process (POMDP) planning algorithms in ways that bridge the gap between the current state-of-art and the planning requirements of the SA domain. POMDP planners can generate high-quality plans that take into account action and sensing uncertainty, but realistic problems in the SA domain are far beyond existing algorithms.

We believe that SA is a particularly interesting application for POMDP planning because of the importance of *sensor planning*, i.e., modeling the information the rover expects to gain from future sensor readings and the effect it will have on subsequent rover decisions. A high-quality SA plan must specify when to invest effort in preliminary sensor readings (like long-range spectrometer samples) that allow the rover to subsequently decide whether costly follow-up observations (like visiting the target and applying contact sensors) are worthwhile.

Other researchers have studied probabilistic planning techniques for the SA problem. [Pedersen et al., 2001] looked at planning sensing actions to enable classification of rock mineral types and meteorites. Prioritized coverage planning for rover exploration was studied in [Moorehead, 2001]. But the proposed work will be the first to apply POMDP planning to this domain.

We should make it clear that we may not be able to field test our POMDP planner onboard a rover. Field testing would require not only that we solve challenging issues in POMDP planning, but also that the software quickly reaches a high level of maturity. Therefore, we plan to test our POMDP planning technology primarily in simulation. We are also interested in ways to inject a few POMDP model elements into an existing planning model, which could support an incremental migration of POMDP capability into the onboard system.

POMDP models have not yet been widely deployed, largely because exact POMDP planning is enormously intractable. Early success stories of approximate approaches

include control of autonomous helicopters [Bagnell and Schneider, 2001] and navigating indoor environments without getting lost [Simmons and Koenig, 1995]. Extending POMDP planning to larger problems is currently a major area of research [Aberdeen, 2002]. We will address these research questions:

- *Heuristic search:* Can we combine heuristic search with efficient value function representations to speed up planning? What heuristics are appropriate? Can we make theoretical performance guarantees about the resulting algorithms?
- *Factored state:* Can POMDP planning scale nearly as well as MDP planning when the state is factored into state elements, and most elements are observable? Can we improve efficient data structures that leverage factored state?
- *Continuous planning:* Can we speed up POMDP planning by interleaving planning and execution? Under what circumstances can the planner save effort by executing part of its plan and gathering the resulting information before making long-term decisions?

The contributions of this thesis will come in two areas. First, we will develop an overall architecture and planning module for one of the first rover SA systems. Our work will be distinguished from comparable projects by the need to generate daily plans that include several kilometers of traverse, and by the primary mission goal, which is biology as opposed to geology (implying a different sensor suite). Second, we will extend POMDP planning techniques and apply them to the SA domain for the first time. We hope that the probabilistic models and algorithms we develop will both improve the quality of SA plans and generalize to other planning domains.

2 Background and Related Work

2.1 Science Autonomy

Several research groups have developed science autonomy systems. Among the most relevant and mature projects is OASIS (Onboard Autonomous Science Investigation System) at JPL [Estlin et al., 2003]. Like the proposed work, OASIS is an onboard system for rovers that includes a continuous planner that can react to science data as it is collected. OASIS has dealt with issues such as robust execution [Estlin et al., 2002] and interdependent goals (e.g. a conditional reward for a camera image if it provides context for a valuable spectrometer reading) [Estlin and Gaines, 2002]. The OASIS system has undergone early testing on the Rocky 7 and 8 rovers in the JPL Mars Yard.

Our long-term goals are broadly similar to those of the OASIS project, but our work is distinguished in that we will support a different mission profile: our rover will generate day-long plans that include several kilometers of traverse, and its primary goal is biology as opposed to geology (implying a different sensor suite). The different mission profile motivates parts of our technical approach: for instance, our planner will need to integrate with existing long-range navigation technology, and our SA operational modes will need to fit into a high-mobility operations strategy.

The ASE (Autonomous Sciencecraft Experiment) at JPL is currently in the process of deploying science autonomy onboard the EO-1 (Earth Orbiter 1) satellite during its extended mission [Chien et al., 2003].[1] There is considerable overlap between ASE and OASIS in terms of personnel and underlying software systems. The ASE architecture supports onboard continuous planning in reaction to incoming science data (e.g. fires, flooding, or volcanic eruptions). As of November 2003, the system was partially deployed, and the science autonomy loop had yet to be closed onboard. Compared to ASE, the rover experiments for the proposed work motivate a planner with different capabilities: rover operations require fewer parallel activities and looser time constraints, but more reasoning about uncertainty.

Researchers at NASA ARC have demonstrated advanced planning and execution technology onboard the K9 rover [Pedersen et al., 2003b]. Their system uses probabilistic planning, and their technique of heuristic search using plan graphs to estimate utility may be applicable to our problem [Dearden et al., 2003]. In contrast to our proposed POMDP planning techniques, the non-determinism in their model is limited to resource usage; decision-making based on science data collected by the rover is not considered.

The RAMS project (Robotic Antarctic Meteorite Search) at CMU demonstrated detection and classification of meteorites onboard the Nomad rover in the Patriot Hills of Antarctica [Wagner et al., 2001]. The onboard classifier was probabilistic, and the RAMS science autonomy system used a simple form of information-gain planning, prioritizing actions that reduce the entropy of classification [Pedersen et al., 2001]. Compared to RAMS, the proposed system will support more complex daily plans, a broader set of science goals, and more Mars-relevant operations.

One of the SA operational modes we will develop is intelligent site survey, applied to sites (on the scale of tens of meters) identified from orbital imagery. We anticipate that the sites we survey will have simple shapes and only scattered obstacles, so that we should be able to use pre-computed coverage patterns such as straight rows or spirals [Shillcutt, 2000]. We do not expect to need more complex coverage algorithms that accommodate structured obstacles or regions with more complex shapes [Choset, 2001].

The more challenging part of the intelligent site survey mode is to use incoming science data during the survey to prioritize time spent on different targets. Prioritized coverage planning for rover exploration was studied in [Moorehead, 2001]. Compared with the proposed work, that planner did not model the outcomes of future observations, and considered only what path the rover should follow, not what sensor readings to take.

Solar-powered rovers are more effective when controlled by power-cognizant traverse planning [Shillcutt, 2000]. The TEMPEST planner returns optimal plans according to its model of power generation and consumption, and supports quick replanning [Tompkins et al., 2002]. The planning module in the proposed SA architecture can work with TEMPEST to provide high-quality plans that take into account both science priorities and resource constraints.

[1] ASE was also scheduled to be included in the Three Corner-Sat imaging constellation (a multi-university effort, originally scheduled for launch on the Space Shuttle in 2003, now indefinitely postponed), and the Air Force TechSat-21 radar constellation (originally scheduled for launch in 2004, now cancelled).

2.2 Probabilistic Planning

Classical planning approaches assume that the planning agent has full knowledge of the world and can predict the results of its actions with certainty. The real world is seldom so kind. Robotics as a field has recently invested a great deal of effort in probabilistic planning techniques that relax the classical planning assumptions. These techniques can work robustly on robots with imperfect sensors and actuators [cite somebodies].

This section discusses a series of probabilistic planning formalisms, building up to the very general POMDP model that we intend to use for the proposed work.

2.2.1 The Markov Process

A Markov process is a stochastic process with the special property that there is no "unmodeled" state: all of the information needed to make the best possible prediction of the future evolution of the process is encapsulated in the current state.

A Markov process is defined by a tuple $(\mathcal{S}, T, s_0)$, where $\mathcal{S}$ is a finite set of states, T is the stochastic transition function, and $s_0 \in \mathcal{S}$ is the initial state. Define s_t to be the state of the process at time t. The process evolves stochastically according to the transition function:

$$\Pr(s_{t+1} = s' \mid s_t = s) = T(s, s')$$

What makes this a Markov process is that if one wants to predict the next state s_{t+1} and one knows the current state s_t, then knowing the history $s_0, \ldots, s_{t-1}$ provides no additional information. All of the information is encapsulated in the current state.

2.2.2 The Markov Decision Process (MDP)

An MDP is a planning model that describes an agent acting in a Markovian world. In addition to the underlying Markov process, the MDP includes an agent with goals (expressed as a reward function) and a means of influencing the evolution of the process [Howard, 1960].

An MDP is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma, s_0)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, T is the stochastic transition function, R is the reward function, $\gamma < 1$ is the discount factor, and $s_0 \in \mathcal{S}$ is the initial state. Define s_t and a_t to be the state of the process and the agent's choice of action, respectively, at time t. The agent's action now influences the evolution of the process according to

$$\Pr(s_{t+1} = s' \mid s_t = s, a_t = a) = T(s, a, s').$$

The agent's reward at each time t is given by $R(s_t, a_t)$. Since the process is Markov, history provides no extra information and the agent can act optimally by choosing a_t solely on the basis of s_t. We say in this case that the agent has a policy $\pi : \mathcal{S} \to \mathcal{A}$, and it chooses its actions such that $a_t = \pi(s_t)$. The long-term (expected discounted) reward of a policy π starting from state s is defined to be

$$J^\pi(s) = E\left[\sum_t \gamma^t R(s_t, a_t) \mid \pi, s_0 = s\right].$$

J^π is also called the *value function* for π. The agent's goal is to choose the optimal policy π^*:

$$\pi^* = \operatorname*{argmax}_\pi J^\pi(s_0).$$

It is also useful to define the *regret* of a policy π starting from s:

$$\mathrm{regret}(\pi, s) = J^{\pi^*}(s) - J^\pi(s).$$

The usual goal of approximate MDP planning is to return a policy π with small $\mathrm{regret}(\pi, s_0)$.

Value iteration is a standard approach to solving MDPs. The optimal policy π^* is derived by first calculating its value function J^{π^*}. In the case of the discounted ($\gamma < 1$) MDP formulation that we use, the optimal value function is the unique fixed point of the following recursion, called the Bellman equation [Bellman, 1957]:

$$J^{\pi^*}(s) = \max_a \left[R(s,a) + \sum_{s'} T(s,a,s') J^{\pi^*}(s') \right].$$

This recursion is globally convergent.

The MDP planning model can represent uncertainty in the outcomes of actions, but it assumes that the world is fully observable: the agent always learns the outcome of an action with full certainty after it is randomly determined. Thus, the MDP model cannot represent hidden state or noisy sensing.

2.2.3 The Partially Observable Markov Decision Process (POMDP)

A POMDP generalizes the MDP notion of uncertainty [Sondik, 1971, Cassandra et al., 1994]. The agent is not assumed to have full certainty about the initial world state. At each time step, the agent selects an action that has some stochastic outcome. The agent cannot observe the outcome directly, but instead receives a noisy observation.

The sequence of events can be viewed as a tree structure (fig. 2.1). Nodes of the tree are points where the agent must make a decision. Each node has a corresponding belief b that the agent would have if it reached that node. The root node is labeled with the initial belief, b_0. Starting from node b, selecting action a, and receiving observation o, the agent proceeds to a new belief $\tau(b, a, o)$, corresponding to one of the children of b in the tree structure.

The POMDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma, b_0 \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $\mathcal{O}$ is the set of observations, T is the stochastic transition function, O is the stochastic observation function, R is the reward function, $\gamma < 1$ is the discount factor, and b_0 is the agent's belief about the initial state. Let s_t, a_t, and o_t denote, respectively, the state, action, and observation at time t. Then we define

$$\begin{aligned} b_0(s) &= \Pr(s_0 = s) \\ O(s,a,o) &= \Pr(o_t = o \mid s_{t+1} = s, a_t = a). \end{aligned}$$

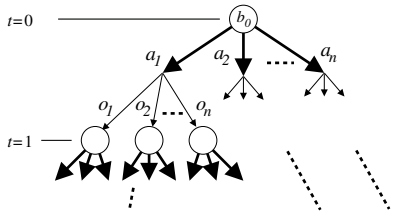

Figure 2.1: POMDP tree structure.

R and T are defined as in the MDP case. In the POMDP model, at time $t+1$, the agent does not know s_{t+1}, but does know the initial belief b_0, and the history of actions and observations up to time t.

The key to solving POMDPs is to realize that the agent can act optimally on its history information by folding the most recent action and observation into its current belief at every step. The belief is recursively updated as follows:

$$b_{t+1} = \tau(b_t, a_t, o_t),$$

If $b' = \tau(b, a, o)$, then

$$b'(s') = \eta O(s', a, o) \sum_s T(s, a, s') b(s),$$

where η is a normalizing constant. The agent now specifies its policy π by giving an action $\pi(b)$ to follow given any current belief b.

In taking this step, we have transformed the POMDP into a *belief-space MDP*. A "state" in the belief-space MDP is a belief of the POMDP. An "outcome" of an action is the result of folding the action and its resulting observation into the current belief.

Value iteration can be used to solve the belief-space MDP as with any other MDP, but unfortunately, the Bellman recursion must now be evaluated not over the finite set of states, but over the continuous space of beliefs. The challenges of solving POMDPs are discussed in the next section.

2.2.4 Solving POMDPs

Solving POMDPs exactly is known to be intractable. The infinite-horizon case is EXPTIME-hard with boolean rewards, and is not known to be decidable for general rewards [Littman, 1996]. Even with finite horizons, the general-reward case is PSPACE-hard.

A discussion of complexity is beyond the scope of this proposal, but we can provide some brief intuition as to why POMDPs are so difficult. First, the search tree for a POMDP with $|\mathcal{A}|$ actions and $|\mathcal{O}|$ observations has an overall branching factor of $|\mathcal{A}||\mathcal{O}|$, which is often very large. This is also called the *curse of history* [Pineau et al., 2003]: each possible history of actions and observations leads to a node of the search tree, and a policy must choose the correct action given any of these histories.

Second, one can consider dynamic programming (DP) approaches, such as value iteration. Using DP, computational effort depends not on the number of nodes in the tree, but on the number of unique beliefs. Unfortunately, POMDP beliefs are drawn from an $|\mathcal{S}|$-dimensional continuous space. There is, however, some additional structure: beliefs that are close to each other have similar values when following the same policy, so we can develop an approximate policy by covering the belief space at some resolution. But then the *curse of dimensionality* arises [Kaelbling et al., 1998]: the number of points required to cover the belief space at a given resolution increases exponentially with its dimensionality $|S|$.

Sondik first introduced POMDPs in the operations research community, and provided the first POMDP value iteration algorithm, called One-Pass [Sondik, 1971]. Since then, a variety of other exact and approximate solution approaches have been suggested. Some well-known techniques include POMDP variants of MDP policy iteration [Hansen, 1998], parameterizing the policy and using gradient ascent [Baxter and Bartlett, 2000], and a particle filter approach that approximates the belief state using Monte Carlo sampling to solve continuous-state POMDPs [Thrun, 2000]. We refer the reader to a survey of POMDP approximation techniques [Aberdeen, 2002].

Perhaps the best-known class of POMDP algorithms are exact value iteration techniques that follow on Sondik's seminal work, such as Witness [Littman, 1994] and Incremental Pruning [Cassandra et al., 1997]. Our own research to date is part of a newer group of algorithms that are inspired by the exact value iteration approach, but that provide speedup using approximate or local value iteration updates. Example algorithms in this class include PBDP [Zhang and Zhang, 2001] and PBVI [Pineau et al., 2003]. §4.5 provides a direct comparison of prior work with our preliminary results.

Because of the intractability of exact approaches, POMDPs have not yet been widely deployed. Some early success stories of approximate approaches include control of autonomous helicopters [Bagnell and Schneider, 2001] and navigating indoor environments without getting lost [Simmons and Koenig, 1995].

3 Technical Approach

3.1 Operations Scenario

Our fundamental goal in developing SA is improving the efficiency of rover science. For any given operation, this means improving the trade-off between resources invested (time, energy, bandwidth, operator attention), and the usefulness of the returned data.

A current-generation Mars rover can move tens of meters per day. Given its constrained mobility, the science team can make efficient use of the rover by giving it specific targets to sample based on data from previous downlinks. But future Mars rovers will be able to move over distances of kilometers and visit multiple sites each day. It will become more important to make efficient use of the rover when it is over the horizon, outside areas visible from the previous day's downlink.

Expanded rover mobility has become technically feasible only recently, and operations strategies have not yet fully adapted. In some rover field tests, the science team

has underused the mobility of the rover [Cabrol et al., 2001]. Nonetheless, a straightforward operations approach is emerging, which we call *baseline high-mobility operations*. Using this approach, the science team selects interesting sites in orbital imagery and can direct the rover to travel long distances and visit multiple sites per day. The rover's time is split between the following operational modes:

- *Directed sampling:* The science team gives the rover specific local targets based on data from previous downlinks.
- *Periodic traverse sampling:* While traveling between interesting sites, the rover takes periodic samples.
- *Periodic site survey:* When the rover reaches an interesting site, it follows a coverage pattern and takes periodic samples. This data constitutes a preliminary survey, and the science team can follow up with directed sampling as necessary.

Our proposal for structuring SA is to add new modes that extend baseline high-mobility operations. Some modes under consideration are the following (in order from least to most advanced):

- *Science on the fly:* While traveling between interesting sites, the rover watches for potential science targets. It uses sampling preferences provided by the science team to decide whether it is worthwhile to delay its traverse in order to perform follow-up observations, such as taking a spectrometer reading or visiting the target and using contact sensors.
- *Intelligent site survey:* When the rover reaches an interesting site, it moves around the site using an exploration strategy that balances coverage with selective sampling based on scientist preferences. This data constitutes a preliminary survey, and the science team can follow up with directed sampling as necessary.
- *Science-aware path following:* The rover follows a path defined by science features. Examples include the margin of a dry riverbed or crater, or evaporite deposits that mark an ancient shoreline.
- *Science-aware region mapping:* The rover identifies areas with uniform science properties and attempts to find their boundaries, in order to determine extent and generate a map. Example areas are geological units and habitats (defined by the presence or absence of particular organisms).

In order to decide where to focus our development effort, we can rank each mode on several criteria:

1. *Efficiency enhancement:* It should provide more useful science data than the comparable baseline strategy given the same resource investment (in terms of energy, time, bandwidth, and operator attention).
2. *Broad applicability:* It should be applicable under circumstances that are commonly encountered in mission operations.

3. *Ease of migration:* It should not be a "disruptive technology". It should integrate easily with existing systems and require minimal retraining for the science team.

4. *Testability:* There should be clear performance criteria, and it should be straightforward to compare performance with the baseline strategy.

5. *Feasibility:* It should be technically feasible to develop.

We believe that all four of the operational modes described earlier have good potential to enhance efficiency and be broadly applicable. (Although this is difficult to determine *a priori* and would best be analyzed through tests in the field.)

However, science on the fly and intelligent site survey have a clear advantage in terms of ease of migration. They are essentially drop-in replacements for periodic traverse sampling and periodic site survey. The science team can specify daily activities just as they would in the baseline strategy, then "flip the switch" to use SA operational modes where appropriate. (Of course, there is additional effort, such as specifying sampling preferences, which is discussed later.) The fact that these modes have corresponding baseline modes also improves testability. We can use existing performance metrics for the baseline modes, and easily compare results. Because of these factors, we will limit the scope of the proposed work to the first two SA modes.

The remainder of the technical approach section addresses the final criterion of feasibility. We argue that an implementation of the first two SA modes is feasible by providing a preliminary SA system design that includes an overall architecture and a design for the planning module.

3.2 System Architecture

In developing an architecture for our SA system, we started with a standard three-layer autonomy architecture [Firby, 1989], and made extensions where necessary to provide additional functionality. The architecture is shown in fig. 3.1. The light-colored planning modules will be developed as part of the proposed work; the other modules either already exist, or will be developed by our colleagues in parallel with the proposed work.

The functional layer, executive, and planner make up the three-layer autonomy architecture. As we move through the layers from bottom to top, the reasoning involved becomes more abstract and the reaction time becomes longer. The functional layer consists primarily of fast reactive behaviors such as low-level control loops. The executive executes a plan by enabling and disabling functional layer behaviors. It also keeps track of system state and reacts to fault conditions. The planner is responsible for generating a plan to achieve system goals and updating the plan based on requests from the executive. The three-layer architecture is commonly used because it provides a balance between reactivity and high-level reasoning [Bonasso, 1991].

In order to support SA operations, we have made several additions to the three-layer architecture. The first is the science observer, which is responsible for monitoring incoming science data and detecting and classifying potential targets for follow-up sampling. It forwards this information to the science planner. Because of its reactive nature, one might consider the science observer to be part of the functional layer.

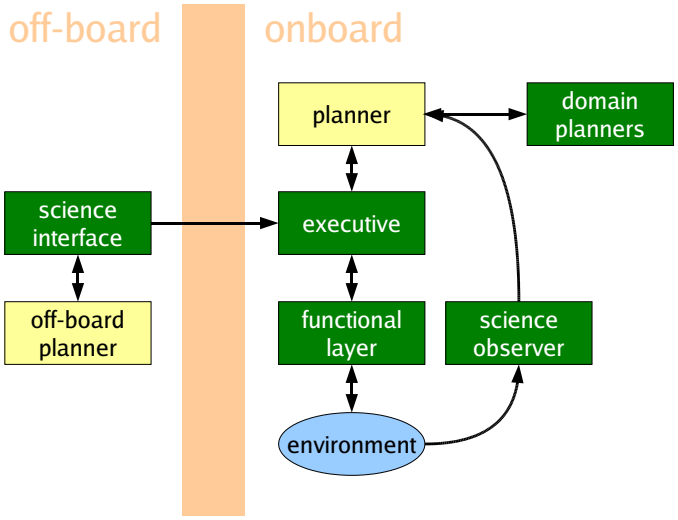

Figure 3.1: Science autonomy system architecture.

Based on the usual constraints of the three-layer architecture, this would mean that information from the science observer should be channeled through the executive. But we argue that the science observer is a special case.

Normally the planner is invoked by the executive in what we would call a *synchronous* manner. The executive requests a plan update at the start of a system run, or at particular points in execution: for example, when a token in the plan is completed with a particular status (success/failure), or because the update was scheduled as part of the plan. In contrast, the science observer runs in the background throughout execution, and can detect a potential science target at any time. Thus, it should have the ability to make *asynchronous* requests to update the plan based on its new information. Channeling this information through the executive may not be appropriate.

Another addition to the architecture is calling out the existence of domain planners as part of the planning layer. Planning for rover operations is such a complex undertaking that it is infeasible to start our planning module development from scratch. Thus, the planner we develop needs to integrate with existing technology. In particular, we will interact with systems for local navigation based on rover sensing [Urmson et al., 2002] and resource-cognizant long-range navigation based on satellite terrain models [Tompkins et al., 2004].

Finally, as part of the overall SA problem, we must consider how the science team commands the rover. The off-board portion of our architecture includes a science interface and off-board planner. The science interface allows the science team to specify explicit goals such as sites to visit and targets for directed sampling, as well as implicit goals, such as sampling preferences for the SA operational modes. The off-board planner is essentially a copy of the onboard planner. It is used by the science team to predict how rover activities will proceed based on the goals they have specified. Using this off-

board analysis, the team can interactively tune their goals until they are satisfied with the predicted results.

3.3 Sampling Preferences and the Science Observer

A key issue in SA system development is (1) how to represent scientist sampling preferences, and (2) how the science observer can extract information that is relevant to those preferences.

Much early work in rover science data understanding focused on *key signatures* [Gazis and Roush, 2001]. The idea is that the science team knows roughly what to expect in an area, and has high-value science targets in mind that match a particular signature: for example, they might want to look for carbonate deposits or vesicular rocks that indicate volcanic activity.

The key signature approach, although often useful, has some limitations. First, it assumes that the science team knows what to expect in an area—but the most interesting areas to explore are precisely the ones that are most surprising. In these areas, a wealth of interesting features may be present, but none that match any of the key signatures. Second, the approach introduces bias into the returned data. If the key signatures cause the rover to preferentially seek out carbonates, the science team may develop a falsely inflated sense of their frequency in an area, missing other features and leading to an incorrect understanding of the site.

Because of the limitations of the key signature approach, some extensions have been proposed [Castaño et al., 2003]. These extensions include:

- *Anomaly detection:* Preferentially sampling features that are unlike any seen previously.

- *Representative sampling:* Cataloging all of the feature types present in an area and ensuring that at least one feature of each type is sampled. (Returning a frequency distribution over the various feature types is also helpful.)

These approaches complement key signatures in that they remain useful when encountering unexpected features, and they help to avoid missing important features due to selective bias.

It remains to specify how the science observer can provide information that is relevant to scientist preferences. Although development of the science observer is beyond the scope of the proposed work, we briefly discuss this issue because it is important to the feasibility of the overall system.

The first step in the science observer's analysis is to detect features and measure a set of characteristics for each feature (such as size, albedo, vesicularity, spectral elements, etc.). Considerable prior work in image and spectral analysis indicates that reasonably effective automatic detection and measurement is feasible, though not yet close to human standards [Gulick et al., 2000].

We must still explain how the measured characteristics of a feature can be related to sampling preferences in order to determine if follow-up observations are needed. With the key signature approach, the science observer needs to classify features only

according to whether they match the pre-specified signatures. Some straightforward approaches include:

- *Manual tuning:* An exemplar feature for each key signature is selected, and features within a particular distance of the exemplar in characteristic space are said to match the key signature. The distance threshold along each characteristic dimension is manually tuned. This is the easiest approach to implement. A downside is that adding a new key signature takes considerable time and may require the science team to consult an imaging or spectral analysis expert.

- *Off-line supervised learning:* The science team codes positive and negative examples of features according to whether they match the signature. The examples are used to train a classifier. This approach is simpler, but still somewhat labor-intensive.

Anomaly detection and representative sampling are more difficult: in order to support these preferences, the rover must make online updates to its classifier so that it takes into account new features [Estlin et al., 1999]. With this in mind, we propose that the science observer's classifier should use unsupervised Bayesian clustering. Bayesian clustering is a familiar tool for scientific data analysis [Cheeseman and Stutz, 1996], and since it models classification uncertainty probabilistically, it can provide useful additional information for POMDP planning.

The clustering algorithm returns a maximum probability posterior model of the observed features that specifies (1) a set of clusters, and (2) for each feature, a probability distribution that states how likely it is to be part of each cluster. Within this framework, anomalous features are those that are not well-explained by any cluster. Representative sampling can be accomplished by sampling one feature from each cluster.

We further propose that the science team can select automatically generated clusters to use as key signatures. This avoids manually tuning or training a signature classifier. Also, the automatic clustering will tend to generate clusters that are distinct from each other based on characteristics the rover can measure, which should help the science team avoid creating signatures that the rover cannot reliably distinguish from background features.

3.4 POMDP Planning: Applicability

The second part of our proposed work is to extend POMDP planning algorithms in ways that bridge the gap between the current state-of-art and the planning requirements of the SA domain. This section explains why SA is a natural domain in which to apply POMDP planning.

Our main motivation for considering a POMDP model for SA is the importance of *sensor planning*, i.e., modeling the information the rover expects to gain from future sensor readings and the effect it will have on subsequent rover decisions. For example, in the LITA domain, when the rover sees a rock, it has the option of applying its long-range spectrometer. Although the spectrum for the rock has some intrinsic value to the science team, it has an additional *informational* value: once the rover has the spectrum,

it can make a better decision about whether to approach the rock and take a fluorescence imager (FI) image.

Consider how we could implement sensor planning. Assume that we start with a deterministic model that, for each rock type, specifies the intrinsic value of both a spectrum and an FI image. In the absence of uncertainty, it is straightforward to design a planner based on this model. When deciding whether to take an FI image of a particular rock, the planner can simply look up the value of the image and compare it to the value of other actions that are competing for its resources.

Now we move closer to the real scenario. Suppose that instead of knowing the type of the rock, the rover analyzes the initial image in which the rock was detected, and has a noisy classification, a distribution over types the rock might have. For the moment, ignore the availability of the spectrometer. It is still easy to decide whether or not to take an FI image. Based on the noisy classification, the planner can calculate the *expected* value of the FI image and use that in place of the certain value we discussed earlier.

But return to the spectrometer. How can we decide whether to use it? We can determine the expected intrinsic value of the spectrum just as we did for the FI image, but this is an underestimate of the overall value, because it does not take into account the *informational* value: the spectrum would decrease the uncertainty in the classification, giving the rover a better estimate of the expected value of the FI image, and therefore a better basis on which to make the decision of whether to take the FI image.

There are several heuristic solutions to this problem. At modeling time, the intrinsic values of targets can be inflated to include a manual estimate of informational value. Or the planner could always choose the action that is expected to most reduce the entropy of the classification [Pedersen, 2000]. Or it could attempt to reduce entropy only when entropy is above a fixed threshold [Cassandra et al., 1996]. A good heuristic approach for LITA might be to assign a pseudo-informational value to each action that scales with how much it is expected to reduce classification entropy.

So to summarize: we can capture the first order effects of uncertainty by planning based on the expected intrinsic value of actions. We can capture the second order effects by assigning a heuristic pseudo-informational value to sensing actions. But now we consider third order effects: the way that informational value depends on context, including:

- *Importance of subsequent decisions:* In cases where the potential value of the FI image is higher or the FI imaging operation is more costly, the decision of whether to take the FI image has more impact and the informational value of the spectrum is higher.

- *Inabiity to apply the information:* If the rover cannot take an FI image of the rock, the spectrum has no informational value. This might happen if there is no traversable path to the rock, or if there is not enough time in the rover's schedule.

- *Irrelevance of uncertainty:* If the rock could have any of three types, but FI images of all three types have the same value, then there is no point in further narrowing it down.

POMDP planning goes beyond the heuristic approaches in that it accurately models these third order effects, and does so in a principled way that relaxes the need for careful tuning of manual heuristics.

3.5 POMDP Planning: Feasibility

Unfortunately, realistic problems in the SA domain are far beyond the abilities of existing POMDP algorithms. This section discusses why there is some hope of overcoming the present intractability of POMDP planning in order to tackle realistically sized problems in the SA domain.

The first part of our proposed research in planning techniques is to combine heuristic search with efficient value function representations to speed up planning. In the POMDP value iteration community, much effort has been focused on compact representations of the value function [Kaelbling et al., 1998]. Most existing value iteration approaches generate policies that are guaranteed to perform well starting from any initial belief. But we believe that in most real-world situations, one should assume that the agent starts with knowledge of a particular initial belief, and can use that knowledge to focus on relevant parts of the belief space.

We propose a POMDP algorithm called heuristic search value iteration (HSVI) that uses heuristic search to repeatedly explore forward from the initial belief. HSVI stores upper and lower bounds on the value function. During forward exploration, HSVI selects actions and observations based on its current bounds. Whenever it visits a belief during exploration, it performs a local update that tightens the bounds at that belief. Because of the representation HSVI uses for its bounds, each local update also generalizes to neighboring beliefs.

§4 describes our preliminary work on HSVI, discusses its soundness and convergence, and compares its performance with other state-of-the-art value iteration algorithms on four benchmark problems from the literature. On some of these problems, HSVI displays speedups of greater than 100. We provide additional results on the *RockSample* problem, a highly simplified SA problem. Our largest instance of *RockSample* has 12,545 states, 10 times larger than most problems in the scalable POMDP literature.

The second part of our proposed research in planning is to leverage a factored state representation in two ways. First, it is commonly the case that some state elements are completely observable, or at least that uncertainty about them is small enough to be ignored. For example, in the SA problem, it appears that representing uncertainty about characteristics of science features is more important than for most other state elements, such as position or remaining battery energy.

We believe that the POMDP community has not paid enough attention to handling these observable state elements efficiently. The tractability of POMDP algorithms should scale according to the amount of uncertainty involved: in the extreme case that all state elements are observable, they should be nearly as efficient as good MDP algorithms. This appears to largely be a matter of selecting POMDP algorithms with MDP analogs, and doing the appropriate bookkeeping. In a sense, we are identifying a POMDP subclass of "mostly observable MDPs" (MOMDPs), and proposing to adapt POMDP algorithms so that they scale well over this subclass.

Factored state can also be used to develop efficient representations of the beliefs and α vectors used in value iteration. One such representation is algebraic decision diagrams (ADDs) [Bahar et al., 1993, Hoey et al., 1999]. ADDs are directed acyclic graphs that branch on boolean variables such as state elements. They can compactly represent state-indexed vectors that have uniform values over many entries (or nearly uniform values [St-Aubin et al., 2000]). We are investigating ADD extensions that can also compactly represent probability distributions with some conditional independence between state elements (see §5.2.1).

The final part of our proposed planning research is to investigate continuous planning (interleaving planning and execution) in the POMDP context. There are two major ways that continuous planning can improve planning performance [Nourbakhsh, 1997]:

- *Fast updates:* The planner can reuse information from past planning episodes to speed up plan updates.

- *Deferred decision-making:* In some cases, the agent can execute the first part of the plan before the overall plan is complete, while still guaranteeing low regret. This helps avoid spinning out what-if scenarios; instead, the agent can defer decision-making until it has the relevant information.

Continuous planning seems like a natural idea in the POMDP model, but to our knowledge it has not been explicitly addressed before. This is apparently because many POMDP algorithms generate a policy with global performance bounds. Once such a policy is found, it never needs to be updated, because it can be applied from any point in the belief space.

However, HSVI and other recent algorithms show that planning can converge more quickly when the planner focuses its attention on a subset of the belief space, producing a policy guaranteed to have good performance only when starting from the specified initial belief. In this framework, policy updates are again relevant, especially when execution of the early parts of a plan leads to unexpected results.

The current version of HSVI (somewhat fortuitously) already supports both main continuous planning goals discussed earlier. We would like to run experiments to determine the benefits, and possibly develop extensions.

When notified of a new initial belief for planning, HSVI can reuse work from past planning episodes simply by keeping its old bounds on the value function. Thus, during plan updates, it would start with tighter bounds, but at the cost of having more elements in its initial bounds representations (which slows updates). It is not clear whether this trade-off is worthwhile. There may be techniques for culling parts of the old bounds that are irrelevant given the new initial belief.

"Convergence" for HSVI normally means that it has tight bounds on the value function at the initial belief. However, it is often the case that a particular first action provably dominates all others well before it becomes clear what the exact value of that action is. In this case, the agent could stop early and execute the first action. Dominance can be checked naïvely by examining the expected value bounds after taking each action. We also have ideas for stronger dominance-checking techniques based on the structure of the bounds representation or on additional look-ahead.

4 Preliminary Work: Heuristic Search Value Iteration

4.1 Algorithm Introduction

HSVI is an approximate POMDP solution algorithm that combines techniques for heuristic search with piecewise linear convex value function representations. HSVI stores upper and lower bounds on the optimal value function V^*. Its fundamental operation is to make a local update at a specific belief, where the beliefs to update are chosen by exploring forward in the search tree according to heuristics that select actions and observations.

HSVI makes asynchronous (Gauss-Seidel) updates to the value function bounds, and always bases its heuristics on the most recent bounds when choosing which successor to visit. It uses a depth-first exploration strategy. Beyond the usual memory vs. time trade-off, this choice makes sense because a breadth-first heuristic search typically employs a priority queue, and propagating the effects of asynchronous bounds updates to the priorities of queue elements would create substantial extra overhead.

We refer to the lower and upper bound functions as $\underline{V}$ and $\bar{V}$, respectively. We use the interval function $\hat{V}$ to refer to them collectively, such that

$$\begin{aligned} \hat{V}(b) &= [\underline{V}(b), \bar{V}(b)] \\ \mathrm{width}(\hat{V}(b)) &= \bar{V}(b) - \underline{V}(b) \end{aligned}$$

HSVI is outlined in algs. 4.1 and 4.2. The following subsections describe aspects of the algorithm in more detail.

4.1.1 Value Function Representation

Most value iteration algorithms focus on storing and updating the lower bound. The *vector set* representation is commonly used. The value at a point b is the maximum projection of b onto a finite set $\Gamma_{\underline{V}}$ of vectors α:

$$\underline{V}(b) = \max_{\alpha \in \Gamma_{\underline{V}}} (\alpha \cdot b).$$

For finite-horizon POMDPs, a finite vector set can represent V^* exactly [Sondik, 1971]. Even for the discounted infinite-horizon formulation, a finite vector set can approximate V^* arbitrarily closely. Equally important, when the value function is a lower bound, it is easy to perform a local update on the vector set by adding a new vector.

Unfortunately, if we represent the upper bound with a vector set, updating by adding a vector does not have the desired effect of improving the bound in the neighborhood of the local update. To accommodate the need for updates, we use a *point set* representation for the upper bound. The value at a point b is the projection of b onto the convex hull formed by a finite set $\Upsilon_{\bar{V}}$ of belief/value points $(b_i, \bar{v}_i)$. Updates are performed by adding a new point to the set.

The projection onto the convex hull is calculated with a linear program (LP). This upper bound representation and LP technique was suggested in [Hauskrecht, 2000], but in that work LP projection seems to have been rejected without testing on time complexity grounds. Note that with the high dimensionality of the belief space in

Algorithm 4.1. $\pi = \mathrm{HSVI}(\epsilon)$.

$\mathrm{HSVI}(\epsilon)$ returns a policy π such that $\mathrm{regret}(\pi, b_0) \leq \epsilon$.[a]

1. Initialize the bounds $\hat{V}$.
2. While $\mathrm{width}(\hat{V}(b_0)) > \epsilon$, repeatedly invoke $\mathrm{explore}(b_0, \epsilon, 0)$.
3. Having achieved the desired precision, return the direct-control policy π corresponding to the lower bound.

[a] In fact, π can be executed starting at any belief b. In general, $\mathrm{regret}(\pi, b) \leq \mathrm{width}(\hat{V}(b))$, which is guaranteed to be less than ϵ only at b_0.

Algorithm 4.2. $\mathrm{explore}(b, \epsilon, t)$.

explore recursively follows a single path down the search tree until satisfying a termination condition based on the width of the bounds interval. It then performs a series of updates on its way back up to the initial belief.

1. If $\mathrm{width}(\hat{V}(b)) \leq \epsilon\gamma^{-t}$, return.
2. Select an action a^* and observation o^* according to the forward exploration heuristics.
3. Call $\mathrm{explore}(\tau(b, a^*, o^*), \epsilon, t + 1)$.
4. Locally update the bounds $\hat{V}$ at belief b.

our larger problems, LP projection is far more efficient than explicitly calculating the convex hull: an explicit representation would not even fit into available memory. We solve the LP using the commercial ILOG CPLEX software package.

4.1.2 Initialization

HSVI requires initial bounds, which we would like to have the following properties:

1. *Validity:* $\underline{V}_0 \leq V^* \leq \bar{V}_0$.[2]

[2] Throughout this paper, inequalities between functions are universally quantified, i.e., $V \leq V'$ means $V(b) \leq V'(b)$ for all b.

2. *Uniform improvability:* This property is explained in the section on theoretical results.

3. *Precision:* The bounds should be fairly close to V^*.

4. *Efficiency:* Initialization should take a negligible proportion of the overall running time.

The following initialization procedure meets these requirements. We calculate $\underline{V}_0$ using the blind policy method [Hauskrecht, 1997]. Let π_a be the policy of always selecting action a. We can calculate a lower bound $\underline{R}_a$ on the long-term reward of π_a by assuming that we are always in the worst state to choose action a from.

$$\underline{R}_a = \sum_{t=0}^{\infty} \gamma^t \min_s R(s,a) = \frac{\min_s R(s,a)}{1-\gamma}$$

We select the tightest of these bounds by maximizing.

$$\underline{R} = \max_a \underline{R}_a$$

Then the vector set for the initial lower bound $\underline{V}_0$ contains a single vector α such that every $\alpha(s) = \underline{R}$.

To initialize the upper bound, we assume full observability and solve the MDP version of the problem [Astrom, 1965]. This provides upper bound values at the corners of the belief simplex, which form the initial point set. We call the resulting upper bound V_{MDP}.

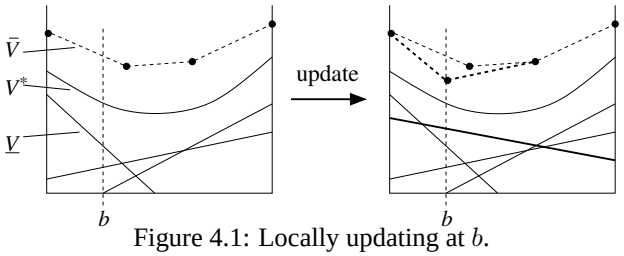

Figure 4.1: Locally updating at b.

4.1.3 Local Updates

The Bellman update, H, is the fundamental operation of value iteration. It is defined as follows:

$$\begin{aligned} Q^V(b,a) &= \sum_s R(s,a)b(s) + \\ &\quad \gamma \sum_o \Pr(o \mid b,a) V(\tau(b,a,o)) \\ HV(b) &= \max_a Q^V(b,a) \end{aligned}$$

Algorithm 4.3. $\beta = \text{backup}(\underline{V}, b)$.

The backup function can be viewed as a generalization of the Bellman update that makes use of gradient information. The assignments are universally quantified, e.g., $\beta_{a,o}$ is computed for every a, o.

1. $\beta_{a,o} \leftarrow \text{argmax}_{\alpha \in \Gamma_V}(\alpha \cdot \tau(b, a, o))$
2. $\beta_a(s) \leftarrow R(s, a) + \gamma \sum_{o,s'} \beta_{a,o}(s') O(s', a, o) T(s, a, s')$
3. $\beta \leftarrow \text{argmax}_{\beta_a}(\beta_a \cdot b)$.

$Q^V(b, a)$ can be interpreted as the value of taking action a from belief b.

Exact value iteration calculates this update exactly over the entire belief space. HSVI, however, uses local update operators based on H. Because the lower and upper bound are represented differently, we have distinct local update operators L_b and U_b. *Locally updating* at b means applying both operators. To update the lower bound vector set, we add a vector. To update the upper bound point set, we add a point. The operators are defined as:

$$\begin{aligned} \Gamma_{L_b \underline{V}} &= \Gamma_{\underline{V}} \cup \text{backup}(\underline{V}, b) \\ \Upsilon_{U_b \bar{V}} &= \Upsilon_{\bar{V}} \cup (b, H\bar{V}(b)), \end{aligned}$$

where $\text{backup}(\underline{V}, b)$ is the usual gradient backup, described in alg. 4.3.

Fig. 4.1 represents the structure of the bounds representations and the process of locally updating at b. In the left side of the figure, the points and dotted lines represent $\bar{V}$ (upper bound points and convex hull). Several solid lines represent the vectors of $\Gamma_{\underline{V}}$. In the right side of the figure, we see the result of updating both bounds at b, which involves adding a new point to $\Upsilon_{\bar{V}}$ and a new vector to $\Gamma_{\underline{V}}$, bringing both bounds closer to V^*.

HSVI periodically prunes dominated elements in both the lower bound vector set and the upper bound point set. Pruning occurs each time the size of the relevant set has increased by 10% since the last pruning episode. This pruning frequency was not carefully optimized, but there is not much to be gained by tuning it, since we do note see substantial overhead either from keeping around up to 10% too many elements or from the pruning operation itself. For the lower bound, we prune only vectors that are *pointwise* dominated (i.e., dominated by a single other vector). This type of pruning does not eliminate all redundant vectors, but it is simple and fast. For the upper bound, we prune all dominated points, defined as $(b_i, \bar{v}_i)$ such that $H\bar{V}(b_i) < \bar{v}_i$.

4.1.4 Forward Exploration Heuristics

This section discusses the heuristics that are used to decide which child of the current node to visit as the algorithm works its way forward from the initial belief. Starting

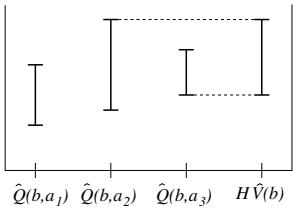

Figure 4.2: Relationship between $\hat{Q}(b, a_i)$ and $H\hat{V}(b)$.

from parent node b, HSVI must choose an action a^* and an observation o^*: the child node to visit is $\tau(b, a^*, o^*)$.

Define the *uncertainty* at b to mean the width of the bounds interval. Recalling that the regret of a policy returned by HSVI is bounded by the uncertainty at the root node b_0, our goal in designing the heuristics is to ensure that updates at the chosen child tend to reduce the uncertainty at the root.

First we consider the choice of action. Define the interval function $\hat{Q}$ as follows:

$$\hat{Q}(b, a) = [\underline{Q^V}(b, a), Q^{\bar{V}}(b, a)]$$

Fig. 4.2 shows the relationship between the bounds $\hat{Q}(b, a)$ on each potential action and the bounds $H\hat{V}(b)$ at b after a Bellman update. We see that the $H\hat{V}(b)$ interval is determined by only two of the $\hat{Q}(b, a)$ intervals: the ones with the maximal upper and lower bounds. This relationship immediately suggests that, among the $\hat{Q}$ intervals, we should choose to update one of these two most promising actions. But which one? It turns out we can guarantee convergence only by choosing the action with the greatest *upper* bound.

$$a^* = \operatorname*{argmax}_a Q^{\bar{V}}(b, a).$$

This is sometimes called the IE-MAX heuristic [Kaelbling, 1993]. It works because, if we repeatedly choose an a^* that is sub-optimal, we will eventually discover its sub-optimality when the a^* upper bound drops below the upper bound of another action. However, if we were to choose a^* according to the highest lower bound, we might never discover its sub-optimality, because further work could only cause its lower bound to rise.

Next we need to select an observation o^*. Consider the relationship between $\hat{Q}(b, a^*)$ and the bounds at the various child nodes $\tau(b, a^*, o)$ that correspond to different observations. From the Bellman equation, we have

$$\mathrm{width}(\hat{Q}(b, a^*)) = \gamma \sum_o \Pr(o|b, a^*) \mathrm{width}(\hat{V}(\tau(b, a^*, o))).$$

Note that this explains the termination criterion of $\mathrm{explore}$, $\mathrm{width}(\hat{V}(b)) \leq \epsilon\gamma^{-t}$. Because the uncertainty at a node b after an update is at most γ times a weighted average of its child nodes, we have successively looser requirements on uncertainty at deeper nodes: we rely on the γ factor at each layer on the way back up to make up the

Algorithm 4.4. $\pi = \mathrm{AnytimeHSVI}()$.

> $\mathrm{AnytimeHSVI}()$ is an anytime variant of HSVI. When interrupted, it returns a policy whose regret is bounded by the current value of $\mathrm{width}(\hat{V}(b_0))$.

Implementation: As HSVI, but in step (2), in the call to $\mathrm{explore}(b_0, \epsilon, 0)$, replace ϵ with $\zeta\ \mathrm{width}(\hat{V}(b_0))$, where $\zeta < 1$ is a scalar parameter. Empirically, performance is not very sensitive to ζ; we used $\zeta = 0.95$ in the experiments, which gives good performance.

difference. Given these facts, we can define *excess uncertainty*

$$\mathrm{excess}(b, t) = \mathrm{width}(\hat{V}(b)) - \epsilon\gamma^{-t}$$

such that a node with negative excess uncertainty satisfies the $\mathrm{explore}$ termination condition. We say such a node is ***finished***. Conveniently, the excess uncertainty at b is at most a probability-weighted sum of the excess uncertainties at its children

$$\mathrm{excess}(b, t) \leq \sum_o \Pr(o|b, a^*)\mathrm{excess}(\tau(b, a^*, o), t + 1).$$

Thus we can focus on ensuring early termination by selecting the depth $t + 1$ child that most contributes to excess uncertainty at b:

$$o^* = \operatorname*{argmax}_o \Big[\Pr(o|b, a^*)\mathrm{excess}(\tau(b, a^*, o), t + 1) \Big].$$

Past heuristic search approaches have usually either sampled from $\Pr(o|b, a^*)$ or maximized weighted uncertainty rather than weighted *excess* uncertainty. We find the excess uncertainty heuristic to be empirically superior. In addition, this heuristic allows us to derive a time bound on HSVI convergence.

4.1.5 Anytime Usage

The definition of $\mathrm{HSVI}(\epsilon)$ given above assumes that we know in advance that we want a policy with regret bounded by ϵ. In practice, however, we often do not know what a reasonable ϵ is for a given problem—we just want the algorithm to do the best it can in the available time. In support of this goal, we define a variant algorithm called $\mathrm{AnytimeHSVI}$ (alg. 4.4). Where HSVI uses a fixed ϵ, $\mathrm{AnytimeHSVI}$ adjusts ϵ at each top-level call to $\mathrm{explore}$, setting it to be slightly smaller than the current uncertainty at b_0. Instead of having a fixed finish line, we have a finish line that is always just ahead, receding as we approach.

$\mathrm{AnytimeHSVI}$ is used for all of the experiments in this paper. However, our theoretical analysis focuses on $\mathrm{HSVI}(\epsilon)$, which is easier to handle mathematically.

4.2 Theoretical Results

This section discusses some of the key soundness and convergence properties of $\mathrm{HSVI}(\epsilon)$.
The actual proofs will be presented in a forthcoming technical report [Smith and Simmons, 2004].

- The initial lower and upper bound value functions are *uniformly improvable*, meaning that applying H brings them everywhere closer to V^*.

- If $\underline{V}$ is uniformly improvable, then the corresponding direct control policy $P_{\underline{V}}$ *supports* $\underline{V}$, meaning that $\underline{V} \leq J^{P_{\underline{V}}}$.[3]

- If $\bar{V}$ is uniformly improvable, then it is valid, in the sense that $V^* \leq \bar{V}$.

- Our local update operators preserve uniform improvability. Thus, throughout the execution of HSVI, the current best policy $P_{\underline{V}}$ supports $\underline{V}$, and $\bar{V}$ is valid.

- Together, these facts imply that HSVI has valid bounds on the direct control policy, in the sense that $\underline{V} \leq J^{\pi_{\underline{V}}} \leq \bar{V}$. This validity holds throughout execution and everywhere in the belief space.

- The $\mathrm{regret}(\pi, b_0)$ of the policy π returned by $\mathrm{HSVI}(\epsilon)$ is at most ϵ. When $\mathrm{AnytimeHSVI}$ is interrupted, the $\mathrm{regret}(\pi, b_0)$ of the current best policy π is at most $\mathrm{width}(\hat{V}(b_0))$.

- There is a finite depth

  $$t_{\max} = \lceil \log_\gamma(\epsilon / \left\| \bar{V}_0 - \underline{V}_0 \right\|_\infty) \rceil$$

  such that all nodes with depth $t \geq t_{\max}$ are finished.

- The uncertainty at a node never increases (thus finished nodes never become unfinished).

- After each top-level call to explore, at least one previously unfinished node is finished. This property depends on our particular choice of heuristics.

- As a result, $\mathrm{HSVI}(\epsilon)$ is guaranteed to terminate after performing at most $u_{\max}$ updates, where

  $$u_{\max} = t_{\max} \frac{(|\mathcal{A}||\mathcal{O}|)^{t_{\max}+1} - 1}{|\mathcal{A}||\mathcal{O}| - 1}.$$

  (Note this is a conservative theoretical bound; empirically, it is much faster.)

[3] Direct and lookahead control policies corresponding to a value function are discussed in, e.g., [Hauskrecht, 2000].

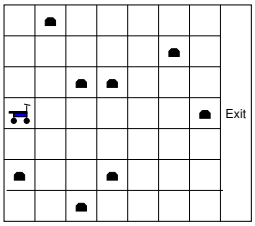

Figure 4.3: *RockSample*$[7, 8]$.

4.3 The *RockSample* Problem

To test HSVI, we have developed *RockSample*, a scalable problem that models rover science exploration (fig. 4.3). The rover can achieve reward by sampling rocks in the immediate area, and by continuing its traverse (reaching the exit at the right side of the map). The positions of the rover and the rocks are known, but only some of the rocks have scientific value; we will call these rocks "good". Sampling a rock is expensive, so the rover is equipped with a noisy long-range sensor that it can use to help determine whether a rock is good before choosing whether to approach and sample it.

An instance of *RockSample* with map size $n \times n$ and k rocks is described as *RockSample*$[n, k]$. The POMDP model of *RockSample*$[n, k]$ is as follows. The state space is the cross product of $k + 1$ features: *Position* = $\{(1, 1), (1, 2), \ldots, (n, n)\}$, and k binary features *RockType*$_i$ = $\{$*Good*, *Bad*$\}$ that indicate which of the rocks are good. There is an additional terminal state, reached when the rover moves off the right-hand edge of the map. The rover can select from $k + 5$ actions: $\{$*North*, *South*, *East*, *West*, *Sample*, *Check*$_1$, ..., *Check*$_k\}$. The first four are deterministic single-step motion actions. The *Sample* action samples the rock at the rover's current location. If the rock is good, the rover receives a reward of 10 and the rock becomes bad (indicating that nothing more can be gained by sampling it). If the rock is bad, it receives a penalty of -10. Moving into the exit area yields reward 10. All other moves have no cost or reward.

Each *Check*$_i$ action applies the rover's long-range sensor to rock i, returning a noisy observation from $\{$*Good*, *Bad*$\}$. The noise in the long-range sensor reading is determined by the efficiency η, which decreases exponentially as a function of Euclidean distance from the target. At $\eta = 1$, the sensor always returns the correct value. At $\eta = 0$, it has a 50/50 chance of returning *Good* or *Bad*. At intermediate values, these behaviors are combined linearly. The initial belief is that every rock has equal probability of being *Good* or *Bad*.

4.4 Experimental Results

We tested HSVI on several well-known problems from the scalable POMDP literature, as well as instances of *RockSample*. Our benchmark set follows [Pineau et al., 2003], which provides performance numbers for PBVI and some other value iteration algorithms. Note that all of the problems have $\gamma = 0.95$.

Experiments were conducted as follows. Periodically during each run, we inter-

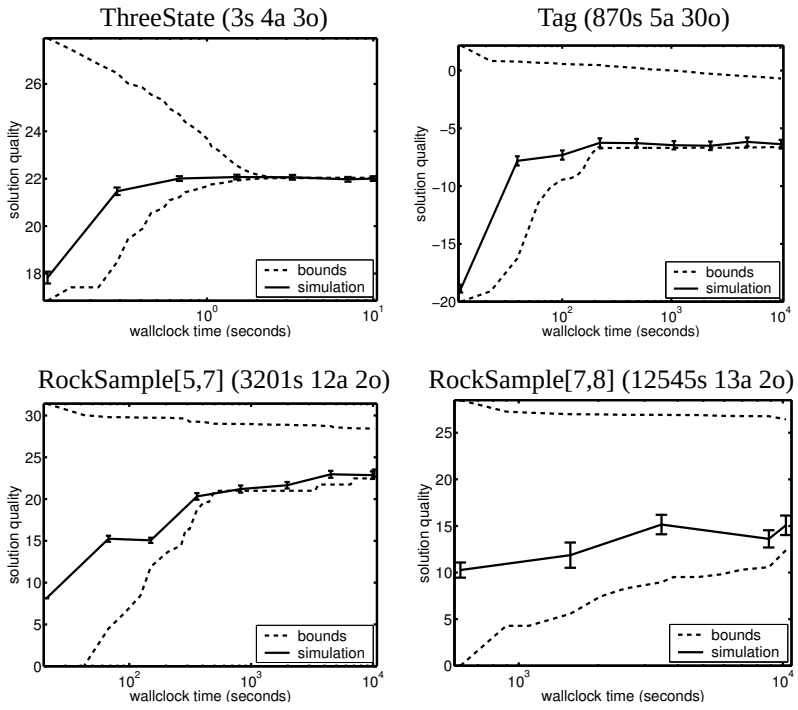

Figure 4.4: Solution quality vs. time.

rupted HSVI and simulated its current best policy π, providing an estimate of the solution quality, $J^{\pi}(b_0)$. The reported quality is the average reward received over many simulation runs (100-1000). Replicating earlier experiments, each simulation was terminated after 251 steps.

For each problem, results are reported over a single run of the algorithm. In a few cases we made multiple runs, but since HSVI is not stochastic, successive runs are identical up to minuscule changes arising from varying background load on the platform we used, a Pentium-III running at 850 MHz, with 256 MB of RAM.

Fig. 4.4 shows HSVI solution quality vs. time for four problems. In these plots, we also track the bounds $\underline{V}(b_0)$ and $\bar{V}(b_0)$. Recall that at every phase of the algorithm, we are guaranteed that $\underline{V}(b_0) \leq J^{\pi}(b_0) \leq \bar{V}(b_0)$. Fig. 4.4 should reflect this, at least up to the error in our estimate of $J^{\pi}(b_0)$ (errorbars are 95% confidence intervals). *ThreeState* is a trivial problem we generated, an example of HSVI running to convergence. On the larger problems, the bounds have not converged by the end of the run.

Fig. 4.4 shows running times and final solution quality for HSVI and some other state-of-the-art algorithms. Unfortunately, not all competitive algorithms could be included in the comparison, because there is no widely accepted POMDP benchmark that we could use. Results for algorithms other than HSVI were computed on different platforms; running times are only very roughly comparable. Among the algorithms compared, HSVI's final solution quality is in every case within measurement error of

the best reported so far, and in one case (the *Tag* problem) is significantly better.

Problem (num. states/actions/observations)	Goal%	Reward	Time (s)	$\|\Gamma\|$
Tiger-Grid (36s 5a 17o)				
QMDP [Pineau et al., 2003]	n.a.	0.198	0.19	n.a.
Grid [Brafman, 1997]	n.a.	0.94	n.v.	174
PBUA [Poon, 2001]	n.a.	2.30	12116	660
PBVI [Pineau et al., 2003]	n.a.	2.25	3448	470
HSVI	n.a.	2.35	10341	4860
Hallway (61s 5a 21o)				
QMDP [Littman et al., 1995]	47.4	n.v.	n.v.	n.a.
PBUA [Poon, 2001]	100	0.53	450	300
PBVI [Pineau et al., 2003]	96	0.53	288	86
HSVI	100	0.52	10836	1341
Hallway2 (93s 5a 17o)				
QMDP [Littman et al., 1995]	25.9	n.v.	n.v.	n.a.
Grid [Brafman, 1997]	98	n.v.	n.v.	337
PBUA [Poon, 2001]	100	0.35	27898	1840
PBVI [Pineau et al., 2003]	98	0.34	360	95
HSVI	100	0.35	10010	1571
Tag (870s 5a 30o)				
QMDP [Pineau et al., 2003]	17	-16.769	13.55	n.a.
PBVI [Pineau et al., 2003]	59	-9.180	180880	1334
HSVI	100	-6.37	10113	1657
RockSample[4,4] (257s 9a 2o)				
PBVI [Pineau, personal communication]	n.a.	17.1	$\sim$2000	n.v.
HSVI	n.a.	18.0	577	458
RockSample[5,5] (801s 10a 2o)				
HSVI	n.a.	19.0	10208	699
RockSample[5,7] (3201s 12a 2o)				
HSVI	n.a.	23.1	10263	287
RockSample[7,8] (12545s 13a 2o)				
HSVI	n.a.	15.1	10266	94

n.a. = not applicable n.v. = not available

Figure 4.5: Multi-algorithm performance comparison.

Fig. 4.4, which shows only a single time/quality data point for each problem, does not provide enough data for speed comparisons. Therefore we decided to make a closer comparison with one algorithm. PBVI was chosen both because it is a competitive algorithm, and because [Pineau et al., 2003] presents detailed solution quality vs. time curves for our benchmark problems.

In order to control for differing lengths of runs, we report the time that each algorithm took to reach a common value v_c, defined to be the highest value that *both* algorithms were able to reach at some point during their run. There is uncertainty asso-

Problem (num. states/actions/observations)	v_c	Time PBVI	Time HSVI	Speedup
Tiger-Grid (36 s 5a 17o)	2.25	3448	1053	3.3
Hallway (61s 5a 21o)	0.52	100-200	163	$\sim$1
Hallway2 (93s 5a 17o)	0.34	360	181	2.0
Tag (870s 5a 30o)	-9.18	180880	39	4600
RockSample[4,4] (256s 9a 2o)	17.1	$\sim$2000	23	$\sim$87

Figure 4.6: Performance comparison, HSVI and PBVI.

ciated with some of the times for PBVI because they were derived from manual reading of published plots; this uncertainty is noted in our comparison table. PBVI and HSVI appear to have been run on comparable platforms.[4]

Fig. 4.6 compares PBVI and HSVI performance. The two algorithms show similar performance on smaller problems. As the problems scale up, however, HSVI provides dramatic speedup. A brief explanation of why this might be the case: Recall that the policy returned by HSVI is based solely on the lower bound. The upper bound is used only to guide forward exploration. But upper bound updates, which involve the solution of several linear programs, often take longer than lower bound updates. Since PBVI keeps only a lower bound, its updates proceed much more quickly. HSVI can only have competitive performance to the extent that the intelligence of its heuristics outweighs the speed penalty of updating the upper bound. Apparently, the heuristics become relatively more important as problem size increases.

Because HSVI combines several existing solution techniques, it can be compared to a wide range of related work. Figure 4.7, although far from exhaustive, lists many relevant algorithms and some of the features they share with HSVI.

	Applied to POMDPs	Asynchronous updates	Examines only reachable states/beliefs	Uses action heuristic	Uses observation/outcome heuristic	Leverages value function convexity	Keeps upper and lower bounds
HSVI	Y	Y	Y	Y	Y	Y	Y
ICUB/ICUL [Hauskrecht, 1997]	Y	Y	-	Y	-	Y	Y
BI-POMDP [Washington, 1997]	Y	Y	Y	Y	Y	-	Y
RTDP-BEL [Geffner and Bonet, 1998]	Y	Y	Y	Y	-	-	-
[Brafman, 1997]	Y	Y	-	Y	Y	Y	-
[Dearden and Boutilier, 1994]	-	Y	Y	Y	-	-	Y
LAO* [Hansen and Zilberstein, 2001]	-	Y	Y	Y	-	-	Y
PBVI [Pineau et al., 2003]	Y	-	Y	-	-	Y	-
PBDP [Zhang and Zhang, 2001]	Y	-	-	-	-	Y	-
Incremental pruning [Cassandra et al., 1997]	Y	-	-	-	-	Y	-

Figure 4.7: Relevant algorithms and features.

[4] PBVI performance on *RockSample*$[4, 4]$ and a rough performance estimate for the computer used in PBVI experiments were provided courtesy of J. Pineau (personal communication).

4.5 HSVI Related Work

[Hauskrecht, 1997], perhaps the closest prior work, describes separate algorithms for incrementally calculating the upper bound (ICUB) and lower bound (ICLB). The ICUB upper bound is similar to that of HSVI in that it is initialized with the value function for the underlying MDP (V_{MDP}), improved with asynchronous backups, and used as an action heuristic. Unlike HSVI, ICUB uses a grid-based representation, and explores forward from belief space critical points rather than a specified initial belief. The ICUL lower bound uses the same vector set representation as HSVI and adds the result of each gradient backup in the same way. But because ICUB and ICUL are separate algorithms, ICUL's forward exploration does not select actions based on the upper bound, and neither algorithm makes use of an uncertainty-based observation heuristic.

Other related work mostly falls into two camps. The first are algorithms that combine heuristic search with dynamic programming updates. RTDP-BEL [Geffner and Bonet, 1998], a POMDP extension of the well-known RTDP value iteration technique for MDPs [Barto et al., 1995], turns out to be very similar to ICUB. BI-POMDP [Washington, 1997] uses forward exploration based on AO* with V_{MDP} as its heuristic. BI-POMDP keeps upper and lower bounds on nodes in the search tree—however, it does not explicitly represent the bounds as functions, so it is unable to generalize the value at a belief to neighboring beliefs. Some other algorithms in this group are [Dearden and Boutilier, 1994, Brafman, 1997, Hansen and Zilberstein, 2001].

The second camp includes algorithms that employ a piecewise linear convex value function representation and gradient backups. There are a host of algorithms along these lines, dating back to [Sondik, 1971]. Most differ from HSVI in that they perform gradient backups over the full belief space instead of focusing on relevant beliefs. One exception is PBVI [Pineau et al., 2003], which performs synchronous gradient backups on a growing subset of the belief space, designed such that it examines only reachable beliefs. Unlike HSVI, PBVI does not keep an upper bound and does not use a value-based action heuristic when expanding its belief set. Other algorithms in this group include incremental pruning [Cassandra et al., 1997] and point-based dynamic programming [Zhang and Zhang, 2001].

HSVI avoids examining unreachable beliefs using forward exploration. [Boutilier et al., 1998] describe how to precompute reachability in order to eliminate states in an MDP context. In a POMDP context their technique would go beyond HSVI by explicitly reducing the dimensionality of the belief space, but the remaining space might still include unreachable beliefs never visited by HSVI.

Finally, there are many competitive POMDP solution approaches that do not employ heuristic search or a PWLC value function representation: too many to discuss here. We refer the reader to a survey [Aberdeen, 2002]. Hopefully, increased adoption of common benchmarks in the POMDP community will allow us to better compare HSVI with other algorithms in the future.

5 Proposed Research

5.1 Science Autonomy

Much of the science autonomy research agenda was laid out in §3. This section explains the project context for the proposed work and provides a preliminary development plan.

5.1.1 Life in the Atacama

The first part of the proposed work is to assist in developing and field testing an SA system. We will create an overall SA architecture and develop the planning module in that architecture. This part of the work will be associated with the Life in the Atacama (LITA) project.

LITA is studying robotic astrobiology, searching for extremophile life in the Atacama Desert of Chile [Wettergreen et al., 2003]. It is part of the NASA ASTEP program, which seeks to perform useful science on Earth while working under operational constraints relevant to planetary exploration. The project includes three rover field expeditions in three years (April 2003, September 2004, September 2005). in 2005.

Each LITA field expedition is based around a single rover operating in the Atacama. The rover carries instruments to detect microorganisms and chlorophyll-based life forms and to characterize habitats. Instruments include panoramic imagers, fluorescence imagers for detection of chlorophyll and other biomolecules, spectrometers, as well as mechanisms for shallow subsurface access.

LITA's primary goals are:

- *Seek life:* Seek and characterize biota surviving in the Atacama and analyze microhabitats. We will question the hypothesis that the most arid regions of the Atacama represent an absolute desert.
- *Understand habitat:* Determine the physical and environmental conditions associated with identified past and current habitats, including the search for structural fossils, the monitoring of current biological oases and microorganic communities, and learning how these organisms have contributed to the modification of their environment.
- *Relevant science:* Develop, integrate, and field test a suite of science instruments that form a complete payload relevant to the NASA Mars Exploration Program and traceable to the Mars Exploration Program Payload Analysis group priority investigations and measurements that will facilitate the exploration of favorable environments for life on Mars in upcoming missions.

In the course of a day, the solar-powered rover can take hundreds of science instrument readings and autonomously navigate over distances of several kilometers. The focus of each expedition is a series of one-week periods of remote operations, during which a science team located in the U.S. controls the rover under operational constraints like those faced with Mars rovers, including:

- A limit of two communications windows per day with extremely limited bandwidth (on the order of 50 MB).

- No access to information about the exact location of the site or any previous surveys in that area.
- Maps similar to those we currently have of Mars, based on satellite or aerial imagery.

A field operations team is on site with the rover. Their primary role is to run preliminary tests that ensure the rover is ready for remote operations. There is a protocol that prohibits them from communicating with the science team or interfering with the rover (except in emergencies). Also on site is a ground truth team of scientists that follow the rover, and later assess the accuracy of the conclusions reached by the remote science team.

Because the rover is solar-powered and needs sunlight for navigation, most of its activities must be carried out during daylight hours. It has two communications windows with the science team, at the beginning and end of its day. The evening window is primarily a data downlink, during which the rover communicates its status and any science data. The morning window is primarily an uplink of goals for the day's operations.

For the 2004 expedition, remote operations will primarily use what was described in §3.1 as the baseline high-mobility operations strategy. Early versions of all of the software modules described in our SA architecture will be present. We may attempt some simple SA-relevant operations: for instance, performing follow-up spectrometer readings on a few interesting rocks in the end-of-day panorama.

For the 2005 expedition, we plan to develop and field test an SA system that implements the science on the fly and intelligent site survey SA operational modes.

5.1.2 Planner Implementations

Over the course of the proposed work, we will implement multiple versions of the SA planning module, to be applied in different ways.

September 2004 LITA Expedition. For use in the field, we will focus on quickly building a working version of the planner with baseline functionality. This version will not use a probabilistic model. It will be based on heuristic search in plan space, and we will likely limit the flexibility of the search. For example, we may require that science goals are pursued in the order that they were specified by the science team. Lower-level operations such as coverage patterns will be scripted. The design will be optimized for stability and ease of development and validation.

Ongoing research: We will produce a series of implementations for POMDP planning research. These algorithms will work with probabilistic models, and they will share many of the properties of HSVI, such as using forward search in belief space and keeping upper and lower bounds on the value function. They will proceed in several research directions as discussed earlier. These planners will be optimized for quick development and easy analysis of results. They will be tested in simulation, using probabilistic SA models based on experience from the 2004 LITA expedition.

September 2005 LITA Expedition: The planner deployed in 2005 will be based on the field experience in 2004 and research during the intervening period. Implementation will start by replicating the deterministic planning functionality of the 2004

planner, but based on an HSVI-like search strategy that is able to deal with probabilistic model elements, which we might add later. If time permits, we will migrate selected elements of the research systems into the field system. The design will be optimized for stability, clean extensible architecture, and ease of validation.

5.2 Probabilistic/POMDP Planning

§3.5 laid out several topics of research in POMDP planning. This section provides additional detail on one of those topics.

5.2.1 Extending Algebraic Decision Diagrams

Factored state can be used to develop efficient representations of the beliefs and α vectors used in POMDP value iteration. One such representation is algebraic decision diagrams (ADDs) [Bahar et al., 1993, Hoey et al., 1999]. ADDs are directed acyclic graphs that branch on boolean variables such as state elements. They generalize the ordered binary decision diagrams that have revolutionized circuit verification [Bryant, 1986]. ADDs can compactly represent state-indexed vectors that have uniform values over many entries (or nearly uniform values [St-Aubin et al., 2000]). We are investigating ADD extensions that can also compactly represent probability distributions with some conditional independence between state elements.

This section shows some examples to give a flavor for our concept, called the product ADD (PADD). Suppose that the world state includes three boolean variables X, Y, and Z. In the SA domain, each of these variables could represent whether a rock is an interesting carbonate (true) or an uninteresting background rock (false). It is reasonable to guess that these variables are independent (even if they are not really independent, we may model them that way because we do not know how they are correlated).

Any joint probability table (JPT) for the variables can be represented using an ADD. The probability of a joint assignment of the variables can be read from the ADD graph as follows:

1. Begin at the root.

2. Each time the graph branches on a variable, follow the right branch if the variable is true and the left branch if it is false.

3. When a terminal node is reached, its value is the probability of the joint assignment.

For example, suppose that the variables are independent with $\Pr(X) = 1$, $\Pr(Y) = 0.5$, and $\Pr(Z) = 0.9$. Fig. 5.1 shows two ADDs that represent this JPT. The ADD on the left is a fully expanded tree. You can verify, for instance, that the rightmost terminal node is $\Pr(X)\Pr(Y)\Pr(Z) = 0.45$. The ADD on the right is in *canonical* form, essentially meaning that the size of the graph has been minimized by sharing subgraphs and eliminating useless branch points (how to canonicalize an ADD is beyond our scope).

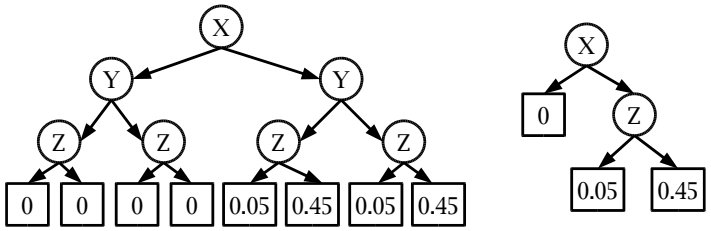

Figure 5.1: Representations of the first example JPT: (left) fully expanded ADD, (right) canonical ADD.

Note that simply listing the entries of the JPT in a vector would require eight entries, but the ADD representation on the right has only five nodes: the ADD has succeeded in compressing the JPT because many of its entries have the same value.

One may ask why we do not simply assume the variables are independent and take a list of their marginal probabilities as our representation. The short answer is that proving any kind of conditional independence between variables always holds in a POMDP model requires unreasonably strong assumptions about the structure of the model. However, we believe that a wider class of POMDP worlds spend most of their time in beliefs with some conditional independence. Therefore, we are interested in a representation that is both fully general and able to make use conditional independence when it exists.

Now suppose the variables are independent with $\Pr(X) = 0.9$, $\Pr(Y) = 0.3$, and $\Pr(Z) = 0.8$. The canonical ADD for this representation is shown on the left in fig. 5.2. Note that despite the independence of the variables, there are no repeated entries in the JPT, and no compression is achieved.

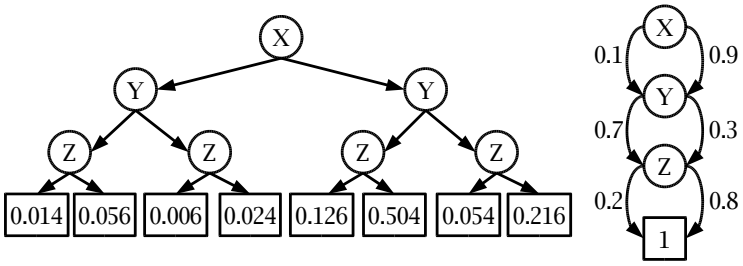

Figure 5.2: Representations of the second example JPT: (left) canonical ADD, (right) canonical PADD.

Our extension to the ADD framework, called the product ADD (PADD), is intended to handle JPTs like this second example. PADDs are optimized to represent the multiplicative structure of a JPT with some conditional independence. The canonical PADD for the second example is shown on the right in fig. 5.2. A JPT entry is read from the PADD as follows:

1. Begin at the root. Define a variable called *multiplier* and set it to 1.

2. Each time the graph branches on a variable, follow the right branch if the variable is true and the left branch if it is false. Either way, multiply the *multiplier* by the label on the branch followed.

3. When a terminal node is reached, the product of *multiplier* and the terminal node value is the probability of the joint assignment.

Note that the PADD has succeeded in compressing the second example.[5] It should be clear, based on the definition and the example, that the size of the PADD representation of a JPT scales linearly in the number of variables as long as complete independence is maintained. PADDs also provide some compression in the presence of various kinds of conditional independence.

The PADD multiplies by each branch label during descent through the graph. An alternative extension, called the sum ADD (SADD), adds each branch label. SADDs may provide a compact representation of the α vectors used in value iteration. Together, PADDs and SADDs make up a class we call extended ADDs (EADDs).

Our preliminary analysis indicates that EADDs essentially dominate ADDs; the number of nodes in an EADD is never larger than in the corresponding ADD, and it may be exponentially smaller. Basic operations such as pointwise multiplication of two JPTs are just as efficient (up to a constant factor) as the corresponding ADD operations on graphs with the same structure.

We propose to study how PADD performance degrades as independence is lost, how much independence is encountered in the beliefs of typical POMDP problems, and whether use of EADDs to represent beliefs and α vectors can speed up POMDP planning.

6 Contributions

The contributions of this thesis will come in two areas. First, we will develop an overall architecture and planning module for one of the first rover SA systems. Our work will be distinguished from comparable projects by the need to generate daily plans that include several kilometers of traverse, and by the primary mission goal, which is biology as opposed to geology (implying a different sensor suite). Second, we will extend POMDP planning techniques and apply them to the SA domain for the first time. We hope that the probabilistic models and algorithms we develop will both improve the quality of SA plans and generalize to other planning domains.

7 Schedule

Summer 04

- Develop deterministic model for LITA domain.

[5] For this small case, the compression is not very dramatic, since there appear to be seven numbers stored in the PADD representation, as opposed to eight naïvely. Larger examples see better compression. But note also that the left and right labels at a branch point always sum to 1, so we can explicitly store one label and infer the other.

- Develop planning module for 2004 expedition and a protocol for continuous planning interaction with the executive.

Fall 04

- Join field operations team during the second LITA expedition (September-October) and assist in analysis and write-up.
- Based on field experience with the deterministic model, develop POMDP models of LITA domain.
- Study factored state and decide feasibility of incrementally adding POMDP model elements to deterministic models.

Spring-Summer 05

- Study POMDP planning techniques.
- Develop planning module for 2005 expedition.

Fall 05

- Join field operations team during the third LITA expedition (September-November) and assist in analysis and write-up.

Spring-Summer 06

- Study POMDP planning techniques.
- Test POMDP planner in simulation using field-validated probabilistic model of the LITA domain.
- Complete written thesis and defend.

References

[Aberdeen, 2002] Aberdeen, D. (2002). A survey of approximate methods for solving partially observable Markov decision processes. Technical report, Research School of Information Science and Engineering, Australia National University.

[Astrom, 1965] Astrom, K. J. (1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205.

[Bagnell and Schneider, 2001] Bagnell, D. and Schneider, J. (2001). Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings of the International Conference on Robotics and Automation*. IEEE.

[Bahar et al., 1993] Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., and Somenzi, F. (1993). Algebraic decision diagrams and their applications. In *Proc. of ICCAD*.

[Barto et al., 1995] Barto, A., Bradtke, S., and Singh, S. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138.

[Baxter and Bartlett, 2000] Baxter, J. and Bartlett, P. L. (2000). Reinforcement learning on POMDPs via direct gradient ascent. In *ICML*.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, NJ.

[Bonasso, 1991] Bonasso, R. P. (1991). Integrating reaction plans and layered competences through synchronous control. In *Proc. of IJCAI*.

[Boutilier et al., 1998] Boutilier, C., Brafman, R. I., and Geib, C. W. (1998). Structured reachability analysis for markov decision processes. In *Proc. of UAI*, pages 24–32.

[Brafman, 1997] Brafman, R. I. (1997). A heuristic variable grid solution method for POMDPs. In *Proc. of AAAI*.

[Bryant, 1986] Bryant, R. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691.

[Cabrol et al., 2001] Cabrol, N. A., Chong-Diaz, G., Stoker, C. R., Gulick, V. C., Landheim, R., Lee, P., Roush, T. L., Zent, A. P., Lameli, C. H., Iglesia, A. J., Arrerondo, M. P., Dohm, J. M., Keaten, R., Wettergreen, D., Sims, M., Pedersen, L., Bettis, A., Thomas, G., and Witzke, B. (2001). Nomad rover field experiment, Atacama Desert, Chile, 1, science results overview. *J. Geophys. Res.*, 106(E4):7785.

[Cassandra et al., 1997] Cassandra, A., Littman, M., and Zhang, N. (1997). Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proc. of UAI*.

[Cassandra et al., 1996] Cassandra, A. R., Kaelbling, L., and Kurien, J. A. (1996). Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proc. of IROS*.

[Cassandra et al., 1994] Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1023–1028, Seattle, Washington, USA. AAAI Press/MIT Press.

[Castaño et al., 2003] Castaño, R., Anderson, R. C., Estlin, T., DeCoste, D., Fisher, F., Gaines, D., Mazzoni, D., and Judd, M. (2003). Rover traverse science for increased mission science return. In *Proc. of IEEE Aerospace*, Big Sky, Montana.

[Cheeseman and Stutz, 1996] Cheeseman, P. and Stutz, J. (1996). Bayesian classification (AutoClass): Theory and results. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press.

[Chien et al., 2003] Chien, S., Sherwood, R., Tran, D., Castaño, R., Cichy, B., Davies, A., Rabideau, G., Tang, N., Burl, M., Mandl, D., Frye, S., Hingemihle, J., D'Augustino, J., Bote, R., Trout, B., Schulman, S., Ungar, S., Gaasback, J. V., Boyer, D., Griffin, M., Burke, H., Greeley, R., Doggett, T., Williams, K., Baker, V., and Dohm, J. (2003). Autonomous science on the EO-1 mission. In *Proc. of iSAIRAS*, Nara, Japan.

[Choset, 2001] Choset, H. (2001). Coverage for robotics—a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126.

[Dearden and Boutilier, 1994] Dearden, R. and Boutilier, C. (1994). Integrating planning and execution in stochastic domains. In *Proc. of the AAAI Spring Symposium on Decision Theoretic Planning*, pages 55–61, Stanford, CA.

[Dearden et al., 2003] Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D., and Washington, R. (2003). Incremental contingency planning. In *Proc. of ICAPS Workshop on Planning under Uncertainty*.

[Estlin et al., 2003] Estlin, T., Castaño, R., Anderson, B., Gaines, D., Fisher, F., and Judd, M. (2003). Learning and planning for mars rover science. In *Proc. of IJCAI Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World Modeling, Planning, Learning, and Communicating*, Acapulco, Mexico.

[Estlin et al., 2002] Estlin, T., Fisher, F., Gaines, D., Chouinard, C., Schaffer, S., and Nesnas, I. (2002). Continuous planning and execution for an autonomous rover. In *Proc. of the International NASA Workshop on Planning and Scheduling for Space*, Houston, TX.

[Estlin and Gaines, 2002] Estlin, T. and Gaines, D. (2002). An optimization framework for interdependent planning goals. In *Proc. of the AIPS Workshop for Planning and Scheduling with Multiple Criteria*, Toulouse, France.

[Estlin et al., 1999] Estlin, T. A., Mann, T. P., Gray, A. G., Rabideau, G., Castaño, R., Chien, S., and Mjolsness, E. D. (1999). An integrated system for multi-rover scientific exploration. In *AAAI*.

[Firby, 1989] Firby, R. J. (1989). *Adaptive Execution in Dynamic Domains*. PhD thesis, Yale University. YALEU/CSD/RR#672.

[Gazis and Roush, 2001] Gazis, P. R. and Roush, T. (2001). Autonomous identification of carbonates using near-IR reflectance spectra during the February 1999 Marsokhod field tests. *J. Geophys. Res.*, 106(E4):7765.

[Geffner and Bonet, 1998] Geffner, H. and Bonet, B. (1998). Solving large POMDPs by real time dynamic programming. In *Working Notes Fall AAAI Symposium on POMDPs*.

[Gulick et al., 2000] Gulick, V. C., Morris, R. L., Bandari, E. B., and Roush, T. L. (2000). Maximizing science return from future Mars missions with onboard image analyses. In *Proc. of Lunar and Planetary Science*, Lunar and Planetary Institute, Houston.

[Gulick et al., 2001] Gulick, V. C., Morris, R. L., Ruzon, M. A., and Roush, T. L. (2001). Autonomous image analyses during the 1999 Marsokhod rover field test. *J. Geophys. Res.*, 106(E4):7745.

[Hansen, 1998] Hansen, E. (1998). Solving POMDPs by searching in policy space. In *Proc. of UAI*, Madison, Wisconsin.

[Hansen and Zilberstein, 2001] Hansen, E. and Zilberstein, S. (2001). LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129:35–62.

[Hauskrecht, 1997] Hauskrecht, M. (1997). Incremental methods for computing bounds in partially observable Markov decision processes. In *Proc. of AAAI*, pages 734–739, Providence, RI.

[Hauskrecht, 2000] Hauskrecht, M. (2000). Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94.

[Hoey et al., 1999] Hoey, J., St-Aubin, R., Hu, A., and Boutilier, C. (1999). SPUDD: Stochastic planning using decision diagrams. In *Proc. of UAI*, pages 279–288.

[Howard, 1960] Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT.

[Kaelbling, 1993] Kaelbling, L. P. (1993). *Learning in Embedded Systems*. The MIT Press.

[Kaelbling et al., 1998] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

[Littman, 1994] Littman, M. L. (1994). The witness algorithm: Solving partially observable Markov decision processes. Technical Report CS-94-40, Brown University, Providence, RI.

[Littman, 1996] Littman, M. L. (1996). *Algorithms for Sequential Decision Making*. PhD thesis, Brown University.

[Maimone et al., 1999] Maimone, M., Nesnas, I. A. D., and Das, H. (1999). Autonomous rock tracking and acquisition from a Mars rover. In *Proc. of iSAIRAS*, pages 329–334, Noordwijk, The Netherlands.

[Moorehead, 2001] Moorehead, S. (2001). *Autonomous Surface Exploration for Mobile Robots*. PhD thesis, Robotics Institute, Carnegie Mellon University. CMU-RI-TR-01-30.

[Nourbakhsh, 1997] Nourbakhsh, I. (1997). *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers.

[Pedersen, 2000] Pedersen, L. (2000). *Robotic Rock Classification and Autonomous Exploration*. PhD thesis, Robotics Institute, Carnegie Mellon University. CMU-RI-TR-01-14.

[Pedersen et al., 2003a] Pedersen, L., Bualat, M., Kunz, C., Lee, S., Sargent, R., Washington, R., and Wright, A. (2003a). Instrument deployment for Mars rovers. In *Proc. of ICRA*, pages 2535–2542.

[Pedersen et al., 2003b] Pedersen, L., Bualat, M., Lees, D., Smith, D., and Washington, R. (2003b). Integrated demonstration of instrument placement, robust execution and contingency planning. In *Proc. of iSAIRAS*, Nara, Japan.

[Pedersen et al., 2001] Pedersen, L., Wagner, M. D., Apostolopoulos, D., and Whittaker, W. L. (2001). Autonomous robotic meteorite identification in Antarctica. In *Proc. of International Conference on Robotics and Automation*, pages 4158–4165.

[Pineau et al., 2003] Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: an anytime algorithm for POMDPs. In *IJCAI*.

[Poon, 2001] Poon, K.-M. (2001). A fast heuristic algorithm for decision-theoretic planning. Master's thesis, The Hong Kong University of Science and Technology.

[Shillcutt, 2000] Shillcutt, K. (2000). *Solar Based Navigation for Robotic Explorers*. PhD thesis, Robotics Institute, Carnegie Mellon University. CMU-RI-TR-00-25.

[Simmons and Koenig, 1995] Simmons, R. and Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. In *Proc. of IJCAI*, pages 1080–1087.

[Smith, 2003] Smith, T. (2003). Science autonomy in the Atacama. In *Proc. of ICML*.

[Smith and Simmons, 2004] Smith, T. and Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *Proc. of UAI*. (to appear; a companion technical report with more detailed theoretical analysis is currently in preparation).

[Sondik, 1971] Sondik, E. J. (1971). *The optimal control of partially observable Markov processes*. PhD thesis, Stanford University.

[St-Aubin et al., 2000] St-Aubin, R., Hoey, J., and Boutilier, C. (2000). APRICODD: Approximate policy construction using decision diagrams. In *Proc. of NIPS*, pages 1089–1095, Denver, CO.

[Thrun, 2000] Thrun, S. (2000). Monte carlo POMDPs. In *Proc. of NIPS*, pages 1064–1070. MIT Press.

[Tompkins et al., 2002] Tompkins, P., Stentz, A., and Whittaker, W. L. (2002). Mission planning for the sun-synchronous navigation field experiment. In *Proc. of International Conference on Robotics and Automation*.

[Tompkins et al., 2004] Tompkins, P., Stentz, A. T., and Whittaker, W. R. L. (2004). Field experiments in mission-level path execution and re-planning. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8)*.

[Urmson et al., 2002] Urmson, C., Dias, M. B., and Simmons, R. (2002). Stereo vision based navigation for sun-synchronous exploration. In *Proc. of IROS*.

[Wagner et al., 2001] Wagner, M. D., Apostolopoulos, D., Shillcutt, K., Shamah, B., Simmons, R., and Whittaker, W. L. (2001). The science autonomy system of the Nomad robot. In *Proc. of International Conference on Robotics and Automation*, pages 1742–1749.

[Washington, 1997] Washington, R. (1997). BI-POMDP: Bounded, incremental, partially-observable Markov-model planning. In *Proc. of European Conf. on Planning (ECP)*, Toulouse, France.

[Wettergreen et al., 2003] Wettergreen, D., Cabrol, N., Calderón, F., Jonak, D., Lüders, A., Pederson, K., Shaw, F., Smith, T., Strelow, D., Teza, J., Tompkins, P., Urmson, C., Verma, V., and Wagner, M. (2003). Life in the Atacama field season 2003: Experiment plans and technical results. Technical Report CMU-RI-TR-03-50, Robotics Institute, Carnegie Mellon University.

[Zhang and Zhang, 2001] Zhang, N. L. and Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of AI Research*, 14:29–51.