1 Abstract

Reinforcement learning problems are typically posed as requiring an autonomous agent to behave near optimally in some domain where often the dynamics of the environment are unknown a priori to the agent. To accomplish near optimal behavior in such a context, the agent is faced with discovering the properties of its environment on its own while behaving reasonably with respect to its current knowledge. Many interesting domains have available a human operator who is capable of providing the agent with expert knowledge directly as an alternative to forcing the agent to, possibly erroneously, discover it. For example, in military domains, agents operating in the field might have access to a human operator who is capable of supplying the agent advice. This paper will focus on the development of a theoretical framework for understanding and improving EMG, a method recently proposed to evaluate the utility of operator queries in an online setting.

2 Background and Related Work

As we are posing the operator query selection problem in a reinforcement learning setting, let us introduce the standard notation. We represent the agent's environment as a finite Markov Decision Process (MDP) defined by a tuple $M = < S, A, T, R>$, where S is a finite set of states, A is the set of actions the agent may take, T is the transition probability function, and R is the reward function. At each time step, the agent will observe its current state s and select an action given by $\pi(s)$, where π is the agent's current deterministic policy. Once an action a is selected, the agent will receive a reward given by R(s,a) and will transition to some state s' with probability T(s,a,s').

Upon this typical RL framework we add the ability for the agent to query an operator at cost C before it decides is next action to clarify unknown aspects of its environment. Note that this extra ability is not encoded directly into the set of actions that may be taken by the agent, but is instead assumed to be a separate part of the agent program which the agent may use to develop better policies when its knowledge of the environment is incomplete. Also note that the operator response is assumed to be correct: that is, the operator has full knowledge of the true MDP.

It is essential to consider what we mean by incomplete knowledge. One could imagine an agent with incomplete knowledge, for example, of the transition function, reward function, available actions, states, or any combination of these. Maxim et al [1] have looked at the case where the agent has knowledge of everything besides the true transition function, and Karmol et al [2] have begun preliminary work in the case where the agent has knowledge of everything besides the reward function. In this project we focus on the latter case, aiming to complete that preliminary work and augment their methods.

2.1 Preliminary Work Summary

The work done in [2] sets up a strong foundation for this project. Here we describe the methods proposed in [2] which are mostly in accordance with [1], but in the unknown reward function setting. Unless otherwise stated, all of the results in section 2 are due to [1] and [2]. This discussion will be kept necessarily brief.

2.1.1 Bayesian Learning

Let us assume the agent assumes a prior distribution over the MDP M, and has current knowledge of the world μ obtained by experience. Suppose the agent learns new information η. At this point we would like to obtain the posterior distribution of M given the agent's full information of the world. This can be done by utilizing Bayes rule:

$$P(M \mid \mu, \eta) \;\alpha\; P(\eta \mid M)\, P(M \mid \mu) \qquad (2.1)$$

And in order to maintain the posterior distribution over all possible worlds as we obtain experience, we can simply maintain the parameter $\theta^{r}_{s,a}$ for rewards r experienced for the state action pair s,a, and calculate the prior as

$$P(M|\mu) = \prod_s \prod_a P(R(s,a) \mid \theta^{r}_{s,a}) \qquad (2.2)$$

Where P $(R(s,a) \mid \theta^{r}_{s,a})$ is maintained by a set of Dirichlet distributions-- one for each state-action pair, where rewards experienced are counted and binned. The initialization of the Dirichlet parameters is crucial for controlling the agent's prior distribution of the world. That is, how the agent views the world in the absence of experience. One reasonable initialization of the parameters is setting all of them to 1, which produces a uniform distribution across all possible rewards for each state-action pair.

2.1.2 Policy Calculation

Now that we have described how the agent may keep track of its experience, let us discuss how the agent can compute policies given its current information. Since the agent has incomplete information of the world in an online setting, we must settle with defining an optimal policy as one that maximizes the expected value of the current state s:

$$\pi^{*} = \max_{\pi} E\,[V^{\pi}(s)] \qquad (2.3)$$

The optimal approach to maximize this quantity would be to evaluate policies through a weighted average across all possible MDPs, and choose the best possible policy through enumeration. However, such an approach is intractable as the space of possible policies and MDPs is vast. Instead we can approximate optimal policy calculation by just computing the mean policy with respect to the possible MDP distribution: that is, choose a policy that is optimal on the MDP formed by setting all unknown R(s,a) values to E(R(s,a)) given the agent's current knowledge. This approach is provably optimal for computing the optimal mean policy described in eq 2.3 in acyclic domains, and tends to be a good approximation otherwise [1].

2.1.3 Query Evaluation

Now that we have a way for the agent to keep track of the true MDP distribution through its experience and prior knowledge, and a way for the agent to compute reasonable policies given its current knowledge, we are ready to discuss how optimal querying can be done.

Consider a query q of the form "what is R(s,a)?" that will induce operator response o. How might we measure the Gain of such a query in order to choose the best possible query from the query space? One possible approach is to maximize the expected value of

$$Gain_{\theta}(q,o) = E_{\theta,o}[V^{\pi^{*}_{\theta,o}}(s) - V^{\pi^{*}_{\theta}}(s)] \qquad (2.4)$$

Where $V^{\pi^{*}\theta,o}(s)$ is the value of the optimal policy value function at current state s computed once o is known, $V^{\pi^{*}\theta}(s)$ is the value of the optimal policy value function at state s computed before operator response o is known. As shown in [2] it turns out that maximizing the expected value of equation 2.4 leads to the optimal query under the following myopic assumptions: the agent asks all its questions offline and the agent asks at most one question. This approach of selecting the best question is termed Expected Myopic Gain, or EMG.

To evaluate $E[\text{Gain}_\theta(q,O)]$, we can take the conventional approach for computing expected value:

$$E[\text{Gain}_\theta(q,O)] = \int_{rmin}^{rmax} \text{Gain}_\theta(q,o)\Pr(O = o \mid \theta)do \qquad (2.5)$$

Where $r_{\min}$ and $r_{\max}$ are the smallest and largest rewards possible in the MDP, which are known apriori to the agent. Note that this integral becomes a sum when the number of possible rewards in the MDP is finite and thus the possible rewards are discrete valued.

[1] and [2] make the argument that it is difficult to analytically evaluate this integral in the continuous reward case since it involves computing optimal policies across the continuous space of o. In order to approximate the integral, they propose leveraging Monte Carlo methods which, in evaluating each question, sample many MDPs from the agent's MDP distribution and compute the average gain throughout those samples. While this approach is computationally expensive, it has the advantage of arbitrarily adjustable accuracy through controlling the number of MDPs sampled at each query evaluation.

Note that queries could also be of the form "what reward do I get when entering this state?" which could correspond to the intuitive notion of asking "how good is it to enter this state?" which a human might be able to answer more naturally than state-action pair queries. Clearly the integral and all results concerning the gain remain the same, except the queries are of a different form. Since the response from the operator is still just a reward, these types of queries do not invalidate any results derived thus far.

In summary, by putting together the ability to keep track of current knowledge of the world through an MDP distribution, the ability to compute reasonable policies given such a distribution, and the ability to evaluate possible state-action pair reward queries, we have the foundation for our desired agent in the specific setting where the agent may ask the operator about the rewards of state-action pairs (or states). The only aspect of the agent program left to define is how it will decide whether the best query is worth asking in the first place, since it incurs cost C when querying the operator.

[1] address this by simply comparing the gain of the query q with C, and if $E(\text{Gain}(q,O)) > C$, then ask the question, otherwise do not. This cost model is clearly appropriate, but only where the operator cost is set by the designer to fit the specific situation—i.e, the designer should know what the expected gain values will look like and set the operator cost appropriately.

3 Policy Change Threshold Points

We may now begin discussing the work done in this project. As mentioned in the previous section, the integral for evaluating the gain of asking query q is difficult to compute in a general setting, so [1] and [2] use sampling in order to approximate it.

While sampling is a good approximation, it makes it difficult to tell the difference between states that have nearly identical gain and those that have identical gain, as shown in figure 3.1. Such information would be valuable for having more control over how the agent should behave when the gains of two or more states are the same, such as allowing for the possibility of

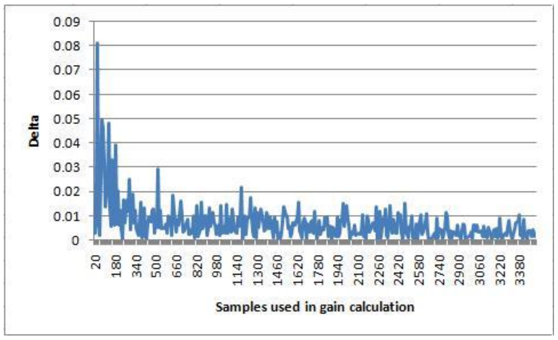


Figure 3.1: Two states in the simple tree domain that have equal gain do not yield equal gain when using the sampling method because of sampling error. Delta is the absolute difference between the sampled gains of each state. The delta decreases on average as the number of samples increases, but the graph indicates that a very large number of samples would be necessary to converge significantly closer to 0.

intelligent tiebreaking (if nothing else, allow the agent designer to force deterministic behavior in tie cases).

In addition, large amounts of sampling slows down the gain calculation tremendously, as a new policy must be computed for each sample and query. This computation is clearly restrictive when ten seconds or even more per gain calculation are required to obtain acceptable accuracy even in a small domain, as the agent would no longer be appropriate in an online setting. Also, when less sampling is done in order to expedite the gain calculation, the agent runs the risk of choosing a suboptimal query because of decreased accuracy.

These are factors that motivate a different approach to computing gain. It turns out that if we restrict the true MDP in a few ways, we can evaluate the integral with full accuracy and with much less effort than the sampling method requires.

In the following discussion we make the restrictive assumptions that A) operator queries are of the form "what reward will I receive upon entering state s'?, B) operator responses give an exact value for the query as opposed to a distribution, C) the MDP is acyclic, and D) the agent knows the minimum and maximum rewards possible in the MDP. For simplicity, we make the assumptions that E) each state has transitions to two or less other states, and F) each state has two or less actions available to be taken.

Assumptions E and F can be relaxed through fairly straightforward generalizations of the following ideas, but these generalizations are omitted from this paper because they do not play an important role in the discussion. On the other hand, it is not clear whether a generalization is possible that allows one or more of Assumptions A, B, C, or D to be relaxed. This is a topic for future work.

3.1 Introducing Threshold Points

Let us examine the relationship between operator response o to query of state s_q, and gain(s_q,o). Before considering possible queries, the agent has computed policy π_0 with respect to E[M|μ]. Operator response o induces a set of policies P^o which are optimal with respect to E[s(M)|μ, o] where s(M) is the subset of states of M that includes the current state s and all states reachable from s (it does not concern us whether any of the other states are optimal because the actions taken at those states do not affect the value of the current state, which is all that is needed in the gain calculation). Clearly, where $o = r_0 = E[R(s_q)]$, $\pi_0 \in P^o$ and gain(s_q, o) = 0. However, as o is increased from r_0, a point t_{+1} may exist where π_0 is no longer a member of P^o. Thus, for $o > t_{+1}$, a new policy π_{+1} must be computed such that $\pi_{+1} \in P^o$ in order to calculate the gain for $o > t_{+1}$.

We call t_{+1} a *policy change threshold point* for π_0, or for short, just a *threshold point*. Continuing across increasing *o*, there may be a threshold point t_{+2} where π_{+1} is no longer a member of P^o and a new policy π_{+2} must be computed. The process is continued until o=r_{max}, where the gain calculation becomes no longer relevant. In total, across increasing *o* from r_0 to r_{max}, some integer n_+ (n_+ will be shown to be finite in section 3.4) threshold points t_{+1}, t_{+2},…, t_{n+} are encountered. Clearly, the same argument may be applied to decreasing *o* from r_0 to r_{min} where n_- threshold points t_{-1}, t_{-2},…, t_{n-} are encountered. We list all threshold points for this particular policy changing sequence at once in increasing order as $t_1, t_2, \ldots, t_T$.

There are three key observations to make regarding the threshold points, which will be addressed in the subsequent subsections.

3.2 Threshold points can be analytically calculated

Define a state to be an *ancestor state* s_a for query state s_q if s_q can be reached with nonzero probability from s_a. Assume that the current state s is an ancestor state for s_q (if it was not, the gain for asking about s_q would be zero).

Clearly, the varying value of *o* will cause optimal policies to become suboptimal only with respect to the action taken at one or more ancestor states of s_q, since the value of actions taken at non ancestor states of s_q will not be affected by changing values of *o*. We say that an ancestor state s_a *favors* s_q if the action taken at s_a maximizes the probability that the agent will eventually enter s_q, and say that a state *disfavors* s_q if the action taken at s_a minimizes that probability. Here we only consider the case that the agent has two actions available at each state, so then a state s_a will favor or disfavor s_q only depending on which of the two actions the policy dictates should be taken at s_a (if neither action favors or disfavors s_q, either action can be chosen to "favor" s_q and the other can be chosen to "disfavor" s_q without affecting the subsequent results).

Given this notion, when will ancestor states change their favoring of s_q? The answer lies in the Q function of the policy π that is calculated at point $o = t_k$. By abuse of notation, $t_0 = r_0$ even though r_0 is not necessarily a threshold point.

Suppose ancestor state s has two children (per Assumption D) and two actions *f,d* available (per Assumption E) and that it currently disfavors s_q under the current policy. Further suppose that, under the current policy, the probability that the agent will enter state s_q given that it takes action *f* in s_a is equal to p_{af} and that the same probability if the agent takes action *d* in s_a is equal to p_{ad} (these probabilities can be easily calculated in an acyclic MDP by summing over the probabilities of each path that takes the agent through s_q from s_a when following π).

Clearly, s_a disfavoring s_q implies that $Q^\pi(s_a, d) < Q^\pi(s_a, f)$ at the point $o = t_k$ where the policy was calculated, or else the optimal policy would take action *f* in s_a instead of *d*.

S_a must change its favoring of s_q for any *o* after

$Q^\pi(s_a, f) \mid o = t_{k+1}$ for some $t_{k+1} > t_k$. Or, equivalently, if $\Delta_a = t_{k+1} - t_k$, then after

$$Q^\pi(s_a, d) = Q^\pi(s_a, f) \mid o = t_k + \Delta_a. \qquad (3.1)$$

Thus, when this condition is met, t_{k+1} would be the next threshold point if we were only considering s_a, since a policy will become suboptimal with increasing *o* only with respect to its action taken at each ancestor state s_a. Hence, we can compute the next threshold point from t_k toward increasing *o* by calculating $t_k + \min_a(\Delta_a)$, since clearly the policy will become suboptimal at the first ancestor state s_p to satisfy equation 3.1 with increasing *o*:

$$t_{k+1} = t_k + \min_a(\Delta_a) \qquad (3.2)$$

Now we just need to specify how to solve for each Δ_a. Since we are dealing with an acyclic MDP, the Bellman equation dictates that $Q(s_a, f)$ is a sum of terms, some of which depend on Δ_a and can be combined into one term by factoring out Δ_a. This implies that (assuming discount = 1) with increasing *o*, or equivalently, increasing Δ_a, $Q(s_a, f)$ increases in direct proportion to $p_{af}\Delta_a$, and similarly, $Q(s_a, d)$ increases in direct proportion to $p_{ad}\Delta_a$. This fact implies that

$$Q^\pi(s_a, d) = Q^\pi(s_a, f)$$

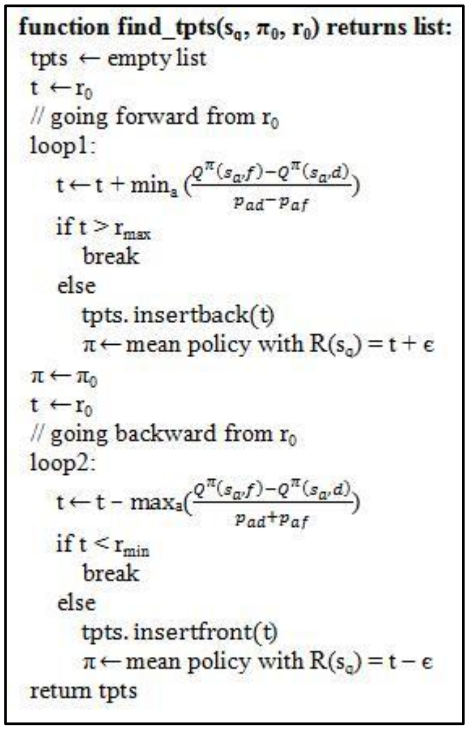
When

$$Q^\pi(s_a, d) + p_{ad}\Delta_a = Q^\pi(s_a, f) + p_{af}\Delta_a \quad (3.3)$$

This equation can be trivially solved for Δ_a since both Q values, p_{af}, and p_{ad} are all known. Thus, s_a must change its favoring of s_q when $o > t_k+\Delta_a$. Note that even though we assumed discount = 1, any discount can be taken into account by modifying p_{ad} and p_{af} to have the proper discount monomials multiplied by the probability of taking each possible path, as dictated by the Bellman equation. Also Note that equation 3.1 changes slightly when decreasing *o* rather than increasing *o*:

$$Q^\pi(s_a, d) - p_{ad}\Delta = Q^\pi(s_a, f) - p_{af}\Delta \quad (3.4)$$

Using these results, we give in figure 3.2 an algorithm for computing the threshold points for a query s_q, initial expected reward r_0, and policy π_0 computed on the mean MDP with $R(s_q) = r_0$.



```
function find_tpts($s_q$, $\pi_0$, $r_0$) returns list:
  tpts ← empty list
  t ← $r_0$
  // going forward from $r_0$
  loop1:
    t ← t + $\min_a(\frac{Q^\pi(s_a,f) - Q^\pi(s_a,d)}{p_{ad} - p_{af}})$
    if t > $r_{max}$
      break
    else
      tpts.insertback(t)
      $\pi$ ← mean policy with $R(s_q) = t + \epsilon$
  $\pi$ ← $\pi_0$
  t ← $r_0$
  // going backward from $r_0$
  loop2:
    t ← t - $\max_a(\frac{Q^\pi(s_a,f) - Q^\pi(s_a,d)}{p_{ad} + p_{af}})$
    if t < $r_{min}$
      break
    else
      tpts.insertfront(t)
      $\pi$ ← mean policy with $R(s_q) = t - \epsilon$
  return tpts
```

Figure 3.2: An algorithm to find the threshold points given a query state s_q, starting point r_0, and starting policy π_0. Note that ϵ is just a small real number-- it is needed in the increasing o case because π is still optimal at the threshold point but is suboptimal past the threshold point (similarly for the decreasing *o* case). Also note that ancestor nodes should only be considered in the increasing phase if they currently disfavor s_q and in the decreasing phase if they currently favor s_q.

3.3 Threshold Points are the same for all optimal policies

It is reasonable to ask the question "given that a policy π_0 is ϵ P^o at $o = r_0$ but no longer ϵ P^o at points $o > t_k$, is it possible that there exists one or more policies that are ϵ P^o at $o = r_0$ but also are ϵ P^o at $o = t_k$? In other words, is it possible that, once a policy change is made after point t_k and we continue increasing *o*, does the next threshold point location depend on the specific optimal policy that we choose when we change policies as we cross threshold points?

The answer is no, as the following theorem states:

Theorem 3.1: With assumptions A-E, the location of all of the threshold points is the same no matter what optimal policies are chosen at each individual threshold point.

Proof: First we will prove a lemma which will be helpful in proving the theorem.

Lemma 3.1: define $p^{\pi_i}_{s\to q}$ to equal the probability that state q will be entered from state s, the current state, given policy π_i. Then for all $\pi_i,\pi_j \in P^o$ for $o \in (t_k, t_{k+1})$ where t_k, t_{k+1} are two threshold points with no other threshold points in between, $p^{\pi_i}_{s\to q} = p^{\pi_j}_{s\to q}$.

Proof: In the case of an acyclic MDP, the value of state s amounts to a sum of terms only one of which involves $p^{\pi_i}_{s\to q}$. Since s is an ancestor state of s_q, we can apply equation 3.3 to get:

$$Q^{\pi}(s, d) + p_{sd}\Delta = Q^{\pi}(s, f) + p_{sf}\Delta \quad (3.5)$$

This equation implies that the Q values for each action in state s grow directly proportionally with the probability that the action eventually takes the agent to state s_q times Δ. Since $V^{\pi}(s) = \max(Q^{\pi}(s,d), Q^{\pi}(s,f))$ at $o=t_k+\epsilon$, $V^{\pi}(s)$ must grow directly proportionally with either $p_{sd}\Delta$ or $p_{sf}\Delta$, implying that $V^{\pi}(s)$ must grow directly proportionally with $p^{\pi}_{s\to q}\,\Delta$.

Because of this, for all $\pi_i,\pi_j \in P^o$ calculated at $o=t_k+\epsilon$, either $p^{\pi_i}_{s\to q} = p^{\pi_j}_{s\to q}$, or else at least one of π_i, π_j would not be optimal at both $o=t_k$ and $o=t_k+\epsilon$. But both π_i, π_j must be optimal at $o=t_k$ because if they were not, a threshold point t for at least one of π_i, π_j would have to exist such that $t_k < t < t_k+\epsilon$ and that being true would make that policy suboptimal on the entire interval $(t_k, t_k+\epsilon)$. Therefore we have a contradiction, because $\pi_i,\pi_j \in P^o$. Thus, for all $\pi_i,\pi_j \in P^o$, $p^{\pi_i}_{s\to q} = p^{\pi_j}_{s\to q}$.

We are now ready to prove Theorem 3.1. Lemma 3.1 implies that for all $\pi_i,\pi_j \in P^o$, $V^{\pi_i}(s)$ increases at the same rate as $V^{\pi_j}(s)$ for increasing o, since the change in V(s) for a policy π_i only depends on $p^{\pi_i}_{s\to q}$. This means that for $o >$ a threshold point for a particular policy $\pi_i \in P^o$, all policies in P^o must share that same threshold point because they are all guaranteed to be optimal before their own threshold points. Thus, all polices in P^o share the same threshold points.

Theorem 3.1 implies that given our previously stated assumptions A-E, the gain can be calculated in each inter-threshold point interval using any of the policies that are optimal between that interval.

3.4 The number of threshold points is finite and can be bounded

We have discussed the existence of threshold points and how they may be computed, but have not discussed how many of them actually exist or whether that number is even finite. Fortunately, the number of threshold points is indeed finite, and further, is surprisingly small.

Theorem 3.2: With assumptions A-E, if there are N ancestor states of s_q and T is the number of threshold points that occur in the interval r_{min} to r_{max}, then

$$T \leq N$$

Further, if the average number of threshold points per query state is T_{avg}, and the total number of states in the MDP is S, then

$$T_{avg} \leq \frac{S+1}{2}$$

Proof: To show this, we use the fact that we provided in figure 3.2 a method for locating threshold points. We show that it is possible to intelligently augment a policy when it becomes suboptimal past a threshold point to make it optimal instead of just finding some policy in P^o. We then show that we can provide an upper bound for how many threshold points could occur when following the algorithm. Leveraging theorem 3.1, the number of threshold points does not depend on the specific policy changes throughout the interval, so the bound derived for this specific algorithm is a bound for the number of threshold points.

With increasing *o*, ancestor states only have the potential of changing from disfavoring to favoring s_q, and vice-versa if we decrease *o*. This statement is true because, as discussed in the previous section, $Q(s_a, f)$ increases in direct proportion to $p_{af}\,\Delta$ and $Q(s_a, d)$ increases in direct proportion to $p_{ad}\,\Delta$ where $p_{af} > p_{ad}$.

With this in mind, the task of augmenting a policy at a threshold point t_k so that it is optimal past t_k turns out to be simple. Since for some ancestor state s_a equation 3.3 is satisfied, the policy at s_a is suboptimal for $o>t_k$. Hence by changing the policy from *d* to *f* , the policy must be optimal at s_a. This, however, may cause the policy to be suboptimal at some number of ancestor nodes because their Q values depend on the policy at s_a. The key observation is that these ancestor nodes will never change from favoring to disfavoring s_q—hence, the number of s_q-favoring ancestor states monotonically increases across increasing threshold points (a similar argument can be made to show that the number of s_q-disfavoring ancestor states is monotonically decreasing across decreasing threshold points).

Suppose that F ancestor states favor s_q and D ancestor states disfavor s_q for an optimal policy at $o = r_0$, where F+D=N. Then at most D states will change from disfavoring to favoring s_q from r_0 up to r_{max}, and at most F states will change from favoring to disfavoring s_q from r_0 down to r_{min}. This implies that the number of policy changes, or equivalently threshold points, $\leq F + D = N$ in the worst case where the policy changes at only one ancestor node per threshold point, hence $T \leq N$.

For the second part of the proof, we derive the bound by assuming that every state in the MDP is a potential query. With S states in an acyclic MDP, we can sum the number of ancestor states per state and divide by S to obtain T_{avg}.

The acyclic property dictates that for any two states s_i and s_j, one can be an ancestor of the other but they cannot both be ancestors of each other. This implies that if we sum the number of ancestor states per state in the worst case in decreasing order, and take the average, we obtain

$$\frac{(S-1)+(S-2)+\ldots+1}{S} \geq T_{avg}$$

Which is equivalent to

$$T_{avg} \leq \frac{S+1}{2}$$

3.5 Threshold points may be used to evaluate the gain integral

We presented in the previous sections an analytical solution to the locations of the threshold points and proved that the number of threshold points is bounded by the number of ancestor states for the query state. These results can be used to compute an exact solution to the gain integral in the case where P(O=o| θ) is uniform throughout the interval r_{min} to r_{min} – whether or not it is uniform depends on the prior distribution chosen by the designer of the agent.

In such a case, the expected gain integral can be evaluated easily by first locating the threshold points and then computing the value of s, the current state, for any optimal policy at that point. Since the value of s increases linearly between the threshold points as shown in equation 7 and P(O=o| θ) in the expected gain integral is constant throughout all values of o=R(s_q) with the uniform prior assumption, the expected gain between two threshold points t_k and t_{k+1}

$$\int_{t_k}^{t_{k+1}} \mathrm{Gain}_\theta(s_q, o) \Pr(O = o \mid \theta) do \qquad (3.5)$$

Defining for all k $V_{t_k} = E[V^*(s)|o = t_k]$, 3.5 is just, through a simple geometric argument (figure 3.3),

$$|t_k - t_{k+1}|(\frac{1}{2}(V_{t_k} + V_{t_{k+1}}) - V_0) \qquad (3.6)$$

Note that this evaluates to 0 for $t_k = t_{-1}$, $t_{+1} = t_{-2}$ because the agent's policy is optimal in the interval $[t_{-1}, t_{+1}]$.

And if there are T threshold points, the gain integral is reduced to

$$\sum_{k=0}^{T+1} \int_{t_k}^{t_{k+1}} \mathrm{Gain}_\theta(s_q, o) \Pr(O = o \mid \theta) do \qquad (3.7)$$

With $\mathrm{Gain}_\theta(s_q, o) \Pr(O = o \mid \theta)$ evaluated according to equation 3.6. Note that $t_0=r_{min}$ is added to the beginning of the list returned by the algorithm and $t_{T+1}=r_{max}$ is added to the end of the list.

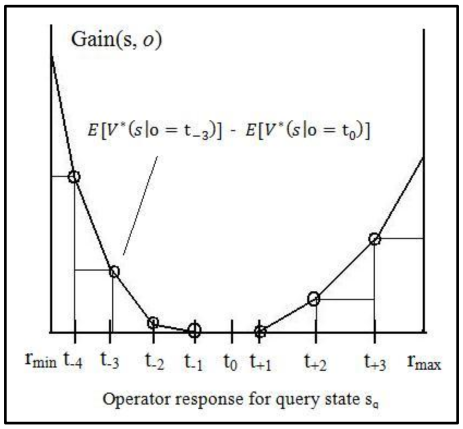


Figure 3.3: Hypothetical view of gain(s_q , O) for some s_q in some domain. The x axis represents possible operator responses for the reward for s_q, and the y axis represents the gain calculated as shown. Threshold points are marked Notice that since the gain across the interval is a piecewise linear function, the integral over the interval can be easily evaluated by computing the areas of the geometric shapes shown.

As discussed in section 2, the integral becomes a sum when the number of possible rewards in the MDP is a finite integer W and known to the agent in increasing order as $r_1, r_2, \ldots, r_W$. In this situation, equation 3.6 changes to

$$(\frac{D}{W})\frac{|E[V^*(s|o = t_k)] - E[V^*(s|o = t_{k+1})]|}{2} \qquad (3.8)$$

Where D is similar to the interval length $(t_k - t_{k+1})$ from equation 3.6 but adapted to

the discrete reward domain—if $t_k = r_i$ and $t_{k+1} = r_j$, then D = |j-i|.

4 Comparing threshold and sampling methods for computing expected gain

Here we show the result of applying the techniques developed in the previous section to an acyclic domain. The domain is the simple tree (figure 4.1), used in both [1] and [2] to test simple functionality and intuitive behaviors for EMG.

Running EMG as described in [2] using the sampling method on the simple tree, with start state 0, p=.9, and 500 samples for computing each gain value, we obtain the gain values for each state shown in figure 4.2a. Running EMG with the same values but using eq. 3.8 along with the algorithm given in figure 3.2, we obtain the result shown in figure 4.2b. Note that since [2] assume the number of rewards is finite and each possible reward is known to the agent, 3.8 is used to evaluate the gain accumulated between threshold points and the discrete reward caveat mentioned in the caption of figure 3.2 is implemented.

Thus we have empirically demonstrated that, when Assumptions A-E are met, threshold point gain evaluation provides exact gain values as opposed to approximated ones as produced by the sampling method.

The threshold point gain evaluation also solved the problem much faster than the sampling method (.052 seconds vs 50.88 seconds on a Pentium 4 3.40Ghz processor to repeat the computation 100 times). This is due to the fact that in this particular problem, only one threshold point exists for each query state, thus only two policies per state were calculated to evaluate the gain. In contrast, in the sampling method 500 policies per query state were calculated.

In general, when the average number of threshold points per state T_{avg} is less than the number of samples used in the sampling method, threshold point gain calculation should be expected to run faster than the sampling method, since the core of the computation in both methods is calculating the necessary mean policies. Of course, when MDPs become vast sampling has the potential to become faster than using threshold points. According to the second bound given by theorem 3.2, it becomes possible for sampling to become faster than threshold point calculation for samples=500

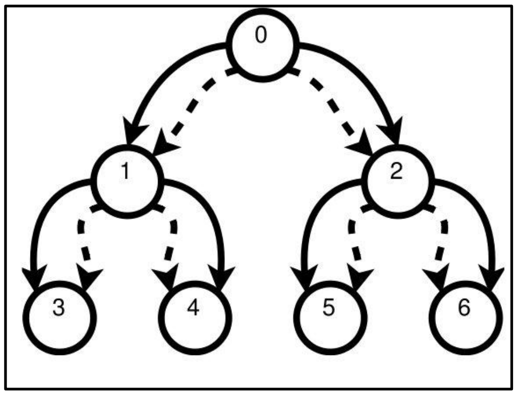
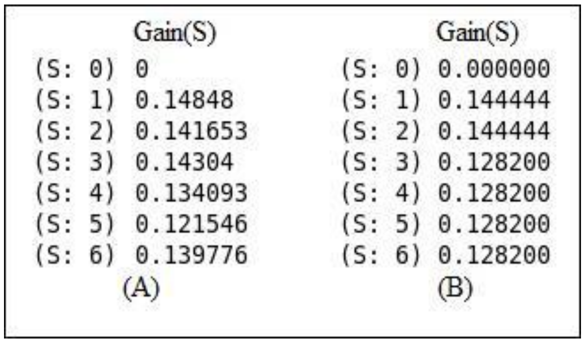


Figure 4.1: The simple tree domain. Two actions are available to the agent at each state, represented by the dotted and solid lines. Taking the solid action transitions to the state's left child with probability p and to the state's right child with probability 1-p. Taking the dotted action transitions to the state's left child with probability 1-p, and transitions to the state's right child with probability p. States 3, 4, 5, and 6 are goal states, each of which may emit a reward upon entrance. Only rewards 0, 1, and 2 are possible and this information is known to the agent. Operator response for a query state consists of the exact reward that will be received upon entrance to that state.



```
          Gain(S)                      Gain(S)
(S: 0)  0                    (S: 0)  0.000000
(S: 1)  0.14848              (S: 1)  0.144444
(S: 2)  0.141653             (S: 2)  0.144444
(S: 3)  0.14304              (S: 3)  0.128200
(S: 4)  0.134093             (S: 4)  0.128200
(S: 5)  0.121546             (S: 5)  0.128200
(S: 6)  0.139776             (S: 6)  0.128200
         (A)                          (B)
```

Figure 4.2: Expected gain values for each query state computed on the simple tree domain where p=.9, start state = 0, and no prior knowledge of the exact values of any of the state rewards. Here we show the values computed by two different methods: (A) averaging gain across sampled operator responses and (B) exact evaluation using threshold points to find policy changes. Notice that even though the exact values of the gains show that gain(1)==gain(2)>gain(3)==gain(4)==gain(5)==gain(6), the sampling method is not only off on the equals relationships but actually has gain(3) > gain(2) due to sampling error.

when $\frac{S+1}{2} \geq 500$. That is, when the number of states in the MDP ≥ 999.

5 Conclusion

At the outset of this project, our goal was to extensively test the methods proposed in [2] in key domains. However, during the implementation and testing period of a subset of those methods, it became necessary to intuitively understand how the gain calculation chose queries in order to evaluate whether the methods were functioning properly. It was at this point that the idea of threshold points came up, and the idea grew more interesting when it became apparent that given the locations of the threshold points, the expected gain integral could be evaluated quickly and exactly. Thus, in the time since the progress report we decided to move towards rigorously understanding the threshold points and providing proofs for what seemed like intuitive notions about them.

In this project we developed the concept of policy change threshold points and proved various desirable properties about them, which may be used as a theoretical framework for understanding the expected gain calculation proposed in [1] and [2]. Restrictive assumptions were made in order to obtain these results, mainly the assumption that the MDP is acyclic. However, in domains that meet the assumptions, EMG's gain calculation was both made exactly accurate and expedited by using threshold points compared to the sampling method, and this fact was demonstrated empirically on a simple example.

It is important to state that the fact that sampling works when all assumptions made here are broken and works faster than the threshold point method on vast domains. These advantages make it the better all around choice even though it introduces inaccuracies in the gain calculation. However, future developments could potentially extend threshold point gain calculation to domains where the assumptions are broken, which would clearly make the threshold point method the better choice in domains that are not tremendously large.

As a last note, even though threshold points were discussed in the context of the gain calculation, their involvement is only due to the fact that the gain calculation centers around the need for optimal policy changes with a changing reward function. Thus, in other problems where the same relationship is crucial, the theory developed here might prove useful.

6 Future Work

As previously stated, extending the threshold point theory to hold in the absence of the assumptions stated here is an important task in making them applicable to interesting domains.

In addition, extensively testing the threshold point gain calculation across multiple domains of various sizes would be useful in understanding

Finally, even though the threshold point gain calculation does not apply to domains that violate the assumptions, it may be possible to apply a modified version of it as a heuristic for only looking at promising queries and skipping the gain calculation for others.

7 References

[1] Maxim, Durfee, and Singh. Selecting Operator Queries Using Expected Myopic

Gain. *Submitted to AAMAS conference,* 2009.
[2] Karmol, Durfee, and Singh. Extending Selective Model Acquisition. *Unpublished writeup*, 2009.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998

[4] Deepak Ramachandran. Bayesian inverse reinforcement learning. In *in 20th Int. Join Conf. Artificial Intelligence*, 2007.

[5] Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *In Proceedings of the fifteenth Conference on Uncertainty in Artificial Intelligence,* pages 150-159, 1999.