Light at the End of the Tunnel: A Monte Carlo Approach to Computing Value of Information

Ece Kamar
Microsoft Research
Redmond, WA 98052
eckamar@microsoft.com

Eric Horvitz
Microsoft Research
Redmond, WA 98052
horvitz@microsoft.com

ABSTRACT

Calculating the expected value of information (VOI) for sequences of observations under uncertainty is intractable, as branching trees of potential outcomes of sets of observations must be considered in the general case [11]. We address the combinatorial challenge of computing ideal observational policies in situations where long sequences of weak evidential updates may have to be considered. We introduce and validate the use of Monte Carlo procedures for computing VOI with such long evidential sequences. We evaluate the procedure on a synthetic dataset and on a challenging citizen-science problem and demonstrate how it can effectively cut through the intractability of the combinatorial space.

Categories and Subject Descriptors

I.2 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, generation

General Terms

Algorithms, Experiments

Keywords

Monte Carlo planning, POMDP, crowdsourcing systems

1. INTRODUCTION

An important pillar of intelligent behavior is the ability to balance the value and costs of collecting information in advance of taking action [11]. Calculating the expected value of information (VOI) for sequences of observations under uncertainty is intractable [9]. The task involves computing expectations over an exponentially growing tree of future evidence-gathering actions and outcomes. Researchers have relied largely on the use of myopic approaches for calculating VOI [5, 23]. For example, calculations of the value of a single next test has been used to guide decision making in medical diagnosis and in machine troubleshooting systems [6, 10].

Such myopic approximations of VOI are poorly characterized as they rely on the assumption that only a single piece of evidence will be observed in advance of action, but are nevertheless applied in sequential information-gathering settings. Some researchers have pointed to proofs of well-defined error bounds for greedy computation of VOI hinging on confirming *submodular* structure in the relevance of information [16]. For submodular problems, acquiring additional information has diminishing marginal value.

Many real-world problems are not submodular. Thus, proofs of bounds on error in VOI relying on the submodularity property are often irrelevant. Indeed, information revelation may even have supermodular influences on belief updates, where dependencies among disparate pieces of evidence come together in synergistic ways. Obtaining a sequence of observations may significantly change a system's belief and consequently the best action to take, but obtaining a single piece of observation may have little or no effect.

Beyond greedy VOI computations, researchers have proposed non-myopic methods for computing VOI, relying on specific independence assumptions among observations and state variables [9, 3, 17]. The assumptions are not satisfied in many real-world evidence gathering tasks, which makes existing non-myopic approaches either invalid or intractable.

In summary, real-world challenges bring to light problems that cannot be solved with available methods. We pursue a solution to computing VOI for *long evidential sequence* (LES) tasks. For an LES task, observations individually provide only weak evidence about the state of the world, yet sets of observations may provide significant value. Computing VOI for decisions about stopping versus collecting additional information is difficult in these situations because the value of an agent's ultimate domain action hinges on the fusion of long sequences of observations. Greedy procedures fail as they do not identify significant value in single pieces of evidence and halt evidence gathering prematurely. And attempts to execute even small amounts of lookahead face a rapidly rising combinatorial wall.

We present MC-VOI, a Monte Carlo algorithm for computing VOI in LES tasks. The procedure considers a special partially observable Markov decision process (POMDP) where we decouple domain actions from observation gathering. The algorithm performs large lookaheads using a sampling technique that can explore multiple observation and action outcome sequences with a single sample, reducing the number of samples required to accurately estimate VOI.

We evaluate the performance of MC-VOI on both a synthetic dataset and a real-world LES problem, in which an automated system collects observations from people in a large-scale citizen science effort. We find that MC-VOI effectively cuts through the intractability of the combinatorial space and outperforms existing Monte Carlo planning algorithms.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

2. RELATED WORK

Our research on generating observational plans for tasks that require the consideration of long evidential sequences relates to multiple lines of work on solving large MDPs, on computing VOI, on solving optimal stopping problems, and on multi-armed bandit problems. Heuristic search has been a popular solution technique for large MDPs when good heuristic functions are available for a domain (i.a., [2]). For solving large MDPs in a general way, Monte Carlo planning has been proposed [14, 15, 22]. However, existing algorithms cannot efficiently explore long horizons in LES tasks and they may fail to make effective decisions when constrained with runtime limitations.

In addition to the related work on computing VOI that we present in Section 1, researchers have sought to develop tractable VOI computations by developing inferential procedures that leverage the special structure of specific classes of problems. Hajishirzi, et al. explores the use of a POMDP on the challenge of detecting changes in the world when the search space corresponding the problem grows linearly with time in contrast to the exponential search space of LES tasks [7]. The timed-decision problem introduced by Reches et. al. studies when to terminate evidence collection, but in tasks in which states are static and the observations about domain actions are independent [20].

Optimal stopping problems are concerned with optimizing the timing of domain actions to maximize performance. Threshold-based algorithms are proposed for a special class of optimal stopping problems for optimizing the exploration of candidates to identify the best candidate [19]. These tractable algorithms are not applicable to LES tasks, as combinations of evidence influence rewards of LES tasks arbitrarily. Dynamic programming approaches for general optimal stopping problems are intractable for long horizons [8]. Methods for solving multi-armed bandit problems are not applicable in LES tasks, as an action taken on an LES task may change the expected utility of taking any action in the future [18].

3. LONG EVIDENTIAL SEQUENCE TASKS

A long evidential sequence task centers on identifying the best domain action to take under uncertainty about the world. An LES task terminates when a domain action is taken, and is assigned a reward based on the action and the state of the world. Agents may delay their domain actions and invest time and effort to collect information that may enhance the expected values of their actions. Thus, agents need to balance the expected utility of collecting additional evidence with the overall cost of the observations.

LES tasks arise in a wide spectrum of arenas from medical diagnosis to supporting people decision making in daily life. As an example, it may be valuable for an agent to wait for new sets of weak evidence streaming in about traffic and to trade the value of the newly arriving information for the cost of delayed assistance to a driver. We shall introduce a challenging example of LES tasks within a citizen science application that involves the acquisition of long sequences of votes from workers about classifications of heavenly bodies.

To solve LES tasks, an agent needs to reason about multiple dimensions of uncertainty. The state of the world is not fully observable, the state can stochastically change, and the evidence that might be collected with future observations is uncertain. Formally, a LES planning task can be modeled as a finite-horizon POMDP [12], which is represented as a tuple $< S, \mathcal{A}, T, R, \Omega, O, l >$. S is a finite set of states of the world. $S_e \subset S$ is a set of terminal states. A is a finite set of actions. $A = D \cup \{c\}$, where $D = \{d_1, ..., d_n\}$ is the set of domain actions, and c is the evidence collection action. $T : S \times A \times S \rightarrow [0, 1]$ is the transition function. Since LES tasks terminate once a domain action is taken, for any state $s \in S$ and any domain action $d \in D$, $T(s, d, s')$, the probability of transitioning to any non-terminal state s' from s by taking action d, is 0. $R : S \times A \rightarrow \mathbb{R}$ is the reward function. For any $d \in D$, $R(s, d)$, the reward for taking action d in state s, depends on the quality of domain action d in state s. $R(s, c)$ may correspond to a negative value that represents the cost for collecting additional evidence. Ω is a finite set of observations available in the domain, and $O : S \times A \times \Omega \rightarrow [0, 1]$ is the observation function. $O(s, a, o)$ represents the probability of observing o after taking action a in state s. l is the horizon of decision making.

An agent solving an LES task typically cannot directly observe the state of the world. The agent maintains a belief state, which is a probability distribution over world state S at each step. The agent has access to a belief update function which updates the belief state based on the observation received, the action taken, and the previous belief state [12]. $\tau(b, a, b')$ represents the probability of transitioning to belief state b' after taking action a at belief state b.

LES tasks terminate after a domain action is taken. Thus, the decision of which domain action to take at any belief state depends only on the immediate rewards of domain actions at the current belief state. This characteristic enables us to decouple decisions about whether to collect more evidence from decisions about the best domain action. We map the POMDP definition given above to a specialized belief MDP representation that we refer to as an LES-MDP, though the same mapping can be achieved with a POMDP. An LES-MDP decouples observation and domain actions, and is represented by a tuple $< B, \mathcal{A}', \tau', r, l >$, where

- B is the set of belief states over S.
- $A' = \{c, \neg c\}$ is a set of actions, where $\neg c$ is the action for termination. When $\neg c$ action is taken, the agent identifies $d^*(b)$, the optimal domain action at belief state b, and executes that domain action as follows:

$$d^*(b) = \arg\max_{d \in D} \sum_{s \in S} b(s) R(s, d)$$

- τ', the belief state transition function, is:

$$\tau'(b, c, b') = \tau(b, c, b')$$
$$\tau'(b, \neg c, b') = \tau(b, d^*(b), b')$$

- r, the reward function on belief states, is:

$$r(b, c) = \sum\nolimits_{s \in S} b(s)\; R(s, c)$$
$$r(b, \neg c) = \sum\nolimits_{s \in S} b(s)\; R(s, d^*(b))$$

A policy π specifies whether it is beneficial for the system to take a domain action or to collect more evidence at any belief state. An optimal policy π^* with value function V^{π^*} satisfies the Bellman equation:

$$V^{\pi^*}(b) \quad = \quad \max_{a \in \mathcal{A}'} \big(r(b, a) + \sum_{b'} \tau'(b, a, b')\, V^{\pi^*}(b') \big)$$

For an LES task, the VOI computed for a belief state represents the expected utility for gathering additional evidence rather than taking an immediate domain action by considering the immediate cost for collecting evidence as follows:

$$\begin{aligned} VOI(b)) &= V^{c}(b) - V^{\neg c}(b) \\ &= r(b,c) + \sum_{b'} \tau'(b,c,b')\, V^{\pi^*}(b') - r(b,\neg c) \end{aligned}$$

If VOI at a belief state is positive, it is beneficial to collect evidence. As demonstrated by the Bellman equation, the number of belief states to be searched for an exact VOI computation grows exponentially in the horizon, which makes exact solution approaches intractable for long horizons.

4. CROWDSOURCING TESTBED

We now turn to a class of crowdsourcing that we refer to as *consensus tasks* to demonstrate the need to develop tractable procedures for computing information value in LES tasks. An owner of a consensus task seeks to identify a correct answer to a prediction or classification problem, and turns to populations of workers who can provide evidence about the answer. Hiring a worker is associated with a cost. An automated system designed to assist with the solution of consensus tasks needs to make a decision at each step: hire a worker or terminate the task with a prediction about the correct answer based on reports collected so far. The goal is to optimize the expected utility for the owner over one or many tasks given utilities and costs for making correct or incorrect predictions, and costs for hiring workers.

We perform studies on a citizen science a project named Galaxy Zoo, one of the largest citizen science efforts to date. Galaxy Zoo was designed to engage members of the public to contribute to consensus tasks aimed at identifying the correct classifications of millions of galaxies [1]. In each session, a worker is asked to classify galaxies into one of six possible galaxy classes (e.g., elliptical galaxy, spiral galaxy, etc.). The system was made available for 20 months and during this time more than 100,000 participants provided 34 million votes (observations) on 886 thousand galaxies. We employed this dataset in our studies.

In previous work, Kamar et. al. defined Galaxy Zoo tasks as a consensus task and formalized solving these tasks as sequential decision making [13]. Following this definition, we use L to denote the set of galaxy classes. For a given galaxy, the system collects a vote from worker i, $v_i \in L$, about the correct classification of the task. According to the consensus definition provided by the designers of the Galaxy Zoo system, after collecting as many votes as possible for a galaxy, the system identifies the correct answer (i.e., correct classification of a galaxy) as the answer that is agreed upon by at least 80% of the workers. If such a consensus is not reached after hiring a large number of workers, the correct answer is called *undecidable*. The set of decisions that can be taken by the system about the classification of a galaxy, D, is defined to be $D = L \cup \{undecidable\}$.

4.1 Predictive Models

A formal representation of a consensus task includes models for predicting (1) the state of the world, (2) future observations (worker votes), and (3) how the state of the world changes. We build these predictive models from data via supervised learning. The predictive models take as input a set of features f which characterizes a task (e.g., visual features of a galaxy), and a history of observations $h_t =< v_1, ..., v_t >$. We perform Bayesian structure learning to build probabilistic models from a training set [4] and evaluate the models on a test set.

The answer model M_{d^*} predicts the state of the world without knowing the number of votes available for each task. $M_{d^*}(d, f, h_t)$ is the probability of the correct answer being d, given features and the history of worker votes. An evaluation of the answer model on a separate test set shows that it has 88% accuracy in the absence of worker votes. Its accuracy increases to 98% when a large number of votes are available. M_v refers to the vote model, which predicts future observations. $M_v(v_{t+1}, f, h_t)$ predicts the probability of the next vote being v_{t+1}, given task features and the history of votes. We found that the model predicts the next vote with 57% accuracy when no votes are available and achieves a 64% accuracy after 15 votes or more are collected.

The number of worker votes for each galaxy varies greatly in the Galaxy Zoo dataset. When we run experiments on the dataset, a task may terminate stochastically when no additional votes are available for a galaxy. We estimate a probabilistic termination model from the training set. For simplicity of representation, we exclude this model from the LES-MDP formalization given below. However, our experimental evaluations are performed on an updated LES-MDP model with transition and reward functions extended with the stochastic termination model.

4.2 Galaxy Zoo as LES-MDP

We model Galaxy Zoo tasks as an LES-MDP, represented by the tuple $< B, A', \tau', r, l >$. The horizon of a task l is determined by the ratio of the maximum utility gained from a correct prediction to the cost of a worker. $b_t =< p_t, f, h_t >$ is the belief state at time t, where p_t is the system's belief about the correct answer as a probability distribution over possible answers of a task. The set of actions is $\mathcal{A}' = \{c, \neg c\}$. Once the system decides to terminate, the system's decision about the correct answer is computed as

$$d^*(b_t) = \arg\max_{d \in D} \sum_{d^c \in D} p_t(d^c) U(d, d^c)$$

where $U(d, d^c)$ is the utility for the system predicting the correct answer as d when the correct answer is d^c.

The belief state transition function τ' models the system's uncertainty about worker votes and the stochastic transitions about the world. When the system either reaches the horizon or takes action $\neg c$, it transitions deterministically to a terminal state. Otherwise, the probability of system's belief transitioning from b_t to b_{t+1} is

$$\tau'(b_t, c, b_{t+1}) = \begin{cases} M_v(v_{t+1}, f, h_t) \\ \quad \text{if } h_{t+1} = h_t \cup \{v_{t+1}\} \text{ and } p_{t+1} = p' \\ 0 \quad \text{otherwise} \end{cases}$$

where $b_t =< p_t, f, h_t >$, $b_{t+1} =< p_{t+1}, f, h_{t+1} >$, and for each $d \in D$, $p'(d) = M_{d^*}(d, f, h_{t+1})$.

The reward for action c is $(-\gamma_w)$, where γ_w is the cost for hiring a worker. The reward for taking action $\neg c$ in belief state b_t is:

$$r(b_t, \neg c) \quad = \quad \sum_{d^c \in D} p_t(d^c)\, U(d^*(b_t), d^c)$$

The Galaxy Zoo domain frames important challenges in solving real-world LES tasks. A Galaxy Zoo task has 44 votes on average, and may have up to 93 votes. Thus, the horizon of these tasks can be large, which makes exact solution approaches intractable. The answer model learned from Galaxy Zoo data is noisy when only a few votes are available, and becomes more accurate as more votes are collected. Consequently, the reward estimation of early belief states may be erroneous. Such errors in the reward estimation may degrade the performance of traditional solution algorithms that evaluate the goodness of taking an action on a belief state, based on the value of the reward function. An early belief state that is overly confident about a classification may mislead these algorithms to terminate prematurely.

5. MONTE CARLO PLANNING FOR VOI

Monte Carlo planning is an approach to solving large planning problems [14, 15]. General Monte Carlo planning algorithms hit a combinatorial challenge in exploring the long horizons typically associated with LES tasks; the number samples for exploring state–action outcomes grows exponentially in the horizon. These algorithms initially favor parts of the search space closer to the root, and thus require large number of samples to explore long horizons. Finally, the way belief state–action outcomes are evaluated by these algorithms are susceptible to noise in belief estimation of LES tasks when few evidences have been observed.

We present MC-VOI, a Monte Carlo planning algorithm that uses the special structure of LES tasks for addressing the drawbacks of existing algorithms for solving these tasks. MC-VOI explores the search space with sampling. Each sample corresponds to an execution path, which is a sequence of belief state outcomes that the system would encounter when it takes action c from an initial belief state to a terminal belief state. For each execution path, the algorithm evaluates the rewards associated with taking actions c and $\neg c$ for any belief state encountered on the path. The evaluation uses a state sampled at the terminal belief state when all available observations are collected. It builds a search tree based on execution paths sampled, and optimizes actions for each belief state on the tree. MC-VOI differs from existing Monte Carlo planning algorithms in a number of ways: Because a LES task terminates after taking a domain action, MC-VOI can evaluate the utility of any sequence of c and $\neg c$ action outcomes with a single sample. Doing so requires fewer number of samples to explore long horizons of LES tasks. Each sample of the algorithm traverses belief states from the root to the horizon, thus the algorithm can simultaneously explore belief states close to the root, as well as ones close to the horizon. The algorithm evaluates all belief state–action outcomes based on a state sampled at the horizon when all available evidence is observed, thus utility estimates are less susceptible to noise in earlier belief updates.

We first present the MC-VOI algorithm for LES tasks where the state of the world is static. Such LES challenges include Galaxy Zoo tasks in which the world state (i.e. the correct classification of a galaxy) does not change in time but the belief state changes over time as the system collects additional worker reports and becomes more confident about the correct answer. We shall later generalize the MC-VOI algorithm for LES tasks where the ground truth of state is changing during evidence gathering.

5.1 MC-VOI for Static Tasks

We present the MC-VOI algorithm for LES tasks with static state as Algorithm 1. For a given LES-MDP and an initial belief state b_0, the MC-VOI algorithm builds a partial search tree iteratively by calling the *SampleExecutionPath* function. As captured in the pseudocode, every call to *SampleExecutionPath* samples one execution path, which includes a sequence of belief states that would be visited when the system continuously takes action c until reaching a terminal belief state. The algorithm grows the search tree by adding a new node when the *SampleExecutionPath* function samples a belief state that is not encountered before. For each encountered belief state b_t, the algorithm keeps four values; $b_t.C$ as the expected immediate cost for taking action c, $b_t.V^c$ as the expected value for taking action c, $b_t.V^{\neg c}$ as the expected value for taking action $\neg c$, and $b_t.V$ as the expected value for taking the best of these actions. In addition, $b_t.N$ keeps count of the number of times b_t is encountered. All these values are initialized to 0.

The *SampleExecutionPath* function samples an execution path by starting from the initial belief state (b_0) and sampling future belief states as it continuously collects more observations until reaching a terminal belief state. For a given belief state b_t, the likelihood of sampling b_{t+1} as the next belief state is equal to $\tau'(b_t, c, b_{t+1})$. The *SampleTrueState* function is called at a terminal belief state to sample a state. At a terminal belief state b_l, the likelihood of *SampleTrueState* sampling any $s \in S$ is $b_l(s)$. This sampled state is used by the *Evaluate* function to evaluate the rewards for taking actions c and $\neg c$ at any belief state on the execution path. The algorithm samples the state at a terminal belief state when all available evidence is collected and when belief estimation is most accurate. By doing so, it can evaluate the rewards of all belief states on an execution path consistently and can reduce the negative effects of noisy belief estimates at earlier states.

The *Evaluate* function updates the statistics (i.e., values and counts) of belief states visited on an execution path from bottom to top based on the sampled state s. For each belief state, the function applies the Bellman equation (Section 3) on the partial search search tree to update its values: $b_t.V^{\neg c}$, the value for terminating, is updated based on $R(s, d^*(b_t))$, the immediate reward at state s for taking the best domain action according to belief b_t. $\Phi(b_t)$ represents the set of belief states that b_t can transition to in the partial search tree. $b_t.V^c$, the value for hiring, is computed based on the values of future states; it is the weighted average of the values of the belief states in $\Phi(b_t)$ minus $b_t.C$, the immediate cost for collecting more evidence. $b_t.V$ is computed as the maximum of $b_t.V^c$ and $b_t.V^{\neg c}$. After the algorithm samples many execution paths, each encountered belief state has an expected value for terminating and an expected value for collecting more evidence. The algorithm chooses the action to take at any belief state by calculating an expected value of information (VOI) as the difference of the expected values for terminating and for collecting more evidence.

We demonstrate the workings of the algorithm on a simple example given in Figure 1. Consider a system that seeks observations from experts on the correct answer of a task, where there are two possible answers ($D = \{1, 2\}$), two possible observations (votes) ($o_i \in \{1, 2\}$), and a horizon of 3. The reward for making the correct prediction is 1.0, and the cost of hiring an expert voter is 0.1. The initial belief state

```
CalculateVOI($b_0$:belief state, $l$:horizon)
begin
    repeat
        $SampleExecutionPath(b_0, l)$
    until $Timeout$
    $VOI \leftarrow b_0.V^c - b_0.V^{\neg c}$
    return $VOI$
end

SampleExecutionPath($b_t$:belief state, $l$:horizon)
begin
    if $\neg IsTerminal(b_t, l)$ then
        $b_{t+1} \leftarrow SampleNextBeliefState(b_t)$
        $s \leftarrow SampleExecutionPath(b_{t+1}, l)$
    else
        $s \leftarrow SampleTrueState(b_t)$
    end
    $Evaluate(b_t, s, l)$
    return $s$
end

Evaluate($b_t$:belief state , $s$:state, $l$:horizon)
begin
    $b_t.N^{\neg c} \leftarrow b_t.N^{\neg c} + 1$
    $b_t.V^{\neg c} \leftarrow \frac{b_t.V^{\neg c}(b_t.N^{\neg c} - 1) + R(s, d^*(b_t))}{b_t.N^{\neg c}}$
    if $\neg IsTerminal(b_t, l)$ then
        $b_t.N^c \leftarrow \sum_{b'_{t+1} \in \Phi(b_t)} b'_{t+1}.N$
        $b_t.C \leftarrow \frac{(b_t.C\ (b_t.N^c - 1) + R(s, c))}{b_t.N^c}$
        $b_t.V^c \leftarrow \frac{\sum_{b'_{t+1} \in \Phi(b_t)} (b'_{t+1}.V\ b'_{t+1}.N)}{b_t.N^c} - b_t.C$
    end
    if $b_t.V^{\neg c} \geq b_t.V^c$ or $b_t.N^c = 0$ then
        $< b_t.V, b_t.N > \leftarrow < b_t.V^{\neg c}, b_t.N^{\neg c} >$
    else
        $< b_t.V, b_t.N > \leftarrow < b_t.V^c, b_t.N^c >$
    end
end
```

Algorithm 1: MC-VOI algorithm.

is b_0^1. The belief state b_j^i is the i^{th} belief state at depth j. o_i is the observation obtained at time i. Figure 1(a) displays a recursive call of the *SampleExecutionPath* function. The flow of control is represented by the directions of the arrows. When the algorithm reaches the terminal belief state b_3^4, it samples the state by sampling a value for the correct answer of the task. The correct answer d^c is sampled as 2. Diamond shapes in 1 represent the calculation of immediate reward for terminating at any belief state. $d^*(b_3^4)$, the prediction of the correct answer at belief state b_3^4, agrees with the value of the correct answer d^c, this belief state is rewarded 1 for terminating. All earlier belief states predict the value of d^c incorrectly and receive a reward of 0. Figure 1(b) displays the partial search tree that is generated as a result of sampling 10 execution paths. N values represent the number of times leaves are sampled. The execution path given in bold represents the execution path given in part (a) of the figure. Let $b_0^1.V^{\neg c}$ be 0.8, $b_1^1.V$ and $b_1^2.V$ be 0.8 and 1 respectively. $b_0^1.V^c$ is computed as 0.91 by taking the weighted average of $b_1^1.V$ and $b_1^2.V$ and subtracting the cost of a worker. $EVOI$ for belief state b_0^1 is 0.11 based on this partial tree.

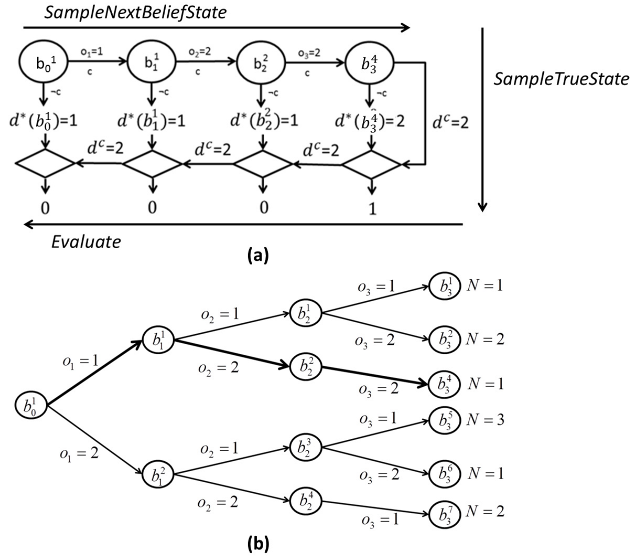

Figure 1: Illustration of MC-VOI algorithm.

5.2 MC-VOI for Dynamic Tasks

Algorithm 2 expands MC-VOI for dynamic LES tasks with stochastic state transitions. Examples of such LES tasks include a robot tracking a moving target, or consensus tasks for which the cost of hiring a worker changes with per dynamics of the market. In dynamic LES tasks, we do not have a single, fixed world state; the world state may be changing as the system collects additional observations. Thus, the state sampled from a terminal belief state with the *SampleTrueState* function cannot be used directly to evaluate earlier belief states. The *SampleEarlierTrueState* function samples states for earlier belief states on an execution path in a manner consistent with the state sampled for the terminal belief state. It uses the transition function T, which models the way the world state changes, to sample earlier belief states consistently. The function takes as input s_{t+1}, the sampled state for time $t + 1$, and b_t, the system's belief state at time t, and samples s_t, a state for time t. Using Bayes rule, the likelihood of sampling s_t is equal to:

$$Pr_c(S_t = s_t | S_{t+1} = s_{t+1}, B_t = b_t) \propto T(s_t, c, s_{t+1})\, b_t(s_t)$$

The convergence analysis for MC-VOI follows from previous work on Monte Carlo planning [15, 22]. The analysis is simplified as MC-VOI does not perform action selection. With each execution path sampled, MC-VOI updates the utility of taking any action on every belief state encountered on the path. Under the assumption of accurate belief states, observation and transition functions, and in the limit of infinite samples, the tree generated by MC-VOI constitutes the complete search tree, and, by induction, the values assigned to each belief state–action pair are the true values that would be computed by an exact solution.

MC-VOI's characteristics generalize to LES tasks that have multiple actions for collecting evidence. For example, a system for solving a consensus task may need to make decisions about which worker to hire, and which observations

```
SampleExecutionPath($b_t$:belief state, $l$:horizon)
begin
    if $\neg IsTerminal(b_t, l)$ then
        $b_{t+1} \leftarrow SampleNextBeliefState(b_t)$
        $s_{t+1} \leftarrow SampleExecutionPath(b_{t+1}, l)$
        $s_t \leftarrow SampleEarlierTrueState(b_t, s_{t+1})$
    else
        $s_t \leftarrow SampleTrueState(b_t)$
    end
    $Evaluate(b_t, s_t, l)$
    return $s_t$
end
```

Algorithm 2: Updated *SampleExecutionPath* function for dynamic LES tasks.

to gather. For such tasks, the action set includes action $\neg c$, representing domain actions, and a set of evidence gathering actions. The algorithm can employ an action selection rule proposed by the previous work to sample evidence gathering actions [15] to generate an execution path. Once an execution path is sampled, the algorithm can call the *Evaluate* function recursively to evaluate the utility of collecting more observations and the utility of taking a domain action simultaneously with a single sampled state.

LES tasks present challenges with long sequences of weak evidence and noisy belief state estimations that have not been addressed by studies of Monte Carlo planning algorithms applied to fully observable domains. MC-VOI differs from existing algorithms in its leveraging of the special structure of LES tasks in both is exploration of the search space and the way it resolves uncertainty. It can evaluate the utility of any action outcome sequence on an execution path with a single sample, thus requires fewer number of samples to explore long horizons associated with LES tasks. The sampling procedure of MC-VOI needs a single sample to explore leaves close to the root as well as leaves close to the horizon. In contrast, the sampling procedures of existing Monte Carlo algorithms initially favor leaves close to the root, requiring significantly larger samples to explore leaves close to the horizon, when the horizon is large. The way the state is sampled in MC-VOI leverages the situation where belief states closer to the horizon have less error, as these states will tend to incorporate a relatively large set of evidence. Because the algorithm samples the first true state at the end of the horizon based on all evidence collected, and evaluates earlier belief states accordingly, errors on the rewards of early belief states can be corrected. This procedure differs from the approach taken by existing algorithms, which sample a true state at the initial state and propagate it to future states [22].

6. EXPERIMENTS AND DISCUSSION

We evaluate the performance of MC-VOI on two separate sets of experiments. The first set of experiments is performed on synthetic data that represents a LES task inspired from a dialogue system. The second set of experiments is performed on a real-world testing set collected from the Galaxy Zoo system.

In these experiments, we compare the performance of MC-VOI with baselines, limited lookahead planning algorithms, and UCT and MCTS–two well-known Monte Carlo planning algorithms. *No collection* represents the baseline that chooses a domain action without collecting observations. *Collect all* baseline collects all available observations before choosing a domain action. The limited lookahead algorithms solve a LES-MDP up to a certain depth. The depth that can be considered by these algorithms is limited in practice, as the complexity grows exponential in the lookahead depth. UCT and MCTS use sampling to explore the search space of LES tasks. These procedures compute values for taking different actions by reasoning about the values of future belief states and the cost of collecting information. UCT uses regret analysis to sample actions [15]. We experimented with different C values for the UCT algorithm and report results for the value that performed the best. We customized the Monte Carlo tree search algorithm (MCTS) given in [22] by limiting it to sample a (belief state, $\neg c$) pair only once to prevent sampling fixed values multiple times. All algorithms are given access to the same belief update models. In our experiments, we vary the computational complexity of lookahead approaches by changing the lookahead depth. MC-VOI, UCT and MCTS algorithms are flexible, anytime algorithms; they can be stopped anytime to produce a result. We provided for each experimental condition, equal running time to MC-VOI, UCT, and MCTS algorithms. Our C# implementations of these algorithms are tested on a machine with 267 GHz CPU and 24 GB RAM.

6.1 Experiments on Synthetic Dataset

We generated a synthetic dataset with methods inspired by efforts to develop a spoken dialog system that assists different users with tasks of daily life. Before engaging, the system works to correctly identify a user based on observations made by its face recognition component. The system makes guesses about the identity of a user based on images collected at every second using its cameras. The observations become more informative as the recognition component collects larger sets of evidences and this long sequence of evidence comes together in synergistic manner. At any time, the system needs to make a decision about whether to engage with the user immediately, or delay the engagement to enable the collection of additional evidence about the user's identity. The system is rewarded for correctly predicting a user, but delaying the engagement incurs costs.

The domain from which the synthetic data is generated is formalized as follows: n is the number of possible users, I is the set of all users, l represents the horizon of the recognition task. The belief state of the system at time t is a probability distribution over I. Prior probabilities over users are generated randomly for each task. $o_t \in I$, the observation at time t, represents the guess of the recognition component about the identity of the user at time t. For simplicity, we assume that the likelihood of an observation at time t only depends on the correct identity of the user and time t. To simulate future observations having supermodular influences in belief updates, we model observations to offer weak evidences initially, but to become more informative as long sequences of observations are collected. The likelihood of an observation matching the true identity at time t is given as $1/n+((n-1)/n)(t/l)$. The likelihood the observation matching any incorrect identity is equally likely. After receiving an observation, the belief is updated using the Bayes rule. The system is rewarded 1$ for a correct prediction, and incurs a constant cost (in cents) for each second the engagement is delayed.

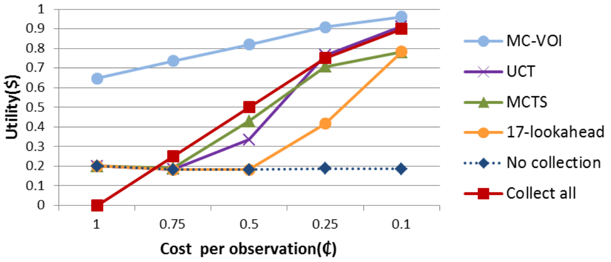

Figure 2: Performance of algorithms on synthetic dataset with varying costs of observation.

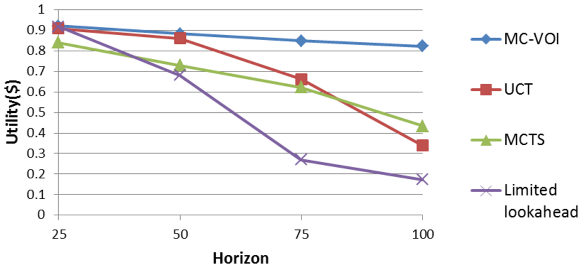

Figure 3: Performance of algorithms on tasks with varying horizon values when cost is 0.5.

The key to solving such streaming observational tasks is to successfully trade the immediate cost of an observation with the expected benefit of collecting more observations on future actions. Accurately performing this tradeoff may be more challenging for some cost values depending on the specifics of a LES task. To investigate how different algorithms perform under various conditions, we vary the cost of an observation from low to high values and report whether these algorithms can make effective decisions in all range of observation costs. Figure 2 compares the performance of different algorithms for 1000 randomly generated problem instances, where n is set to 10, l is 100, and cost varies between 1 and 0.1 cents. The performance gap between MC-VOI and other Monte Carlo planning algorithms grows proportional to the cost of an observation, as, for large observation costs, MCTS and UCT cannot explore long horizons to properly evaluate the value of evidence collection in the given amount of time, and thus terminate evidence collection prematurely.

LES tasks are associated with long horizons which need to be explored for making effective decisions. In the next set of experiments, we increase the horizon from 25 to 100, which also increases the average number of observations to be collected to correctly identify a user. Figure 3 shows that MC-VOI performs consistently well for different horizon values, whereas the gap between MC-VOI and other algorithms increases with the horizon.

For the results given in Figures 2 and 3, the average running times of heuristic approaches are less than 1 millisecond (ms), the running times of MC-VOI, UCT and MCTS algorithms are 330 ms, and the running time of 17 step lookahead is 352 ms. In separate experiments, we varied the running times of MC-VOI, UCT, MCTS and limited lookahead algorithms from 1 ms to 1500 ms. For different running times, MC-VOI consistently performed better than other algorithms when all algorithms are provided the same running time.

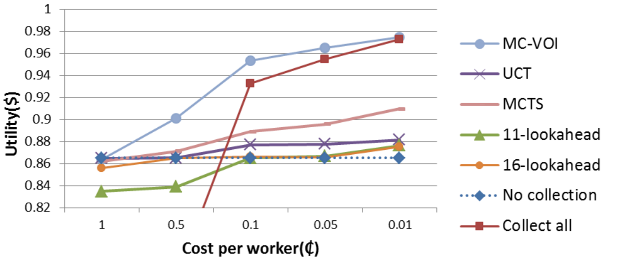

Figure 4: Performance of algorithms for Galaxy Zoo tasks with varying worker costs.

6.2 Experiments on Galaxy Zoo

Next, we evaluate the performance of MC-VOI on a testing set collected from Galaxy Zoo. The dataset includes 44350 votes for 1000 randomly selected galaxies, and a set of features describing each galaxy derived from image processing. We employed limited lookahead (depths 11 and 16) with MC-VOI, UCT and MCTS so as to maximize the total utility for classification of the 1000 galaxies. Every algorithm has access to the same answer and vote models to model Galaxy Zoo tasks. Approaches that reason about a single next worker to hire perform identical to the *No collection* baseline, as no single worker is informative enough to change the system's initial prediction of the correct answer. Heuristic and statistical approaches that are proposed for similar domains in previous work (i.a., [21]) are omitted since they do not have the ability to trade off the expected benefit and cost of hiring a sequence of workers, and do not perform well consistently for varying worker costs.

Figure 4 compares the performance of different algorithms for the case in which the system is rewarded $1 for correctly predicting the correct answer of a task. As the answer models cannot predict the correct classification of a galaxy perfectly, even when all worker reports available for the galaxy are collected, the accuracy of the answer model (97.7%) when the maximum number of votes are collected serves as an upper bound for the average utilities. We vary the cost of hiring a worker between 1 and 0.01 cents. This cost can represent monetary or time costs for hiring workers in different crowdsourcing and citizen science systems. For the results given in Figure 4, the running times for computing baselines are less than 1 ms, the running times of MC-VOI, UCT and MCTS algorithms are 1885 ms, and the average running times of limited lookahead algorithms are 255 and 7891 ms respectively for lookahead depths of 11 and 16. Figure 4 shows that for costs higher than 1 cent, the performance of MC-VOI is significantly better than all other baselines and planning algorithms. Although not obvious from the figure, MC-VOI continues to perform better than *Collect all* baseline when the cost is 0.01 cents; MC-VOI can reach the same accuracy as this baseline by only hiring 54% of all available workers.

Figure 5 compares the performance of different algorithms

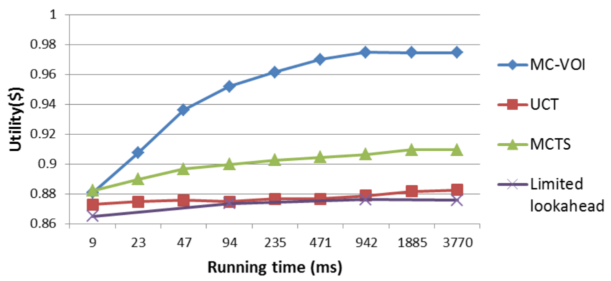

Figure 5: Influence of running time on performance of planning algorithms when cost is 0.01 cents.

for varying running times when the cost of a worker is 0.01 cents. We observe similar behavior for other cost values. The comparison shows that the other planning algorithms cannot reach the performance of the MC-VOI algorithm, even when they are provided larger running times. For many Galaxy Zoo tasks, all planning algorithms except MC-VOI terminate evidence collection prematurely because they cannot accurately estimate VOI. The number of state–action outcome sequences to be explored by UCT and MCTS algorithms grows exponentially in the horizon, which prevents these algorithms from exploring the long horizons of these LES tasks in the running times provided. Moreover, the performances of these algorithms are influenced negatively by the noise in predictive models in evaluating the values of taking different actions. The MC-VOI algorithm is able to explore the search space of a LES task efficiently by using its special structure, and its performance appears to be robust to noise in beliefs inferred by the answer models given the approach to evaluation.

7. SUMMARY AND FUTURE WORK

We presented MC-VOI as a procedure for computing VOI for tasks requiring the consideration of long evidential sequences. The tractability of the procedure hinges on exploiting the special structure of LES tasks for exploring the search space. We demonstrated that the algorithm outperforms existing methods with experiments on a synthetic dataset and on a challenging crowdsourcing problem.

In ongoing work, we seek to understand the behavior of MC-VOI across a spectrum of domains, including systems that perform spoken dialogue and medical diagnosis. We are also investigating extensions of MC-VOI for tasks that may not terminate after taking a domain action. We believe that a Monte Carlo approach to computing information value will be valuable for fielding tractable solutions when we cannot rely on assumptions of submodularity. More generally, we hope that MC-VOI methods and results will inspire efforts to better understand the use of simulation methods to effectively probe intractable combinatorial spaces for solving hard problems in machine intelligence.

8. ACKNOWLEDGMENTS

We thank Chris Lintott for sharing Galaxy Zoo data, Paul Koch for assistance with accessing the data, and Dan Bohus, Rich Caruana, Ashish Kapoor, Paul Koch, Barbara Grosz, and Chris Lintott for discussions and feedback.

9. REFERENCES

[1] Galaxy Zoo, 2007, http://zoo1.galaxyzoo.org/.

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

[3] M. Bilgic and L. Getoor. Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *JAIR*, 2011.

[4] D. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *UAI*, pages 80–89, 1997.

[5] S. Dittmer and F. Jensen. Myopic value of information in influence diagrams. In *UAI*, pages 142–149, 1997.

[6] G. Gorry, J. Kassirer, A. Essig, and W. Schwartz. Decision analysis as the basis for computer-aided management of acute renal failure. *AJM*, 1973.

[7] H. Hajishirzi, A. Shirazi, J. Choi, and E. Amir. Greedy algorithms for sequential sensing decisions. In *IJCAI*, pages 1908–1915, 2009.

[8] E. Hansen and S. Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1):139–157, 2001.

[9] D. Heckerman, E. Horvitz, and B. Middleton. An approximate nonmyopic computation for value of information. *TPAMI*, 15(3):292–298, 1993.

[10] D. Heckerman, E. Horvitz, and B. Nathwani. Toward normative expert systems. Part I: The Pathfinder project. *Methods of Information in Medicine*, 1992.

[11] R. Howard. Information value theory. *Systems Science and Cybernetics*, 2(1):22–26, 1966.

[12] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[13] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in a large-scale crowdsourcing system. In *AAMAS*, 2012.

[14] M. Kearns, Y. Mansour, and A. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, 1999.

[15] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. *ECML*, pages 282–293, 2006.

[16] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, 2007.

[17] W. Liao and Q. Ji. Efficient non-myopic value-of-information computation for influence diagrams. *International Journal of Approximate Reasoning*, 49(2):436–450, 2008.

[18] A. Mahajan and D. Teneketzis. Multi-armed bandit problems. *Foundations and Applications of Sensor Management*, pages 121–151, 2008.

[19] G. Peskir and A. Shiryaev. *Optimal stopping and free-boundary problems*, volume 10. 2006.

[20] S. Reches, M. Kalech, and R. Stern. When to stop? that is the question. In *AAAI*, 2011.

[21] V. Sheng, F. Provost, and P. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *ACM SIGKDD*, 2008.

[22] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. 2010.

[23] Y. Zhang. Multi-task active learning with output constraints. In *AAAI*, 2010.