Comparing Action-Query Strategies in Semi-Autonomous Agents

Robert Cohn
Computer Science and Engineering
University of Michigan
rwcohn@umich.edu

Edmund Durfee
Computer Science and Engineering
University of Michigan
durfee@umich.edu

Satinder Singh
Computer Science and Engineering
University of Michigan
baveja@umich.edu

Abstract

We consider settings in which a semi-autonomous agent has uncertain knowledge about its environment, but can ask what action the human operator would prefer taking in the current or in a potential future state. Asking queries can improve behavior, but if queries come at a cost (e.g., due to limited operator attention), the value of each query should be maximized. We compare two strategies for selecting action queries: 1) based on myopically maximizing expected gain in long-term value, and 2) based on myopically minimizing uncertainty in the agent's policy representation. We show empirically that the first strategy tends to select more valuable queries, and that a hybrid method can outperform either method alone in settings with limited computation.

1 Introduction

A semi-autonomous agent acting in a sequential decision-making environment should act autonomously whenever it can do so confidently, and seek help from a human operator when it cannot. We consider settings where querying the operator is expensive, for example because of communication or attentional costs, and design algorithms to find the best query to ask the operator. Operator responses to agent queries can improve the agent's ability to behave as the operator would. Of the many types of queries one could consider asking the operator, action-queries (asking what action the operator would take if teleoperating the agent in a particular state) are arguably quite natural for a human to respond to. Our goal, then, is to design an agent that can (1) select which action-queries are most useful for efficiently mimicking operator teleoperation, and (2) elect not to query when its cost exceeds its benefit. Here we focus on (1), while our previous work (Cohn et al. 2010) contains insights addressing (2).

When teleoperating, the operator chooses actions according to her model of the world. We assume that the agent fully knows the operator's model of world dynamics, but has an incomplete model of the operator's rewards, and thus risks acting counter to the operator's true rewards. The agent represents its uncertainty as a probability distribution over reward functions, and the *only* information it can acquire to improve its behavior is operator responses to its queries.

Our objective is for the agent to identify the query that will maximize its gain in expected long-term value with respect to the operator's true rewards and the agent's current state. This objective is myopic because it ignores future queries that could be made, such as if the agent could ask a sequence of queries, or wait to query later. Although desirable, non-myopic optimization would require solving an intractable sequential decision-making problem (see Duff 2003) to find an optimal action-query policy.

Our contributions in this paper are threefold. First, we adapt the Expected Myopic Gain (EMG) algorithm (Cohn et al. 2010), which had hitherto been applied to selecting transition-probability and reward queries, to create a new EMG algorithm for **A**ction-**Q**uery **S**election (EMG-AQS). Second, we compare our algorithm with an alternative algorithm called Active Sampling (AS) (Lopes, Melo, and Montesano 2009) that applies directly to our problem setting, but selects action-queries based on maximally reducing uncertainty in policy representations. EMG-AQS is computationally expensive but directly optimizes our objective, while AS is computationally inexpensive but only indirectly optimizes our objective. We exploit this fact through our third contribution, which is a new hybrid algorithm combining EMG-AQS and AS, that we show in computation-time-limited problems to perform better than either method alone.

This paper is organized as follows. We provide background on the EMG and AS algorithms, derive our adaptation of EMG to action-query evaluation, and then provide an empirical comparison spanning two domains, where the second motivates the introduction of our hybrid algorithm.

2 Related Work

Our problem is related to that of apprenticeship learning (Abbeel and Ng 2004) where the agent is provided with a trajectory of teleoperation experience, and charged with learning by generalizing that experience. The difference is that rather than *passively* obtaining teleoperation experience, our agent is responsible for *actively* requesting such information. In our setting, the agent can even ask about potential future states that may never be experienced.

Chernova and Veloso (2009) address a version of the action-query selection problem, where unlike our setting, the agent has neither a model of dynamics nor a prior distribution over reward functions, and decides *whether or not* to

query *only* the current state. Since our focus is on selecting the best action-query from a *space* of possible action-queries, we do not include their method in our comparisons.

Our problem is related to preference elicitation in the presence of utility uncertainty (Chajewska, Koller, and Parr 2000). In fact, our action-queries can be viewed as choice queries as in Viappiani and Boutilier (2010), but in a sequential decision making setting. Indirect information provided by action-queries in this setting requires the application of Bayesian Inverse Reinforcement Learning to update the agent's reward belief, which we describe in Section 4.

More generally, our problem is related to Active Learning (Cohn, Atlas, and Ladner 1994), where the learner seeks to induce the correct hypothesis that maps datapoints to a label space, and may request labels for datapoints. However, our MDP setting introduces significant structure and emphasizes that the agent maximize its long-term value at its current state, as opposed to its expected label accuracy.

3 Background

MDPs and Bayesian MDPs: In a Markov Decision Process (MDP), the agent's environment is modeled as a tuple $M = \langle S, A, T, R \rangle$ where the components represent the state space, action space, transition function, and reward function, respectively. At each time step, the agent observes its current state s, takes action a, probabilistically transitions to state s' according to $T(s'|s,a)$, and receives reward $R(s')$. A policy π maps $S \rightarrow A$, and the expected value of acting according to π in MDP M beginning from state s is denoted $V_M^\pi(s)$ and defined as $\mathbb{E}[\sum_{t=1}^{\infty} \gamma^t r_t]$, where γ is a discount factor $\in [0,1]$ and r_t is the reward received at time t. The optimal policy for MDP M, π_M^*, can be computed by algorithms such as value or policy iteration.

In this paper, we focus on the case of uncertainty only over a discrete set of reward functions. The agent represents its uncertainty as a probability distribution over possible reward functions and thus over MDPs. We denote that distribution, or interchangeably the parameters that define that distribution, as ψ. For state s under a policy π, the expected value over distribution ψ is the expected discounted sum of rewards when the MDP is drawn from ψ, the start state is s, and the agent behaves according to policy π, i.e., $\mathbb{E}_{M\sim\psi}[V_M^\pi(s)]$ where $M \sim \psi$ denotes a random MDP M drawn from ψ. The *Bayes-optimal* policy is optimal in expectation over ψ, denoted π_ψ^*, and satisfies $\pi_\psi^*(s) = \arg\max_\pi \mathbb{E}_{M\sim\psi}[V_M^\pi(s)]$.

Active Sampling

Lopes et al. (2009) introduce Active Sampling (AS), a method that selects action queries in a Bayesian MDP setting with a distribution over reward functions but knowledge of all other components of the MDP. Intuitively, AS reduces the agent's uncertainty about the operator's policy by querying the state that has maximum mean entropy (uncertainty) in its action choices. Next we describe AS in more detail.

Each MDP has a stochastic optimal policy (stochastic only in that ties between actions are broken with uniform probability), and the action-choice probability for a state for each action is binned into K uniform intervals between 0 and 1. Given a current distribution over rewards ψ, the mean-entropy for state s, $\bar{H}(s)$, is given by

$$\bar{H}(s) = -\frac{1}{|A|} \sum_{a\in A} \sum_{k=0}^{K-1} \mu_{sa}(k) log\, \mu_{sa}(k), \text{ where}$$

$$\mu_{sa}(k) = \sum_{r\in R} \Big\{ \pi^*(a|r;s) \in I_k \Big\} Pr(r|\psi).$$

Here $I_k = (\frac{k}{K}, \frac{k+1}{K}]$ for $k \neq 0$ and $I_0 = [0, \frac{1}{K}]$, $Pr(r|\psi)$ is the probability of reward function r given distribution ψ over rewards, and $\{\pi^*(a|r;s) \in I_k\}$ is 1 if the probability of action a in state s under the optimal policy given r falls in the interval I_k and 0 otherwise. The AS algorithm queries the state with maximum mean-entropy.

Note that the dynamics of the world may dictate that some states are less likely to be reached than others, especially when taking into account the agent's current state. Also, taking the wrong action in some states may be catastrophic, whereas in others benign. Directly reducing policy uncertainty does not consider these factors, and thus is only a proxy for achieving our objective.

Expected Myopic Gain

Expected Myopic Gain (EMG) (Cohn et al. 2010) is a myopic Expected Value of Information (EVOI) technique for assessing the goodness of a query in terms of how much long-term value the agent is expected to gain from it. EMG can be split into two parts. First, for a query q (in general about any aspect informative about the distribution over MDPs) in current state s_c, define the expected gain if the answer to the query is o as follows:

$$\begin{aligned} Gain(\langle q,o\rangle|\psi,s_c) &= \mathbb{E}_{M\sim\psi_{post}}[V_M^{\pi^*_{\psi_{post}}}(s_c)] \\ &\quad - \mathbb{E}_{M\sim\psi_{post}}[V_M^{\pi^*_{\psi}}(s_c)], \end{aligned} \quad (1)$$

where ψ is the agent's knowledge before the query and $\psi_{post} = \psi, \langle q,o\rangle$ is the agent's knowledge after receiving response o to query q. Intuitively, Equation 1 states that the value of knowing that o is the answer to query q is the difference between the expected value at the current state of the policy calculated according to the new information and the policy calculated beforehand, evaluated on the MDP distribution induced by the new information.

Since the agent does not know which response $o \in O$ it will receive to its query q, EMG computes an expectation over the possible responses to q:

$$EMG(q|\psi,s_c) = \int_o Gain(\langle q,o\rangle|\psi,s_c)\, P(o\,|\,\psi;q) do. \quad (2)$$

The query selected is $\arg\max_{q\in Q} EMG(q|\psi,s_c)$ which will, in expectation, most increase the agent's long-term value (Cohn et al. 2010). Thus EMG achieves our objective, but in our previous work we only showed how to compute EMG for transition queries and reward queries, which give direct information about the MDP's transition or reward function respectively, and did not address the more challenging case of the more natural action-queries, which reveal indirect information about the underlying reward function.

4 EMG-based Action-Query Selection

Adapting the EMG algorithm of Equations 1 and 2 for action-queries requires a method to update the distribution over reward functions, given the operator's preferred action at a state. This is exactly the problem solved in Bayesian Inverse Reinforcement Learning (BIRL) (Ramachandran and Amir 2007), and like Lopes et al. (2009) we use BIRL to perform belief updates over the reward space.

In BIRL, the starting assumption is a noisy-model of the operator's action selection. For the MDP given by reward function $r \in R$, there is an associated action-value function Q^* such that $Q^*(s, a, r)$ is the expected long-term value obtained when the start state is s, the first action is a, and the optimal policy is followed thereafter. The operator is assumed to respond with action a to an action-query for state s with probability

$$P(a|r; s) = \frac{1}{Z_r} e^{\alpha Q^*(s,a,r)}, \tag{3}$$

where Z_r is the normalization term and α is a noise (or confidence) parameter such that the larger the α, the more confident the agent is that the response received is indeed optimal for the operator. Setting α lower can help in situations in which the operator's responses are noisy, or inconsistent with respect to all rewards in the reward space; this is a departure from our assumption in Cohn et al. (2010) that the operator acts optimally with respect to at least one model within the agent's MDP distribution. Given a response a to query s, and a current distribution over rewards ψ, the posterior distribution over rewards is defined by the Bayes update $P(r|\psi, a; s) = \frac{1}{Z} P(a|r; s)P(r|\psi)$, where again Z is the appropriate normalization term.

We have shown how to utilize BIRL to perform posterior updates given query responses, which allows the calculation of the distribution ψ_{post} from distribution ψ and query-response pair $\langle s, a\rangle$. We now turn to simplifying the terms of Equation 1, for which the agent must compute the expected value of a policy over ψ for the current state, as well as compute the Bayes-optimal policy for ψ. For general sources of uncertainty this would be a computationally prohibitive calculation, but in our case of uncertainty only in rewards, drastic simplification applies. Specifically, the expected value of a policy over a reward distribution is its value for the single *mean*-reward function $\bar{\psi}$, which implies that the Bayes-optimal policy for ψ is the optimal policy for $\bar{\psi}$. (See proof of Thm. 3 in Ramachandran and Amir 2007.)

Thus, Equation 1 can be simplified as

$$Gain(\langle s, a\rangle|\psi, s_c) = V^{\pi^*_{\bar{\psi}_{post}}}_{\bar{\psi}_{post}}(s_c) - V^{\pi^*_{\bar{\psi}}}_{\bar{\psi}_{post}}(s_c).$$

With Equation 1 made tractable, we examine $P(a|\psi; s)$ in Equation 2, which represents the probability that the operator would respond with a for query s given ψ:

$$P(a|\psi; s) = \sum_{r\in R} P(a|\psi, r; s)P(r|\psi) = \sum_{r\in R} P(a|r; s)P(r|\psi).$$

Notice that we already defined $P(a|r; s)$ using the above model, and that $P(r|\psi)$ is the prior probability of reward function r. This completes the description of how the agent updates its reward function distribution after each query, and utilizes the properties of reward function distributions to compute Equations 1 and 2. Our resulting algorithm is called **EMG**-based **A**ction-**Q**uery **S**election (EMG-AQS).

Reward Parameter Distributions. Parameterizing the reward function allows for a compact representation of the reward function distribution, even when defined over an infinite state space. Consider a distribution over a discrete parameter space W and a function f that maps $w \in W$ to a reward function $R_w \in R$. We can use the mean reward function for computing optimal policies and value functions with respect to a reward distribution, but we can use the reward function associated with the mean reward parameters instead only when $\mathbb{E}[f(w)] = f(\mathbb{E}[w])$ (an extension of Theorem 3 in Ramachandran and Amir 2007). Due to linearity of expectation, this equality holds when $f(w)$ is a linear function of w, but not necessarily otherwise. Therefore we use the mean reward parameters when this equality holds, but otherwise use the mean reward function, calculated as $\mathbb{E}[f(w)] = \sum_{w\in W} f(w)P(w)dw$. We make use of parameterized reward functions in all experiments in this paper.

Finite Reward Sets. We have assumed a finite reward function set, and accordingly, maintain a categorical reward parameter distribution which we update exactly according to the equations presented above. The finiteness of the reward set allows EMG-AQS and AS to precompute (or cache) value functions for each reward parameter, resulting in substantial computational savings when updating the reward distribution. Although continuous reward spaces can account for more behaviors, and thus in theory fit agent behavior more closely to operator behavior, performing BIRL in continuous reward spaces comes at a significant computational cost. (See Lopes et al. 2009 and Ramachandran and Amir 2007 for Monte Carlo methods for approximating BIRL in continuous reward spaces.)

5 Comparisons

We now empirically compare the relative suitability of EMG-AQS versus AS for choosing the most valuable queries to pose to the operator. *Prima facie*, one might expect EMG-AQS to perform better, as it is Bayes-optimal (Cohn et al. 2010) for selecting a single query. However, neither method is Bayes-optimal when selecting a sequence of queries, or when limited computation time is available so that only some of the possible queries can be evaluated. We test the former condition in our first experiment, the latter condition in our second experiment, and both combined in our third experiment.

The principal metric of comparison that we use is *policy loss*, which is the difference in long-term value for the current state between the operator's policy and the policy based on uncertain knowledge (evaluated on the operator's rewards). In the experiments that follow, we report average policy loss over trials (error bars shown are 95% confidence intervals), where each trial uses a reward function whose parameters are uniformly randomly drawn from the uniform-prior reward parameter space.

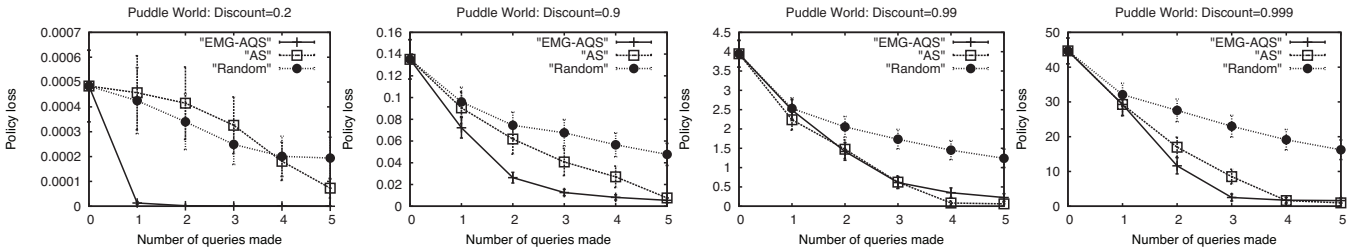

Figure 1: Average policy loss for EMG-AQS, AS, and Random on the Puddle World across a sequence of queries shown, from left to right, for discount 0.2, 0.9, 0.99, and 0.999. Error bars are 95% confidence intervals.

Puddle World

We conducted our first set of experiments in the Puddle World, as used in the evaluations of Lopes et al. (2009). The puddle world is a 21x21 gridworld, and so has 441 states (locations). The agent can move (deterministically) in any of the 4 cardinal directions to a neighboring state (unless it bumps into a grid edge in which case it does not change state). The agent acts over an infinite time horizon, with discount factor $\gamma \in [0, 1)$. The reward function is $f(w, s) = r_1 \exp(\frac{-||s_1 - s||}{2\sigma^2}) + r_2 \exp(\frac{-||s_2 - s||}{2\sigma^2})$, where s_1,s_2 in w represent the centers of reward emitting regions, and r_1,r_2 in w constitute the sign and magnitude of reward emitted from each. The pre-set scalar σ determines the rate of decay of the magnitude of reward emitted from each center. The parameterization permits cases with one "goal" and one "puddle" (one of the r parameters is positive and the other is negative), and similarly two goals or two puddles.

As in Lopes et al.'s experiments, our agent's start state is always $(0, 0)$. While their formulation uses a continuous state space, their deterministic direction and distance of movement actions resulted in a *de facto* discretization of space. Thus, we explicitly represent the state space discretely, which also allows for much more accurate and rapid Q-value approximations than would be computed through function approximation. We discretized the parameter space to allow the reward decay scalars to independently take on values -1 or 1 and the centers of the reward emitting regions to be located on a 5x5 discretization of the state space. This results in a set of reward parameters of size 2500.

The "operator" in this experiment is modeled as the optimal policy given the actual reward parameters for the particular trial: a response to a query is the action in this policy corresponding to the state being asked about.

Experiment 1. Here we perform our comparison across a space of discount factors. Since the agent's value is always measured with respect to the start state, a smaller discount factor reduces the benefit of collecting rewards in the future – good queries will inform the agent how to behave near the start state. As the discount factor increases, however, rewards the agent receives in the future are increasingly important. We test over a range of discounts to gain insights into each method's proficiency in exploiting domain properties.

Figure 1 shows the performance of each method for a sequence of queries, across discount factors 0.2, 0.9, 0.99, and 0.999. Note the different scales of the *y-axes*: a smaller discount factor leads to lower policy values, which in turn leads to lower policy loss magnitudes.

On all graphs, EMG-AQS shows the best performance for its first query. This is expected because EMG-AQS's first query choice is Bayes-optimal, so no method can do better in expectation for the first query. For the low discount factor of 0.2, EMG-AQS significantly outperforms AS across the entire sequence. In fact, AS's performance is comparable to Random, a strategy that simply selects random queries. Because EMG-AQS directly aims to increase the agent's expected discounted reward, it outperforms AS which instead improves decisions that are most uncertain regardless of when those decisions might be needed. Similarly, for discount 0.9, which is the exact (discretized) setting Lopes et al. (2009) use to test their method, EMG-AQS outperforms AS for queries 2-4.

We expected that by raising the discount factor closer to 1, the relative benefits of EMG-AQS would diminish. We indeed see that with a discount of 0.99, the difference between EMG-AQS and AS shrinks. In fact, AS outperforms EMG-AQS for queries 4 and 5. This is not entirely surprising: both approaches make myopic decisions based on different criteria, so even though EMG-AQS will greedily make Bayes-optimal decisions at each point in the query sequence, the combination of queries asked by AS can be better.

Surprisingly, though, this trend does not persist as we continue to raise the discount factor to 0.999: EMG-AQS once again outperforms AS. By investigating the queries being asked, we discovered that the explanation for this is essentially the other side of the coin for the larger discount factors. That is, when the discount factor is high enough, the penalties incurred by, for example, wandering too near the puddle on the way to the goal location become negligible compared to the accumulation of reward the agent gets as it hovers around the goal. Because of this, EMG-AQS focuses on queries that effectively map the goal region rather than the puddle region, while AS is not biased either way.

Hence, in the Puddle World, EMG-AQS effectively biases its queries towards the start state (for low discounts) or towards finding the best region to occupy in steady state (for high discounts), while AS is more evenhanded. These results suggest that EMG-AQS often makes better choices by exploiting domain properties, though in a balanced situation where no obvious properties can be exploited, AS can be better. Our next experiments additionally account for the computational efficiency of the methods.

Driving Domain

To test the robustness of the trends we observed in the Puddle World, and to take computation efficiency into account, we ran additional comparisons between EMG-AQS and AS in our implementation of the Driving Domain (Figure 2a), which is a traffic navigation problem often used in studies of apprenticeship learning (Abbeel and Ng 2004).

Dynamics. At each discrete time step the agent controls a car on a highway by taking one of three actions: move left, null action, or move right. Attempting to move off the edge results in a null action. Other cars are present, which move at random continuously-valued constant speeds (this makes the state space infinite) and never change lanes. The agent lacks an explicit transition function, but can simulate to sample trajectories influenced by its choices of actions.

State Features. The agent represents state as a vector of 68 features, consisting of three parts. The first has 5 binary features to specify the agent's current lane. The second contains 3 binary features to specify whether the agent is colliding with, tailgating, or trailing another car. The third part consists of 3 sets of features, one for the agent's current lane, and one for each adjacent lane, each containing 20 binary features specifying whether the agent's car (disregarding lane) will collide with another car in $2X$ time steps, where X ranges from 0 to 19. This part encodes the agent's view of traffic, taking into account car velocities.

Rewards. The reward function is parameterized by a weight vector w as $f(w, s) = \phi(s) \cdot w$, where $\phi(s)$ is a vector of reward features corresponding to state s. In particular, $\phi(s)$ contains four binary features, which specify whether the agent is colliding with, tailgating or trailing another car, and whether the agent is currently driving on the shoulder. In our experiments, the agent has *a priori* knowledge that the reward weight corresponding to driving on the shoulder is -0.2, but must learn which of 1000 possible assignments of the other parameters best encodes operator preferences.

Practicality of Action-Queries. Since in our evaluations the "operator" is a policy, queries can be answered by invoking the operator policy. However, in a practical scenario with a human operator, the agent should translate its representation of state into one that a human could more easily understand. Since a hypothetical state includes information about car positions and velocities as well as the agent car's position, the agent could present a state to the human operator as a simulated video clip, or an annotated picture.

Value Calculations. Due to the infinite state space, the optimal value function for a given reward parameter can no longer be calculated exactly, and so we employ $Sarsa(\lambda)$ (Sutton and Barto 1998) with linear function approximation to approximate value functions. This often makes the operator suboptimal with respect to the reward function chosen for a particular trial. We accordingly set α in Equation 3 to 40.0, which represents relatively high confidence in operator responses but allows robustness to possible inconsistencies in responses. Unlike our experiments in the Puddle World, the agent makes action-queries to learn the operator's driving preferences *before* it enters the world. As a result, there is no notion of current or start state, and so policy values, including those used in our policy loss performance metric, are computed as an expectation over possible start states.

Query Space. The infinite state space also makes it is impossible to exhaustively consider and rank every possible query with either EMG-AQS or AS. We need a means to approximately find each query evaluation algorithm's maximum. While various heuristics (perhaps domain dependent) might recommend themselves, to avoid biases in our experiments that might favor one approach over another, queries to consider were simply drawn at random from the query space. The larger the number of randomly-drawn queries, the greater the chances of finding a good query among them. In our experiments, we consider differently-sized sets of randomly-drawn queries to assess the impact of the set size.

Hybrid Method. Since EMG-AQS is computationally costly, we examine a hybrid AS-EMG-AQS approach, where AS is given a set of random candidate queries, and uses its less costly evaluations to pass a subset of its top-ranked options to EMG-AQS, which then selects from those.

Experiment 2. One way to scale action-query selection to real world problems would be to, like here, sample possible queries instead of exhaustively evaluating them. This motivates a comparison between EMG-AQS and AS on a computational efficiency level: even if EMG-AQS could select a better query than AS from the same set of candidate queries when allowed to evaluate all of them, AS might be able to evaluate a much larger set in the same amount of time and select a better one. Indeed, while the computation of both methods is linear in the number of queries considered, our implementation of AS evaluates a single query about 200 times faster than EMG-AQS (both using precomputed value functions), since EMG-AQS performs the time-consuming operation of formulating hypothetical policies for each of the answers to the query.

In this experiment, we measure the policy loss for each method asking one query, allotting each method the same amount of time to select from a large set of randomly-selected candidate queries. We also test our Hybrid method, which uses AS to rank 16 randomly-selected queries, and then uses the remaining allotted time for EMG-AQS to evaluate them in order until time expires and then return the best one found. Figure 2b shows that AS's superior speed allows it to find a better query more quickly than EMG-AQS, but does not benefit from additional time unlike EMG-AQS, allowing EMG-AQS to outperform AS once at least 3 seconds are available for computation. The Hybrid algorithm, on the other hand, gets the best of both worlds and meets or surpasses the performance of both methods at all points. Intuitively, this is due to AS's ability to quickly find queries whose answers are very uncertain, allowing the slower but more effective EMG-AQS to determine which of those is most helpful in terms of value gain.

Experiment 3. Here we test how the methods perform over query sequences in the Driving Domain, when again

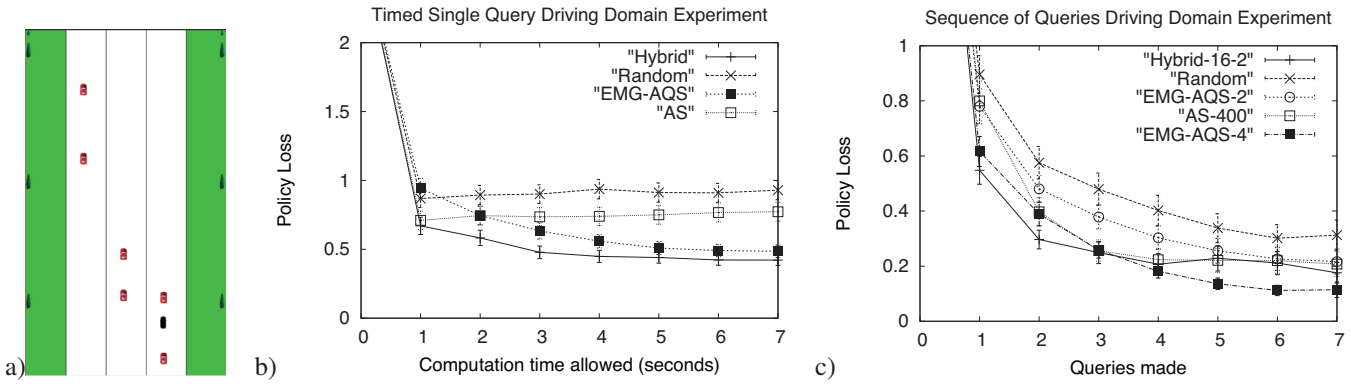

Figure 2: a) The Driving Domain. Average policy loss for various action-query methods according to b) computation time allotted and c) number of queries made. Error bars are 95% confidence intervals.

computation time is limited. We employ 3 methods that each take roughly 2 seconds to select a query per step: AS applied to a set of 400 random queries, EMG-AQS applied to a set of 2 random queries, and Hybrid where AS selects 2 queries from a set of 16 random queries and EMG-AQS selects the best of those (these roughly correspond to T=2 for each strategy on the previous graph). We also employ EMG-AQS applied to a set of 4 random queries, which consumes 4 seconds of computation per step, as well as Random.

Figure 2c shows that AS-400 outperforms EMG-AQS-2, but that the Hybrid method, which adds a small computational cost (about 4%) to EMG-AQS-2, outperforms both. EMG-AQS-4, consuming twice the computation time, outperforms the rest (except when beaten by Hybrid for the first two queries). We also ran the methods over larger query sets, and found that EMG-AQS and Hybrid continued to improve as they were allowed to sort through more random queries, while AS did not (not shown, due to lack of space). These results build on the trends we observed for a single query: EMG-AQS only outperforms AS when a certain level of computation is available, but Hybrid's usage of AS as a query filter can boost EMG-AQS's performance when computation is limited to get the best of both worlds.

6 Conclusion and Future Work

In this paper, we introduced a new action-query selection method, called Expected Myopic Gain-based Action-Query Selection (EMG-AQS), by adapting Expected Myopic Gain from its previous use in reward and transition queries via Bayesian Inverse Reinforcement Learning. We then presented an empirical comparison between EMG-AQS and an existing action-query selection method, Active Sampling (AS). Although EMG-AQS is Bayes-optimal for a single query, we tested the effects of two conditions under which EMG-AQS is not Bayes-optimal: performance over a sequence of queries, and performance given a fixed amount of time allotted to select queries. Under the former condition, we found that in most cases EMG-AQS outperformed AS over a sequence of queries due to its superior ability to exploit domain properties. Under the latter condition, we found that when little computation time was available, AS outperformed EMG-AQS, but once enough time was available, EMG-AQS outperformed AS. In addition, we devised the Hybrid algorithm, which performed better than either method alone in settings with limited computation time.

In future work we plan to apply action-query selection to more realistic problems that include a human operator. Sampling from the query space and applying our Hybrid method within computational limits could be a way to scale to more complex problems, along with leveraging work in the literature for efficiently calculating value functions.

Acknowledgments. Our thanks to Mike Maxim, Dave Karmol, and the anonymous reviewers. This research was supported in part by the Ground Robotics Reliability Center (GRRC) at the University of Michigan, with funding from government contract DoD-DoA W56H2V-04-2-0001 through the US Army Tank Automotive Research, Development, and Engineering Center. UNCLASSIFIED: Dist. A. Approved for public release.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Chajewska, U.; Koller, D.; and Parr, R. 2000. Making rational decisions using adaptive utility elicitation. In *AAAI*, 363–369.

Chernova, S., and Veloso, M. 2009. Interactive policy learning through confidence-based autonomy. *J. Artif. Int. Res.* 34(1):1–25.

Cohn, D.; Atlas, L.; and Ladner, R. 1994. Improving generalization with active learning. *Mach. Learn.* 15(2):201–221.

Cohn, R.; Maxim, M.; Durfee, E.; and Singh, S. 2010. Selecting operator queries using expected myopic gain. *IAT* 2:40–47.

Duff, M. O. 2003. Design for an optimal probe. In Fawcett, T., and Mishra, N., eds., *ICML*, 131–138.

Lopes, M.; Melo, F.; and Montesano, L. 2009. Active learning for reward estimation in inverse reinforcement learning. In *ECML PKDD*, 31–46.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *IJCAI*, 2586–2591.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Viappiani, P., and Boutilier, C. 2010. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *NIPS*, 2352–2360.