Efficient Online Learning for Optimizing Value of Information: Theory and Application to Interactive Troubleshooting

Yuxin Chen
Caltech

Jean-Michel Renders
Xerox Research Center Europe

Morteza Haghir Chehreghani
Xerox Research Center Europe

Andreas Krause
ETH Zurich

Abstract

We consider the optimal value of information problem, where the goal is to sequentially select a set of tests with a minimal cost, so that one can efficiently make the best decision based on the observed outcomes. Existing algorithms are either heuristics with no guarantees, or scale poorly (with exponential run time in terms of the number of available tests). Moreover, these methods assume a known distribution over the test outcomes, which is often not the case in practice.

We propose a sampling-based online learning framework to address the above issues. First, assuming the distribution over hypotheses is known, we propose a *dynamic hypothesis enumeration* strategy, which allows efficient information gathering with strong theoretical guarantees. We show that with sufficient amount of samples, one can identify a near-optimal decision with high probability. Second, when the parameters of the hypotheses distribution are unknown, we propose an algorithm which learns the parameters progressively via posterior sampling in an *online* fashion. We further establish a rigorous bound on the expected regret. We demonstrate the effectiveness of our approach on a real-world interactive troubleshooting application, and show that one can efficiently make high-quality decisions with low cost.

1 INTRODUCTION

Optimal information gathering for decision making is a central challenge in artificial intelligence. A classical approach for decision making is the decision-theoretic *value of information* (VoI) [15], where one needs to find an optimal testing policy which achieves the maximal value of information. Informally, the goal of the optimal policy is to reduce the uncertainty about some *hidden state* of the system in question, by efficiently probing it via a sequence of *tests* and observations, so that one can make the best *decision* with the minimal cost. For example, consider the automated troubleshooting problem, where a customer reaches a contact center and wants to resolve some problem with her cell phone. To provide a solution, a virtual agent has to ask the customer a few questions regarding the symptoms of the cellphone to diagnose the root-cause. Here, "decision" corresponds to the solution, "hidden state" corresponds to the root-cause which we want to learn about, "tests" correspond to questions on the symptoms, and the "hypothesis space" consists of full realizations of all tests. We want to develop a virtual agent, which can identify the best solution for the customer with a minimal set of questions asked.

Optimization of VoI. The optimal VoI problem has been studied in various contexts, including active learning [10, 29], Bayesian experimental design [6], policy making [27] and probabilistic planning [17, 30], etc. We refer interested readers to §6 for a more detailed review of the related work. Deriving optimal policies is NP-hard in general [5]; however, under certain conditions some approximation results are known. In particular, if test outcomes are deterministic functions of the hidden state (i.e., noise-free setting), then a simple greedy algorithm, namely *generalized binary search* (GBS), is guaranteed to provide a near-optimal approximation of the optimal policy in terms of the cost [20]. These results have recently been generalized to decision making, where information gathering policies no longer aim to resolve all uncertainty of the hidden state – but just enough to make the optimal decision. Such problem, known as the *Decision Region Determination* (DRD) problem [16], relates the problem of learning the optimal policy with maximal utility, to the problem that aims at resolving the

uncertainty amongst the decisions. Following this line of work, Javdani et al. [16] and Chen et al. [8] propose a principled framework for optimizing the VoI using surrogate objective functions. The theoretical guarantees of these algorithms rely on the fact that the objective functions exhibit *adaptive submodularity* [12], a natural diminishing returns property that generalizes the classical notion of submodularity to adaptive policies. It follows that a simple greedy policy can provide near-optimal approximation to the optimal (intractable) solution.

Limitations of existing methods. In many data-intensive decision making applications, however, evaluating these surrogate objectives is expensive. First, let us assume that the underlying distribution over hypotheses is given. At each iteration, one needs to perform a greedy search over the tests and find the one that myopically maximizes the expected gain in the corresponding objective, whose runtime depends linearly on the size of the support of the probability distribution over the hypotheses. However, with the size of the hypothesis space growing exponentially in the number of tests, it is often computationally prohibitive to work with the original distribution. Second, in practice, the underlying distribution over hypotheses is often unknown, and requires to be estimated (and learned) over time. Within an online framework for solving the troubleshooting problem, we assume that the virtual agent does not possess a perfect knowledge of which root-causes correspond to which symptoms. Thus, to provide better diagnosis in the long run, the virtual agent must engage customers to answer more explorative questions during each session, while not spamming the customer with excessive queries.

Our contribution. In this paper, we make contributions to both fronts. First, assuming that the prior over hypotheses is given, we propose an efficient hypothesis enumeration scheme, which makes the class of adaptive submodular surrogates practically feasible, while still preserving strong theoretical guarantees. In particular, through a "divide-and-conquer" strategy, we generate the most probable hypotheses conditioning on each hidden state with a novel and efficient *priority search* procedure, and then merge them over all states to compute their marginal likelihood. In comparison with prior art, our sampling scheme utilizes the specific structure of the underlying model, and thereby offers increased efficiency and better approximation guarantees.

As our second contribution, we integrate our hypothesis enumeration strategy for optimizing VoI into an *online* sequential information gathering framework, where the conditional probabilities of test outcomes given the hidden states are unknown, and can only be learned from data in an online fashion. For instance, in troubleshooting, the conditional probabilities of symptoms given a root-cause might be unknown. For this purpose, we employ a *posterior sampling* approach, where for each decision-making session (i.e., for each customer), we first sample parameters of the conditional probability distributions according to their probabilities of being "optimal" (in the sense that they reflect the true parameters), and then use our hypothesis enumeration algorithm to generate hypotheses for that session.

We further establish a rigorous bound on the *expected regret* (defined in terms of the value of information) of our algorithm. Our online learning strategy can be interpreted as Thompson sampling across multiple sessions of interaction. Several recent empirical simulations [7, 14, 28] and theoretical studies [1, 4, 19] have demonstrated the effectiveness of Thompson sampling in different settings. However, different from our framework, the classical usage of Thompson sampling [32] suggests to choose an action according to its probability of being optimal, i.e. the action which maximizes the reward in expectation; whereas in our model, the "action" can be interpreted as the set of tests performed in one decision-making session.

Finally, we demonstrate our online learning framework on a real-world troubleshooting platform. Our empirical results show that one can efficiently run the "submodular surrogate"-based approaches with our dynamic hypothesis enumeration strategy, while achieving much better performance comparing with existing commonly-used heuristics (we observe a 16% improvement on the average cost on our troubleshooting dataset). Our experiments under the online setting imply that our framework encourages efficient exploration, which, combined with the hypothesis enumeration algorithm, leads to efficient online learning of the optimal VoI.

2 VALUE OF INFORMATION

In this section we introduce basic notations, and formally state the VoI problem and existing methods for solving it.

Formulating VoI as a DRD problem. Let $Y \in \mathcal{Y} \triangleq \{y_1, \ldots, y_m\}$ be a random variable that represents some *hidden state*, upon which we want to make a *decision*. The reward of making decision $d \in \mathcal{D}$ for hidden state $y \in \mathcal{Y}$ is modeled by a utility function $u : \mathcal{D} \times \mathcal{Y} \to [0, 1]$. We are given a set of *tests* $\mathcal{T} \triangleq \{1, \ldots, n\}$ of *binary* outcomes; performing each test $t \in \mathcal{T}$ reveals some information about Y, and incurs some cost which is given by a cost function $c : \mathcal{T} \to \mathbb{R}_{\geq 0}$. Let $\mathcal{H}$ denote the ground set of *hypotheses*; each hypothesis $h \in \mathcal{H}$ corresponds to a possible realization of the out-

comes of all tests in $\mathcal{T}$. In other words, the *outcome* of test t can be modeled as a deterministic function of h, i.e., $g_t : \mathcal{H} \to \{0,1\}$. Let $X_t \in \{0,1\}$ be the random variable corresponding to the outcome of test t, and $H = [X_1, \ldots, X_n]$ be the random variable over $\mathcal{H}$. We use x_t to denote the observed outcome of test t. Crucially, we assume that X_i's are *conditionally independent* given the hidden state Y, i.e., $\mathbb{P}[Y, X_1, \ldots, X_m] = \mathbb{P}[Y] \prod_{i=1}^{n} \mathbb{P}[X_i \mid Y]$. In such *offline* setting, we assume that the parameters of the above distributions are given.

Denote the set of performed tests by $\mathcal{A}$ and their outcome vector by $\mathbf{x}_\mathcal{A}$. We define $U(d \mid \mathbf{x}_\mathcal{A}) \triangleq \mathbb{E}_y[u(y,d) \mid \mathbf{x}_\mathcal{A}]$ to be the expected utility of making decision d after observing $\mathbf{x}_\mathcal{A}$. The *value* of a specific set of observations $\mathbf{x}_\mathcal{A}$ is then defined as: $\mathrm{VoI}(\mathbf{x}_\mathcal{A}) \triangleq \max_{d \in \mathcal{D}} U(d \mid \mathbf{x}_\mathcal{A})$, i.e., the maximum expected utility achievable when acting upon observations $\mathbf{x}_\mathcal{A}$. For each decision $d \in \mathcal{D}$, we define its associated *decision region* as $\mathcal{R}_d \triangleq \{h : U(d \mid h) = \mathrm{VoI}(h)\}$, i.e., the set of hypotheses for which d is the optimal decision.

Formally, a *policy* $\pi : 2^{\mathcal{T} \times \{0,1\}} \to \mathcal{T}$ is a partial mapping from the set of test-observation pairs to (the next) tests. The expected cost of a policy π is $\mathrm{cost}_{av}(\pi) \triangleq \mathbb{E}_h\left[\sum_{i:(i,x_i) \in \mathcal{S}(\pi,h)} c(i)\right]$, and worst-case cost is $\mathrm{cost}_{wc}(\pi) \triangleq \max_h \sum_{i:(i,x_i) \in \mathcal{S}(\pi,h)} c(i)$, where $\mathcal{S}(\pi,h)$ represents the set of tests (and their outcomes) seen by π given that hypothesis h happens. The goal of the DRD problem is to find an optimal policy π^* with a minimal cost (expected or worst-case), such that upon termination, there exists at least one decision region that contains all hypotheses consistent with the observations acquired by the policy. Formally, we seek

$$\pi^* \in \arg\min_{\pi} \mathrm{cost}(\pi), \quad \text{s.t. } \forall h \; \exists d : \mathcal{H}(\mathcal{S}(\pi,h)) \subseteq \mathcal{R}_d. \tag{1}$$

where $\mathcal{H}(\mathcal{S}(\pi,h)) = \{h' \in \mathcal{H} : (i,x) \in \mathcal{S}(\pi,h) \Rightarrow g_i(h') = x\}$ is the set of hypotheses consistent with $\mathcal{S}(\pi,h)$.

The *Equivalence Class Edge Cutting* algorithm. For simplicity, we consider the special case of the DRD problem where the decision regions are disjoint[1]. In such case, the DRD problem reduces to the *equivalence class determination* problem, which can be solved near-optimally by the *Equivalence Class Edge Cutting* (EC^2) algorithm [13].

As is illustrated in Fig. 1, EC^2 employs an edge-cutting strategy based on a weighted graph $G = (\mathcal{H}, E)$, where vertices represent hypotheses, and edges link hypotheses that we want to distinguish between. Formally, $E \triangleq \bigcup_{d \neq d'} \{\{h,h'\} : h \in \mathcal{R}_d, h' \in \mathcal{R}_{d'}\}$ consists of all (unordered) pairs of root-causes corresponding to different target decisions (see Fig. 1b). We define a weight function $w : E \to \mathbb{R}_{\geq 0}$ by $w(\{h,h'\}) \triangleq \mathbb{P}[h] \cdot \mathbb{P}[h']$, i.e., as the product of the probabilities of its incident root-causes. We extend the weight function on sets of edges $E' \subseteq E$, as the sum of weight of all edges $\{h,h'\} \in E'$, i.e., $w(E') \triangleq \sum_{\{h,h'\} \in E'} w(\{h,h'\})$.

Performing test $t \in \mathcal{T}$ with outcome x_t is said to "*cut*" an edge, if at least one of its incident root-causes is inconsistent with x_t (See Fig. 1c): Formally, the set of edges cut by observing x_t is $E(x_t) \triangleq \{\{h,h'\} \in E : \mathbb{P}[x_t \mid h] = 0 \;\vee\; \mathbb{P}[x_t \mid h'] = 0\}$. EC^2 then greedily selects the test that maximizes the total expected weight of edges cut per unit cost. The performance guarantee of EC^2 relies on the fact that the objective function (i.e., the total weight of edges cut) is *adaptive submodular*, and *strongly adaptive monotone* [12]. In particular, let $p_{\min} \triangleq \min_{h \in \mathcal{H}} \mathbb{P}[h]$ denote the minimal prior probability of any hypothesis; the expected cost of EC^2 is bounded by an $O(\log(1/p_{\min}) + 1)$ factor[2] of the minimal expected cost.

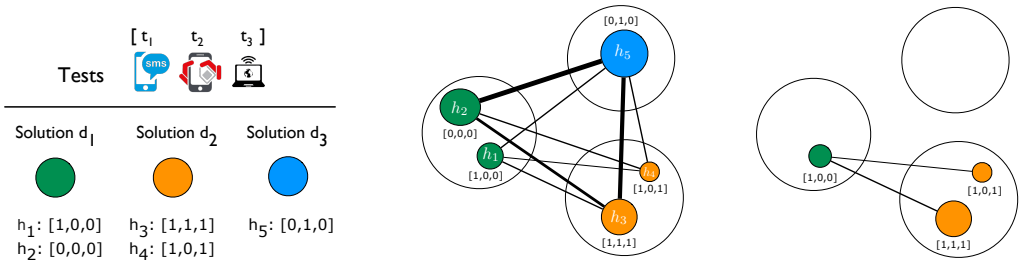
3 EFFICIENT OPTIMIZATION OF VOI VIA HYPOTHESIS ENUMERATION

Note that to compute the exact EC^2 objective, we have to enumerate all hypotheses in $\mathcal{H}$ of non-zero prior probability. The total number of hypotheses is exponential with respect to the number of tests; hence, direct application of EC^2 does not scale up to several hundreds of tests or more. To facilitate efficient optimization, we must consider effective sampling schemes to explore the hypothesis space. We aim to maintain a "confident set" of hypotheses to efficiently approximate the EC^2 objective. Concretely, we consider the following problem:

The optimal hypothesis enumeration problem. Assume the prior $\mathbb{P}[Y]$ on the hidden state is known, and the prior distribution over hypotheses are fully specified by the conditional probability distribution table (CPT): $\boldsymbol{\theta} = [\theta_{ij}]_{n \times m}$, where $\theta_{ij} \triangleq \mathbb{P}[X_i = 1 \mid Y = y_j]$ for test $i \in [n]$ and hidden state $j \in [m]$. Let $\tilde{\mathcal{H}}$ be the set of hypotheses sampled from the CPT. Clearly, an "ideal" set $\tilde{\mathcal{H}}$ for EC^2 should be (1) rich enough to enclose promising candidates of *true* underlying hypotheses, and (2) compact enough so that it excludes hypotheses that are extremely rare and ensures feasibility of the algorithm. To this end, we define the *coverage* of $\tilde{\mathcal{H}}$ as its total proba-

[1] Note that our algorithmic framework can also be directly applied to the general DRD setting with overlapping decision regions.

[2] Throughout this paper all the logs are in base 2.



(a) A troubleshooting example (b) Initialization of EC^2 (c) Observing $X_1 = 1$

Figure 1: Illustration of the EC^2 algorithm of [13]. Fig. (a) shows an illustrative example of the troubleshooting problem. In this example, we assume that there are three possible troubleshooting decisions we can make (i.e., solutions); each of them corresponds to one or many hypotheses (e.g., realizations of test outcomes). The goal is to identify the best solution for a given problem as quickly as possible. In (b), we initialize EC^2, by drawing edges between all pairs of hypotheses (solid circles) that are mapped into different decisions (hollow circles). In (c), we perform test 1 and observe $X_1 = 1$. As a result, all the edges incident to hypotheses $h_2 : [0, 0, 0]$ and $h_5 : [0, 1, 0]$ are cut/ removed.

bility mass: $Z(\tilde{\mathcal{H}}) = \sum_{h\in\tilde{\mathcal{H}}} \mathbb{P}[h]$, and the coverage of $\tilde{\mathcal{H}}$ conditioning on y as $Z(\tilde{\mathcal{H}} \mid y) = \sum_{h\in\tilde{\mathcal{H}}} \mathbb{P}[h \mid Y = y]$. We aim to attain a high coverage over $\mathcal{H}$ using samples, while keeping the sample size as small as possible. Formally, to achieve $1 - \eta$ coverage, we seek

$$\tilde{\mathcal{H}}^* = \underset{\tilde{\mathcal{H}}: Z(\tilde{\mathcal{H}})\geq 1-\eta}{\arg\min} \; |\tilde{\mathcal{H}}|.$$

Existing approaches for generating hypotheses, such as Monte-Carlo sampling, often require a large sample size to reach a certain coverage of the total probability mass. To illustrate this, let us consider a simple multinomial distribution that describes the probability distribution of four mutually exclusive hypotheses (h_1, h_2, h_3, h_4), with probabilities (0.94, 0.03, 0.02, 0.01). A Monte-Carlo hypothesis generator simply samples hypotheses according to their probabilities (as we were rolling a dice). If we require to observe a subset of hypotheses that cover at least 98% of the total probability mass (i.e. h_1, h_2 and at least one of h_3 or h_4) with a confidence level of at least 99%, then we need at least a sample of average size 174, to cover the "rare" observations.

Dynamic hypothesis enumeration. The problem of the Monte-Carlo approach is that it is ignorant of the structure of the VoI problem. Instead, our method aims at providing the most likely configurations – covering up to a pre-specified fraction of the total probability mass – in an efficient and adaptive way. In a nutshell, we adaptively maintain a pool of hypotheses that constitute a small sample with sufficient coverage. In particular, our hypothesis enumeration scheme consists of two modules: (1) Algorithm 1 *locally* enumerates the *most likely* hypotheses for each hidden state, which will cover – by taking the union over all hidden states – at least

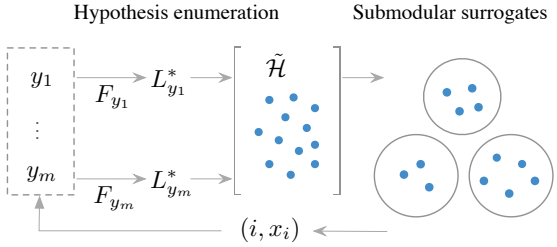


Figure 2: The dynamic hypothesis enumeration framework for optimizing VoI.

$(1 - \eta)$ fraction of the total probability mass of all hypotheses; and (2) Algorithm 2, as illustrated in Fig. 2, provides a *global* mechanism that, after observing a test outcome, adaptively filters out inconsistent hypotheses and *re-generates* new hypotheses by calling Algorithm 1.

3.1 ENUMERATING HYPOTHESES

The basic module of our hypothesis enumeration framework is a "local" hypothesis generator, which enumerates the most likely hypotheses for any given hidden state. It incrementally builds a Directed Acyclic Graph[3] (DAG) of hypotheses, starting from the most likely configuration. At each step, the leaf nodes of the DAG represent the current *candidate frontier*, i.e., the set of hypotheses that dominate all other candidate hypotheses in terms of likelihood. This set is used to generate the remaining hypotheses through a "children generation" mechanism: the next most likely hypothesis of *candidate frontier* is identified, and its (at most two) children

[3]Note that this DAG is used as a data structure for hypothesis enumeration, and is *different* from the (undirected) weighted graph used for running the EC^2 algorithm.

Algorithm 1: Generate the most likely hypotheses for y

1 **Input**: Hidden state y, Conditional probability table $\boldsymbol{\theta}$, coverage threshold η, (optional) frontier F_y;
begin
2 Sort tests in decreasing order of $\mathbb{P}[X_i = 1 \mid y]$;
foreach $i \in \{1, \ldots, n\}$ **do**
3 $p_i \leftarrow \log(\mathbb{P}[X_i = 1 \mid y])$;
4 $q_i \leftarrow \log(\mathbb{P}[X_i = 0 \mid y])$;
if F_y *is empty* **then**
5 $F_y \leftarrow \{h_1 = [1, 1, \ldots, 1]\}$, with log-weight $\lambda_y(h_1) = \sum_i \log p_i$;
6 $L_y^* \leftarrow \emptyset$;
while $\sum_{h \in L_y^*} \exp(\lambda_y(h)) < 1 - \eta$ **do**
7 $h^* \leftarrow \arg\max_{h \in F_y} \lambda_y(h)$;
8 $F_y \leftarrow F_y \setminus \{h^*\}$, $L_y^* \leftarrow L_y^* \cup \{h^*\}$;
9 Generate (at most) 2 children h_{c_1}, h_{c_2} from h^*;
10 $F_y \leftarrow F_y \cup \{h_{c_1}, h_{c_2}\}$;
11 **Output**: Most likely hypotheses L_y^* for y, log-probabilities $\lambda_y(h) = \log(\mathbb{P}[h \mid y, \mathbf{x}_\mathcal{A}])$, and F_y.

Algorithm 2: Iterative Filtering and Re-sampling

1 **Input**: Conditional probability table $\boldsymbol{\theta}$, Prior $\mathbb{P}[Y]$, coverage threshold η;
begin
2 $\tilde{\mathcal{H}} \leftarrow \emptyset$;
while *stopping condition for* EC^2 *not reached* **do**
foreach $y \in \{y_1, \ldots, y_m\}$ **do**
3 Call Algorithm 1 to generate L_y^*;
4 $\tilde{\mathcal{H}} \leftarrow \tilde{\mathcal{H}} \cup L^*$;
foreach $h \in \tilde{\mathcal{H}}$ **do**
5 $p(h \mid \mathbf{x}_\mathcal{A}) \leftarrow \sum_y \exp(\lambda_y(h)) \cdot \mathbb{P}[y \mid \mathbf{x}_\mathcal{A}]$;
6 Run EC^2 to determine the next test t;
$\mathcal{A} \leftarrow \mathcal{A} \cup \{t\}$;
7 Observe x_t; $\mathbf{x}_\mathcal{A} \leftarrow \mathbf{x}_\mathcal{A} \cup \{x_t\}$;
8 Update $\mathbb{P}[y \mid \mathbf{x}_\mathcal{A}]$;
9 $\lambda_y(h) \leftarrow \lambda_y(h) - \log \mathbb{P}[x_t \mid y]$;
10 Filter out inconsistent hypotheses in L_y^* and F_y;
11 Remove test t from the list of available tests;
12 **Output**: (test - outcome) vectors $\mathbf{x}_\mathcal{A}$, decision d

are added as new leaf nodes to the DAG.

The input of Algorithm 1 consists of the given hidden state value y, the associated outcome probability vector over n tests, i.e., $\mathbb{P}[x_i \mid y]$ $(i = 1, \ldots, n)$, and the threshold of coverage η. Optionally, it might be given a candidate frontier F_y, which is defined as a list of consistent hypotheses h with their log-probability weights $\lambda_y(h) = \mathbb{P}[h \mid y, \mathbf{x}_\mathcal{A}]$ conditioned on the hidden state value y and current observations $\mathbf{x}_\mathcal{A}$. F_y is obtained as a by-product when calling the same module for the same y at the previous iterations, and is used as a seed set of nodes to further expand the DAG.

W.l.o.g., we can assume that tests' outcomes are defined in such a way that $\mathbb{P}[X_i = 1 \mid y] \geq 0.5$.[4] Initially (line 2), the tests are rearranged in decreasing order of $\mathbb{P}[X_i = 1 \mid y]$. Thereby, the last test will be the one with the highest uncertainty; hence flipping the sign of this test will have the minimal effect on the overall likelihood. The generator then proceeds to enumerate the most likely hypotheses corresponding to the given hidden state y. At line 9, the two children hypotheses are generated as follows. For the first child, if the last (rightmost) bit of h^* is 1, we then create h_{c_1} by switching the last bit to 0. For instance, the child hypothesis h_{c_1} of $h^* = [0, 1, 1, 0, \mathbf{1}]$ is $[0, 1, 1, 0, \mathbf{0}]$. Its log-probability is obtained by $\lambda_y(h_{c_1}) = \lambda_y(h^*) + q_n - p_n$. For the second child, we first need to locate the right-most "$[1, 0]$" pair in h^* (if there exists any; otherwise we do nothing), and the create h_{c_2} by switching "$[1, 0]$" into "$[0, 1]$". For instance, the child hypothesis h_{c_2} of $h^* = [0, 1, \mathbf{1}, \mathbf{0}, 1]$ is $[0, 1, \mathbf{0}, \mathbf{1}, 1]$. Its associated log-probability is computed by $\lambda_y(h_{c_2}) = \lambda_y(h^*) + q_i - p_i + p_{i+1} - q_{i+1}$, where i is the bit index of the "1" in the right-most "$[1, 0]$" pair.

As output, Algorithm 1 produces a ranked list L_y^* of the most likely hypotheses for a given y, and their log-probabilities $\lambda_y(h) = \log(\mathbb{P}[h \mid y, \mathbf{x}_\mathcal{A}])$, such that $\sum_{h \in L_y^*} \exp(\lambda_y(h)) \geq 1 - \eta$. In addition, it also produces a residual frontier F_y that will be used as a new "seed" list for the next iteration.

3.2 ITERATIVE FILTERING AND HYPOTHESIS RE-SAMPLING

After generating the most likely hypotheses for each hidden state, we merge them into a global set and compute their marginal likelihoods. We dynamically re-generate new hypotheses as more observations are made. This step is necessary in order to constantly guarantee that the sample set covers at least $1 - \eta$ of the total remaining mass, after new observations become available.

As shown in Algorithm 2, the global iterative filtering and re-sampling module consists of a global loop, where after initializing all ranked lists L_y^* to $\emptyset$ and $\mathbb{P}[y \mid \mathbf{x}_\mathcal{A} = \emptyset]$ to the prior distribution over the hidden states, it iteratively performs the following sequences of operations: First, for each hidden state y, it calls Algorithm 1 to generate enough hypotheses so that L_y^* covers at least $(1 - \eta)$ of its current mass, i.e., $Z(L_y^* \mid y, \mathbf{x}_\mathcal{A}) \geq 1 - \eta$ (line 3). L_y^* might not be initially empty due to a

[4] Otherwise, we can redefine a test with flipped labels.

previous call to Algorithm 1. In this case, the generator produces only new additional hypotheses starting from the frontier F_y until the desired coverage is achieved. This step is not necessary for the y's that are inconsistent with $\mathbf{x}_\mathcal{A}$, i.e., for those hidden states whose posterior distribution given $\mathbf{x}_\mathcal{A}$ is zero.

Once we merge the hypotheses associated with each hidden state (line 4), the sample set $\tilde{\mathcal{H}}$ covers at least $(1-\eta)$ fraction of the total mass that is consistent with all the observations up to $\mathbf{x}_\mathcal{A}$: $Z(\tilde{\mathcal{H}} \mid \mathbf{x}_\mathcal{A}) = \sum_{h\in\tilde{\mathcal{H}}} \mathbb{P}[h \mid \mathbf{x}_\mathcal{A}] \geq \sum_y \sum_{h\in L_y^*} \mathbb{P}[h \mid y, \mathbf{x}_\mathcal{A}]\,\mathbb{P}[y \mid \mathbf{x}_\mathcal{A}] \geq \sum_y (1-\eta)\mathbb{P}[y \mid \mathbf{x}_\mathcal{A}] = (1-\eta)$. The procedure is then followed by performing EC^2 on $\tilde{\mathcal{H}}$ to identify the next test to be performed (see Fig. 2).

3.3 UPPER BOUNDS ON THE COST

Assume that we only enumerate the hypotheses *once* at the beginning of each experiment, i.e., we do not regenerate the hypotheses after observing the outcome of a test. If the underlying true hypothesis is included in the sampled set $\tilde{\mathcal{H}}$, then by construction, Algorithm 2 is guaranteed to make the optimal decision. Otherwise, with small probability it fails to output the optimal decision. Theorem 1 states a trade-off between the size of $\tilde{\mathcal{H}}$ and the expected cost of Algorithm 2.

Theorem 1. *Suppose we have generated hypotheses $\tilde{\mathcal{H}}$ with coverage $1-\eta$. Define $\tilde{p}_{\min} = \min_{h\in\tilde{\mathcal{H}}} \frac{\mathbb{P}[h]}{1-\eta}$. Let $\pi^g_{\tilde{\mathcal{H}}}$ be the policy induced by Algorithm 2, OPT be the optimal policy on the original distribution of $\mathcal{H}$, and $c(\mathcal{T})$ be the cost of performing all tests. Then, it holds that $\mathrm{cost}_{av}(\pi^g_{\tilde{\mathcal{H}}}) \leq (2\ln(1/\tilde{p}_{\min})+1)\,\mathrm{cost}_{av}(\mathsf{OPT}) + \eta\cdot c(\mathcal{T})$. Moreover, if we stop running $\pi^g_{\tilde{\mathcal{H}}}$ once it cuts all edges on $\tilde{\mathcal{H}}$, then with probability at least $1-\eta$, $\pi^g_{\tilde{\mathcal{H}}}$ outputs the optimal decision with $\mathrm{cost}_{wc}(\pi^g_{\tilde{\mathcal{H}}}) \leq (2\ln(1/\tilde{p}_{\min})+1)\,\mathrm{cost}_{wc}(\mathsf{OPT})$.*

We defer the proof to the supplemental material. Note that the expected cost is computed w.r.t. the original hypothesis distribution $\mathbb{P}[H \mid H \in \mathcal{H}]$. Theorem 1 establishes a bound between the cost of the greedy algorithm on the samples $\tilde{\mathcal{H}}$, and the cost of the optimal algorithm on the total population $\mathcal{H}$. The quality of the bound depends on η, as well as the structure of the problem (which determines $\tilde{p}_{\min}$). Running the greedy policy on a larger set of samples leads to a lower failure rate, although $\tilde{p}_{\min}$ might be significantly smaller for small η. Further, with adaptive re-sampling we constantly maintain a $1-\eta$ coverage on posterior distribution over $\mathcal{H}$. With similar reasoning, we can show that the greedy policy with adaptively-resampled posteriors yields a lower failure rate than the greedy policy which only samples the hypotheses once at the start of the session.

4 ONLINE LEARNING FOR OPTIMIZING VOI

In the online setting, the exact decision making model (i.e., the conditional probability table of test outcomes given the hidden state) is unknown, and we need to learn the model as we get more feedback. We employ an efficient *posterior sampling* strategy, described as follows. Suppose that initially we have access to a prior over the model parameters, for example, in the troubleshooting application, we assume a Beta prior on the parameters $[\theta_{ij}]_{n\times m}$ of the CPT, In the beginning of session ℓ, we sample $\boldsymbol{\theta}^{(\ell)}$ from $B(\boldsymbol{\alpha}^{(\ell)}, \boldsymbol{\beta}^{(\ell)})$. In particular, for each (root-cause - symptom) pair (y_j, x_i), we simply generate the parameter $\theta_{ij}^{(\ell)} = \mathbb{P}[x_i = 1 \mid y_j]$ from $B(\alpha_{ij}^{(\ell)}, \beta_{ij}^{(\ell)})$, where α_{ij}, β_{ij} depend on historical data. Then we run Algorithm 2 with $\boldsymbol{\theta}^{(\ell)}$ to sequentially pick tests for session ℓ. When a decision making session is over, we observe y_ℓ, together with a set of test-outcome pairs. We then update the distribution on $\boldsymbol{\theta}$ before we enter the next conversation. See Algorithm 3 for details[5].

Algorithm 3: Online sequential decision making

1 **Input**: α_{ij}, β_{ij} parameters of Beta distributions, prior $\mathbb{P}_\mathcal{Y}[Y]$, sessions / test scenarios $\{\mathrm{S}_1, \ldots \mathrm{S}_k\}$;
begin
2 Set $\alpha_{ij}^{(1)} \leftarrow \alpha_{ij}; \beta_{ij}^{(1)} \leftarrow \beta_{ij}$ for all i, j;
3 **foreach** $\ell = 1 \ldots k$ **do**
4 $A \leftarrow \emptyset, \mathbf{x}_\mathcal{A} \leftarrow \emptyset$;
5 Draw $\boldsymbol{\theta}^{(\ell)} = \{\theta_{ij}^{(\ell)} \sim B(\alpha_{ij}^{(\ell)}, \beta_{ij}^{(\ell)})\}$;
6 Call Algorithm 2 to engage session ℓ;
7 Observe $\mathbf{x}_\mathcal{A}$ and hidden state y_ℓ with index ϑ;
8 **foreach** $(i, x_i) \in \mathbf{x}_\mathcal{A}$ **do**
  if $x_i = 1$ **then**
  Set $\alpha_{i\vartheta}^{\ell+1} \leftarrow \alpha_{i\vartheta}^{\ell} + 1$;
  else
  Set $\beta_{i\vartheta}^{\ell+1} \leftarrow \beta_{i\vartheta}^{\ell} + 1$;

4.1 ONLINE REGRET BOUND

Suppose we have drawn $\boldsymbol{\theta}(\ell)$ for session ℓ. Denote the optimal policy w.r.t. this distribution by OPT_ℓ, and the policy induced by Algorithm 2 by $\pi^g_{\tilde{\mathcal{H}}_\ell}$. Further let $\tilde{p}_{\min,\ell} = \min_{h\in\tilde{\mathcal{H}}_\ell} \mathbb{P}[h]/1-\eta$. By definition, all *feasible* policies that satisfy the condition of the DRD problem (Problem (1)) achieve the same VoI (at different costs). Hence, Theorem 1 implies that with probability

[5]For simplicity, we assume that the prior $\mathbb{P}[Y]$ over root-causes is given. In principle, we can drop this assumption, and instead assume a prior over the parameters of $\mathbb{P}[Y]$ so that we can sample from it, similarly as how we sample $\boldsymbol{\theta} \sim B(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

at least $1-\eta$, running Algorithm 2 at session ℓ achieves the same VoI with OPT_ℓ, with at most $(2\ln(1/\tilde{p}_{\min})+1)$ times of the optimal cost.

In principle, we want to design an adaptive policy for each session that is competitive with the *hindsight-optimal* policy that knows the true distribution. We define the expected *regret* of a policy $\pi^g_{\tilde{\mathcal{H}}_\ell}$ at session ℓ with respect to the *true* distribution as

$$\tilde{\Delta}_\ell = \mathrm{VoI}^*(\pi^g_{\tilde{\mathcal{H}}_\ell}) - \mathrm{VoI}^*(\pi^*),$$

where $\mathrm{VoI}^*(\pi) = \mathbb{E}_h[\max_{d\in\mathcal{D}} \mathbb{E}_y[u(y,d) \mid \mathcal{S}(\pi,h)]]$ denotes the expected utility of policy π w.r.t. the *true* distribution to be learned, and π^* denotes the optimal policy (w.r.t. the true distribution). Suppose we are given a fixed budget τ for performing tests in each session. Define the expected regret incurred by running policies $\{\pi_1,\ldots,\pi_k\}$ over k sessions as $\mathrm{Regret}(k,\tau) = \sum_{i=1}^k \tilde{\Delta}_i$. We establish the following bound on the expected regret of running Algorithm 3:

Theorem 2. *Fix $\eta \in (0,1)$. Let $\tau = (2\ln(1/\delta)+1)\, c^{wc}_{\mathsf{OPT}}$, where $\delta = \min_t \tilde{p}_{\min,\ell}$ denotes the minimal probability of any hypothesis in the sampled distributions, and c^{wc}_{OPT} denotes the worst-case cost of the optimal algorithm over any of the k sessions. Then, Algorithm 3 achieves expected regret*

$$\mathbb{E}[\mathrm{Regret}(k,\tau)] = O\left(\tau S\sqrt{nk\tau\log(Snk\tau)} + kn\right),$$

where S is the total number of possible realizations of τ tests, and n is the number of tests.

To prove Theorem 2, we view the Optimal VoI problem as optimizing a (finite horizon) Partially Observable Markov Decision Process (POMDP) over repeated episodes of a fixed horizon τ. The parameter S in the regret bound corresponds to the number of (reachable) belief states of the POMDP. Once we have established this connection, we can interpret the online learning problem as a reinforcement learning problem via posterior sampling, in a similar way to Osband et al. [24]. Notice that a conservative bound on S in $2^{\binom{n}{\tau}}$, which is doubly-exponential in the horizon. However, in practice, the number of reachable belief states is limited by the structure of the problem (e.g., configuration of the CPTs), and hence could be far smaller. In any case, Theorem 2 implies that the expected regret of Algorithm 3 in the limit (as $k\to\infty$) is bounded by η/τ.

5 EXPERIMENTAL RESULTS

Data set and experimental setup. We carry out our experiments on a *real-world* troubleshooting platform.

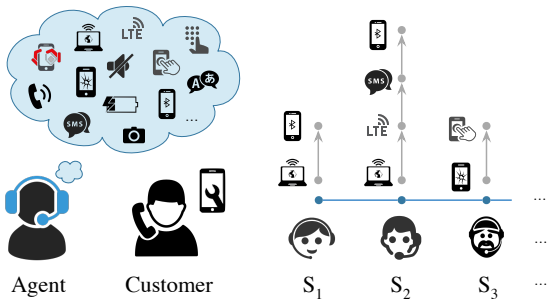
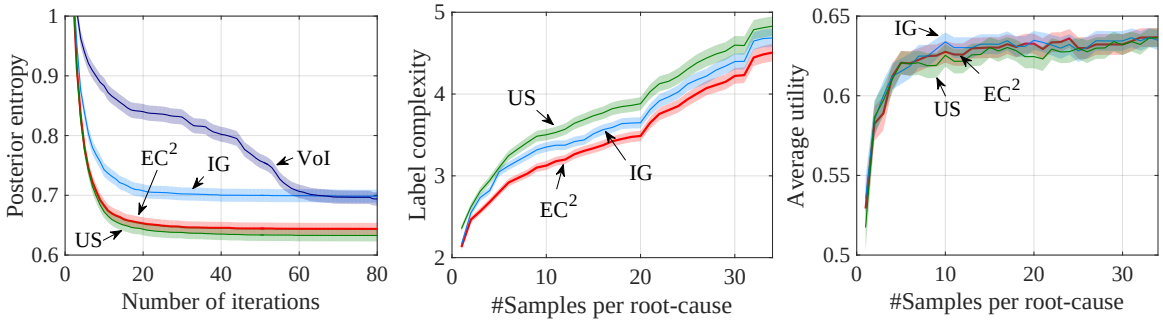


Figure 3: Online troubleshooting: customers reach a call center for diagnosis of their devices. The troubleshooting virtual agent resolves their issues by sequentially asking each customer a series of diagnosis questions.

Our data is collected from contact center agents and knowledge workers who solve complex troubleshooting problems for mobile devices (see Fig. 3). These training data involve around 1100 root-causes (the possible values of the hidden state Y) and 950 tests (questions on symptoms customers may encounter) with binary outcomes. From the training data, we derived a distribution over $[X_1,\ldots,X_n]$ and Y as $\mathbb{P}[x_1,\ldots,x_n,y] = \mathbb{P}_{\mathcal{Y}}[y]\prod_{i=1}^n \mathbb{P}[x_i \mid y]$, where $\mathbb{P}_{\mathcal{Y}}[y]$ is the prior distribution over the root-causes which we assume to be uniform.

We simulated over 10,000 test scenarios (10 scenarios for each y), where a customer enters the system with an initial symptom x_{t_0} (i.e. a test outcome) with probability $\mathbb{P}[x_{t_0} \mid y]$. Each scenario corresponds a to a root-cause y and an underlying hypothesis h. The number of decisions is the number of root-causes (which correspond to making a diagnosis), plus one extra decision of "give-up". Intuitively, if two root-causes result in the same symptoms, then the virtual agent cannot decide which one is the true root-cause, and therefore will forward such case to a human agent, corresponding to the "give-up" decision. In practice, introducing such "give-up" decision guarantees that there are no overlaps between decision regions. The utility function $u(d,y)$ corresponds to the cost of mis-prediction (either by mis-predicting a root-cause, or simply "give-up"), which is specified by the business *domain expert* as: $u(d,y) =$ (1) 0 if d^* is "give-up"; (2) 1 if $d^* = y$; and (3) -19 if $d^* \in \mathcal{Y} \wedge d^* \neq y$. In this way, the "give-up" decision is optimal when the posterior distribution over Y given *all* test outcomes has no "peak" value higher than 95%.

Information gathering with full knowledge of $\boldsymbol{\theta}$. In our experiments we set the coverage parameter $\eta = 0.02$. In addition to EC^2, we also run Algorithm 2 with several other different subroutines (by replacing EC^2 at line 6 of Algorithm 2): myopic Value of Information (VOI), Information Gain (IG), and Uncertainty Sampling (US).



(a) Budget VS. Conditional Entropy (b) Cost VS. # Samples (c) Utility VS. # Samples

Figure 4: Experimental results by running Algorithm 2 with different subroutines under the *offline* setting.

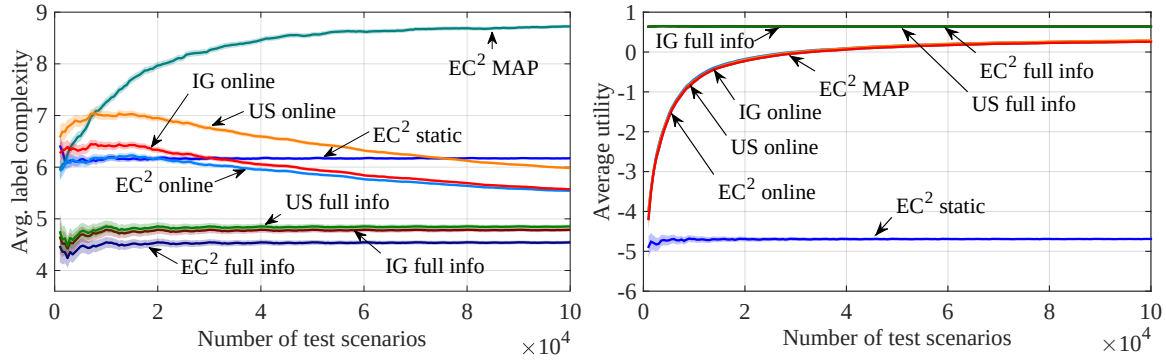


Figure 5: Experimental results by running Algorithm 3 with different subroutines under the *online* setting.

The myopic VoI criterion greedily picks the test leading to the highest expected utility[6]; the IG (resp., US) criterion picks tests that greedily maximizing the reduction of entropy over the decision regions (resp., hypotheses).

Fig. 4a shows the expected entropy on Y while increasing the test budget. Clearly, myopic VoI performs comparably worse than others. In Fig. 4b and Fig. 4c, we report the average label complexity (i.e., the expected number of tests required to solve a case), and the average utility of making decisions while varying the maximal number of samples allowed for each root-cause[7]. We see that as we generate hypotheses more extensively, all algorithms require more tests in order to make a decision; on the other hand, the quality of decisions also increases with more samples. This behavior is reasonable, since having too few samples excludes a large amount of good candidates which in turn leads to poor utility. Moreover, there is a $\sim$16% improvement in the average query complexity when using EC^2. This shows clear advantage of using submodular surrogates for this kind of sequential problem: EC^2 by construction is "less myopic".

[6]Myopic VoI does *not* require enumeration of hypotheses.

[7]The EC^2, IG and US scores of a test can be computed efficiently in *linear* time w.r.t. the number of hypotheses.

Online sequential information gathering We evaluate our online learning framework (Algorithm 3) over 100,000 simulated test scenarios. For the initial distribution over the CPT parameters, we set Beta priors $B(\boldsymbol{\alpha}, \boldsymbol{\beta})$ on the CPT, where α_{ij}/β_{ij} are set to be roughly proportional to the ratio between the number of positive and negative symptoms (i.e., $(x_i = 1, y_j)$ and $(x_i = 0, y_j)$ pairs) in the training set. Then we inject noise into these estimates, by flipping the values of α_{ij} and β_{ij} with some small probability. In our experiments, we assume a uniform prior distribution over root-causes.

Fig. 5 demonstrates the behavior of the three algorithms. In our experiments, we set a maximal budget for each session/test scenario, and keep running the policy till it identifies a decision region or exceeds the budget. At the end of session ℓ, we report the average label complexity (i.e., the accumulated mean of the number of questions asked over ℓ sessions) and the average utility (i.e., the accumulated means of the utility over ℓ sessions). In "EC^2 MAP", we update the CPT with their MAP estimators. We observe that its cost is much higher as it does not encourage exploration as much as the other online methods. In "EC^2 static", we run EC^2 only based on the initial sample of CPT without any updates. In the "full

info" versions of the algorithms, the ground truth CPT is used, while in the "online" versions, we use Algorithm 3 to update the CPT's. The Y axis shows the accumulated means of different evaluation criteria. The two notable observations are in order: First, by integrating the two levels of sampling strategy, the average utility (a.k.a reward) of all algorithms approach the optimal utility over time. Second, the EC^2 variants consistently outperform the alternatives in terms of query complexity, which is consistent with the results in the offline setting.

6 RELATED WORK

Bayesian experimental design for troubleshooting. There has been substantial research in diagnosis and troubleshooting using graphical models in the last two decades, in particular using Bayesian networks. One research question is how to perform efficient inference. For example, Nielsen et al. [23] focuses on efficient belief updating in Bayesian networks, in particular in the context of troubleshooting; Ricks and Mengshoel [26] consider the inference problem with more complex Bayesian network structures for multi-fault diagnosis. In contrast, we focus on an orthogonal direction, in the sense that we assume one can efficiently compute the posterior distribution over Y (i.e., the inference itself is tractable), and aim to minimize the cost of the troubleshooting policy.

Optimal VoI. Many greedy heuristics have been proposed for optimizing VoI; for instance, Breese and Heckerman [3] proposes a myopic policy for the single fault troubleshooting problem, which chooses the action/test that minimizes the expected cost of fixing the device in question. Their objective function can be viewed as a particular form of the myopic VoI. Unfortunately, unlike the greedy algorithms based on submodular surrogates, these algorithms can perform arbitrarily badly [13, 21].

Sampling-based Bayesian inference. In [13], the authors suggest to use a rejection sampling approach to estimate the original distribution over hypotheses for the DRD problem. Another work, which is similar to our work in spirit, suggests to generate hypotheses in descending order of prior likelihood [22] for approximate inference in hybrid Bayesian networks. In comparison, our sampling scheme is based on the specific structure of the underlying model, which potentially offers increased efficiency and better approximation guarantees.

Sequential decision making as POMDP. Value of information is known as a reward function in Partially Observable Markov Decision Process [17, 30], where each belief state represents a set of tests and their outcomes. Unfortunately, this gives us an exponentially large state space. The idea of using samples to speed up planning has been explored, e.g., POMCP [25] which is based on Monte Carlo tree search that samples from states and action histories, and DESPOT [31] which samples scenarios for evaluation of all policies at each iteration. These approaches rely on a finite set of policy to be evaluated. As the number of tests increases, they will require more samples. Consequently, they are limited to short planning horizons and small state and action spaces.

Online learning. Many different schemes have been investigated in online learning [2] and online decision making [11, 18]. We look into a different setting, where we integrate sequential information gathering into an *online* learning framework wherein we learn the decision model over time. For this purpose, we lift our framework into a higher level sampling procedure, whose goal is to sample appropriately the decision model parameters. Several recent empirical simulations [7, 14, 28] and theoretical studies [1, 4, 19] have demonstrated the effectiveness of Thompson sampling in different settings. However, different from our framework, the classical usage of Thompson sampling [32] suggests to choose an action according to the probability of being optimal, i.e. the action which maximizes the reward in expectation.

7 CONCLUSION

We pursue practical and efficient approaches for the challenging problem of optimal VoI: (1) For a given prior over hypothesis space, we propose a novel hypothesis enumeration strategy for efficient optimization of VoI, and show that it compensates the inefficiency of the popular "submodular surrogate"-based greedy algorithms, while still enjoying provable theoretical guarantees; (2) when the prior is unknown, we propose an efficient, principled online learning framework with low expected regret in the long run. In addition, we demonstrate promising empirical performance of our framework on a real-world troubleshooting platform. We believe that our work is an important step towards practical and robust adaptive information acquisition systems.

Acknowledgments

We would like to thank the anonymous reviewers and Christopher Dance for their helpful comments. This work was done in part while Andreas Krause was visiting the Simons Institute for the Theory of Computing. This work was supported in part by ERC StG 307036, a Microsoft Research Faculty Fellowship, an SNSF Early Postdoc.Mobility Fellowship and a Google European Doctoral Fellowship.

References

[1] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*, 2012.

[2] Avrim Blum. On-line algorithms in machine learning. In *Developments from a June 1996 Seminar on Online Algorithms: The State of the Art*, 1998.

[3] John S. Breese and David Heckerman. Decision-theoretic troubleshooting: A framework for repair and experiment. In *UAI*, 1996.

[4] Sebastien Bubeck and Che-Yu Liu. Prior-free and prior-dependent regret bounds for thompson sampling. In *NIPS*, 2013.

[5] V. T. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi, and M. Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. In *SIGMOD/PODS*, 2007.

[6] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 1995.

[7] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *NIPS*, 2011.

[8] Y. Chen, S. Javdani, A. Karbasi, J. Andrew Bagnell, S. Srinivasa, and A. Krause. Submodular surrogates for value of information. In *AAAI*, 2015.

[9] Y. Chen, S. H. Hassani, and A. Krause. Near-optimal bayesian active learning with correlated and noisy tests. In *AISTATS*, 2017.

[10] S. Dasgupta. Analysis of a greedy active learning strategy. In *NIPS*, 2004.

[11] Dean P. Foster and Rakesh Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29:735, 1999.

[12] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *JAIR*, 2011.

[13] D. Golovin, A. Krause, and D. Ray. Near-optimal bayesian active learning with noisy observations. In *NIPS*, 2010.

[14] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML*, 2010.

[15] R. A. Howard. Information value theory. In *IEEE T. Syst. Sci. Cyb.*, 1966.

[16] S. Javdani, Y. Chen, A. Karbasi, A. Krause, D. Bagnell, and S. Srinivasa. Near-optimal bayesian active learning for decision making. In *AISTATS*, 2014.

[17] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[18] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. In *J. Comput. Syst. Sci*, pages 26–40, 2003.

[19] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *ALT*, 2012.

[20] S. Rao Kosaraju, Teresa M. Przytycka, and Ryan S. Borgstrom. On an optimal split tree problem. In *WADS*, 1999.

[21] A. Krause and C. Guestrin. Optimal value of information in graphical models. *JAIR*, 2009.

[22] Uri Lerner and Ronald Parr. Inference in hybrid networks: Theoretical limits and practical algorithms. In *UAI*, 2001.

[23] T. Nielsen, P.-H. Wuillemin, F. Jensen, and U. Kjaerulff. Using robdds for inference in bayesian networks with troubleshooting as an example. In *UAI*, 2000.

[24] Ian Osband, Dan Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *NIPS*, 2013.

[25] Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Anytime point-based approximations for large pomdps. *JAIR*, 27(1):335–380, 2006.

[26] Brian Ricks and Ole J. Mengshoel. Diagnosis for uncertain, dynamic and hybrid domains using bayesian networks and arithmetic circuits. *International Journal of Approximate Reasoning*, 2014.

[27] M. C. Runge, S. J. Converse, and J. E. Lyons. Which uncertainty? using expert elicitation and expected value of information to design an adaptive program. *Biological Conservation*, 2011.

[28] Steven L. Scott. A modern bayesian look at the multi-armed bandit. *Appl. Stoch. Model. Bus. Ind.*, 26(6):639–658, 2010.

[29] B. Settles. *Active Learning*. Morgan & Claypool, 2012.

[30] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[31] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *NIPS*, 2013.

[32] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.