Experiential Preference Elicitation for Autonomous Heating and Cooling Systems

Andrew Perrault*
University of Southern California
Los Angeles, California

Craig Boutilier
Google
Mountain View, California

ABSTRACT

AI systems that act on behalf of users require knowledge of user preferences, which can be acquired by *preference elicitation.* In many settings, users can respond more easily and accurately to preference queries reflecting their *current, or recently experienced, context* (e.g., state of the environment), than to those reflecting contexts further removed. We develop and study a formal model of *experiential elicitation (EE)* in which query costs and response noise are state-dependent. EE settings tightly couple the problems of control and elicitation. We provide some analysis of this abstract model, and illustrate its applicability in household heating/cooling management. We propose the use of *relative value queries*, asking the user to compare the immediate utility of two states, whose difficulty is related to the degree and recency of a user's experience with those states. We develop a Gaussian process-based approach for modeling user preferences in dynamic EE domains and show that it accrues higher reward than several natural baselines.

KEYWORDS

Single and multi-agent planning and scheduling; Multi-user/multi-virtual-agent interaction

ACM Reference Format:
Andrew Perrault and Craig Boutilier. 2019. Experiential Preference Elicitation for Autonomous Heating and Cooling Systems. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, IFAAMAS, 9 pages.

1 INTRODUCTION

AI systems or agents that interact with users generally require information about user preferences to act effectively on their behalf. Such preferences can be obtained in a variety of ways, which we divide into two broad categories. The first is using *revealed preference data*, in which a user's past decisions in the target (or a related) setting provide a (partial) view of their utility function [2, 19, 28]. The second is through *preference elicitation (PE)*, i.e., directly asking the user about their preferences for possible decisions or outcomes, using queries of various types [4, 6, 8]. Each approach (or some combination) may be better-suited to particular settings. Revealed preference is ineffective when costs of acquiring user data are prohibitive (e.g., due to privacy rules, or scarcity) or when relevant data accrues slowly or not at all. Revealed preference may also fail when the introduction of a user-representing agent *changes* the user's preferences. For example, if electricity prices drop at night while a user sleeps, she may be unable to exploit the decrease without an agent. PE avoids this by *querying* the user in advance about how she would act in hypothetical situations.

PE has its own limitations. Asking a user to introspect and analyze her preferences for specific outcomes—usually at some remove from the scenario where the data will be used—incurs *cognitive costs* and risks inaccurate responses. For instance, an apartment selection agent may ask a user to compare two "hypothetical" apartments without the user being able to view them [6]. In reality, however, people rely heavily on "experiential" information when making real estate decisions. Prospective home buyers visit an average of 10 homes over 5 months and 80% of prospective buyers know whether a home is right for them as soon as they step inside [21]. Their reliance on "experiential" information to make decisions can be explained by *dual process theory* [16], which includes a primarily subconscious system used to make decisions on issues with which one has extensive experience. For example, we can quickly evaluate the comfort of a prospective living space by visiting it in person. Performing the same evaluation from a description is a difficult, unfamiliar task.

In this work, we develop a model of elicitation in *sequential* decision-making settings with relevant experiential information, where the cost and accuracy of a user's response to a preference query depend on the state of the system. Specifically, if a user is asked to assess (e.g., compare) outcomes, both cost and response noise decrease with how "closely" the user has *experienced* these outcomes, allowing different forms of distance (e.g., state similarity, recency of experience) to play a role. We dub this approach *experiential elicitation (EE)*, following Hui and Boutilier [14]. The EE setting introduces new challenges: because query cost and noise depend on the current state, an agent's policies for both system control and elicitation are tightly coupled. Indeed, there is an explicit trade-off between exploitation (optimal control given the current preference information) and exploration (having the user encounter new circumstances that may allow preference queries that can improve control).

We motivate our methods by considering a smart home agent that controls the heating, ventilation and cooling (HVAC) system, changing settings in response to variable electricity prices and a user's complex temperature preferences. The increasing presence of renewable sources of power generation has created highly variable pricing, and users cannot realistically adapt their behavior to price variations in real time (due to attentional costs, inability to forecast price and temperature changes, etc.). For an autonomous agent to adapt HVAC on a user's behalf, considerable preference information about small changes in comfort levels is needed. PE queries can be difficult to answer unless a user is actually experiencing (or has

*This research was performed while the author was at University of Toronto.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

recently experienced) the conditions in question (e.g., temperature, humidity, price). As such, this a natural domain for EE.

The main contributions of this paper are as follows:

(1) We introduce and motivate the EE approach for a variety of AI systems that interact with users over time.
(2) We provide a theoretical analysis connecting optimal EE to other well-known problems in AI.
(3) We study the interplay (and synergies) of *relative value queries (RVQs)*, which are natural in HVAC settings, with a preference prediction model based on *Gaussian processes (GPs)*.
(4) We develop a system for EE in a smart-home HVAC setting using RVQs and GPs, and analyze it empirically using a combination of real and synthetic data, showing that it outperforms natural baselines.

The paper is organized as follows. We review Markov decision processes, GPs, and related work in PE. Then we provide a formal model of EE and relate it to other problems in AI. We introduce RVQs and our GP-based model for EE. Next we outline a natural cost and noise model for RVQs, and we conclude with experimental results.

2 BACKGROUND

Markov Decision Processes (MDPs): Our elicitation methods are engaged by an agent acting in a stochastic control environment modeled as an *MDP* [23]. The problem of HVAC control naturally lends itself to being solved as an MDP. At each time step, the agent inputs a control action, heating or cooling the house. The interior temperature in the next time step depends on the interior and exterior temperatures in the previous step and the action that was taken. We assume a fully observable MDP $\mathcal{M} = \{S, A, \{P_{sa}\}, \gamma, \beta, r\}$ with finite state set S, action set A, transition models P_{sa}, discount factor γ, initial state distribution β and reward function $r(s, a)$. We consider infinite-horizon models. A (deterministic) *policy*, i.e., a mapping from each state to an action, $\pi : S \rightarrow A$ has *value* V^{π}, given by: $V^{\pi} = \sum_{s_1 \in S} \beta(s_1) \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^{i-1} r(s_i, \pi(s_i)) | \pi\right]$, where expectation is taken over the distribution of state sequences induced by π. An *optimal policy* π^* maximizes expected value, with $\pi^* \in \operatorname{argmax}_{\pi} V^{\pi}$, and can be computed by a variety of means [23].

Preference Elicitation (PE): PE refers to the process by which an agent, when making a decision on behalf of a user, learns the user's preferences for outcomes of that decision by querying the user [9, 29]. In this work, we assume that the agent starts with some prior distribution over the user's utility, from which the true utility function is drawn. To gain knowledge of the user's utility, the agent *queries* the user, who responds (perhaps with some noise) in a fashion that depends on her utility. The type of queries allowed is a modeling choice that depends on the user's capabilities and the structure of the problem. The cognitive and communication costs of queries also play a role in PE; otherwise an agent could potentially ask the user to reveal her entire utility function. After asking some number of queries, the agent makes a decision and is "rewarded" with the expected value of that decision w.r.t. to its posterior beliefs about the true utility function given query responses received. We provide a formal description of PE (including query types and strategies) when we describe experiential elicitation below.

Gaussian Processes (GPs): GPs [24] are universal function approximators that model the points of a function as a multivariate Gaussian distributed with known covariance given by applying the *kernel function* to their inputs. In addition to estimating the function value at any point, GPs quantify the uncertainty in that estimate. GPs have been used previously in PE because they address several of the main desiderata of preference models: a) they can capture any utility function; b) they model uncertainty in a principled manner; c) they facilitate query selection to improve the user's objective; and d) they allow for the incorporation of prior knowledge [3]. A GP is represented as $\{k, \boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}, \sigma^2\}$ with kernel function $k(\cdot, \cdot)$, training data $(\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}})$ and Gaussian observation noise σ^2.[1] Typical kernel functions include the squared exponential $\exp[\frac{(x_0 - x_1)^2}{2\ell^2}]$, where the *lengthscale* parameter ℓ governs the distance over which observations have a "significant" effect on estimates.

Calculating a GP's predicted mean $\boldsymbol{\mu}_{\text{test}}$ and covariance $\bar{\Sigma}$ at any number of test points $\boldsymbol{x}_{\text{test}}$ can be done by applying the standard conditional expectation and covariance expressions for multivariate Gaussians and using the fact that the sum of Gaussians is Gaussian (for observation noise). Let $\boldsymbol{K}$ denote the covariance matrix that results from stacking the test points and the training points:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{\text{test, test}} & \boldsymbol{K}_{\text{test, train}} \\ \boldsymbol{K}_{\text{train, test}} & \boldsymbol{K}_{\text{train, train}} \end{bmatrix} \tag{1}$$

where $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Then,

$$\boldsymbol{\mu}_{\text{test}} = \boldsymbol{K}_{\text{test, train}} (\boldsymbol{K}_{\text{train, train}} + \sigma^2 I)^{-1} \boldsymbol{y}_{\text{train}} \tag{2}$$

$$\bar{\Sigma} = -\boldsymbol{K}_{\text{test, train}} (\boldsymbol{K}_{\text{train, train}} + \sigma^2 I)^{-1} \boldsymbol{K}_{\text{train, test}} + \boldsymbol{K}_{\text{test, test}} + \sigma^2 I. \tag{3}$$

Matrix inversions make both operations (2, 3) scale cubically with the number of training points. Since our work aims to minimize this number (i.e., user queries asked), this is not a practical obstacle. Since test points correspond to states of the MDP, performance of the conditional expectation operation is more critical—it is linear in data set size.

While we present GPs with uniform observation noise, we make use of the fact that the matrix $\sigma^2 I$ can be swapped with an arbitrary covariance matrix, e.g., to allow for non-identically distributed Gaussian noise.

Related Work: Regan and Boutilier [25, 26], in their work on PE in MDPs, assume that elicitation and control are separable. They query the user until enough reward information has been acquired to solve the MDP (approximately) optimally. In our work, the ability of the user to effectively assess reward depends on the current system state, inducing a tight coupling between elicitation and control. PE in sequential decision making has also been studied from the perspective of "active" inverse reinforcement learning [15, 20], but without an experiential component.

Shann and Seuken [30, 31] study PE and optimal control in an MDP-based HVAC model. Their work differs from ours in three main ways. First, the cost and accuracy of their queries are not modeled to allow context dependence. Second, they assume a highly

[1] We assume a prior mean of zero for simplicity and w.l.o.g.

restrictive query structure: the user is queried once per day regardless of whether the query is valuable to the algorithm or not, and it is impossible to query about any context except the current one, i.e., the model can't learn about the user's utility function when she can't be queried directly (e.g., is asleep). Third, they assume user utility has a specific functional form, whereas we allow learning of arbitrary utility functions, a harder task.

Truong et al. [34] and Le et al. [17] study PE in the context of scheduling. Both works aim to maximize user utility and minimize query cost. The key difference between their approaches and ours is their models are not "experiential:" the cost of a query depends only on the query and the *query* history. Neither the cost nor accuracy of a query response depends on the current *state* or state history.

GPs were first used for preference learning by Chu and Ghahramani [10], who study the problem of learning to rank a set of alternatives given the results of pairwise preference queries without elicitation. Bonilla et al. [3] extend this model to support multi-user collaborative filtering, and perform elicitation using expected value of information estimates of each query, though they do not consider experiential information.

PE with a GP-based uncertainty model resembles GP-based approaches to Bayesian optimization [18, 33]. The key difference is the experiential component of our approach. In addition, queries have costs in our model (we maximize the payoff minus the query cost, rather than minimizing the number of queries to reach some target solution quality), and the function we maximize at each stage differs due to its dependence on the current state.

3 A MODEL OF EXPERIENTIAL ELICITATION

In this section, we describe an abstract model for *experiential elicitation (EE)* and draw theoretical connections to other sequential decision models.

An *EE problem instance* is $\{\mathcal{M}, R, Q, N = \{N_q\}, D = \{D_q\}, C\}$, where $\mathcal{M}$ is a fully observable MDP. Unlike a standard MDP, the true reward function r is not (initially) known to the agent. R is the *reward uncertainty model* capturing the agent's prior over possible rewards. We take a probabilistic perspective—R is a distribution over reward functions [35]. We sometimes abuse notation and use R to refer to the support of the distribution (i.e., set of feasible reward functions). The remaining parameters specify the process available to the agent to elicit user's reward function. Q is the set of available *queries* that can be asked, while N associates a set of possible user *responses* N_q with each $q \in Q$. The *response function* D specifies a response distribution for each query. It maps from $R \times S^*$ to a distribution N_q for each $q \in Q$, defining a distribution over user responses given any possible true reward function r and history of past states; this reflects possible noise in user responses. The *query cost function* $C : Q \times S^* \to \boldsymbol{R}^+$ specifies the cost of asking a query given a state history; this reflects the cognitive, communication, interruption, delay or other costs imposed on the user by a query.

An agent acting in an EE instance knows all problem elements except r. In other words, it doesn't know the true reward r, but only the space and distribution R of possible rewards. This reflects the fact that an agent has incomplete and imprecise information about a user's true preferences. The agent can at each stage choose to ask queries or take actions, specifically:

(1) An initial state s, drawn from β, is revealed (the MDP state is fully observable).
(2) Repeat infinitely:
   (a) The agent asks the user zero or more queries. The user responds to each according to the query response function (w.r.t. r and the state history).
   (b) The agent executes some $a \in A$ and the state transitions according $\{P_{sa}\}$, with the new state revealed.

A *query strategy* $\{\{Q \times N\}^* \times S\}^* \to Q \cup A$ maps from a query, response and state history to either the next query to ask or control action to take. The goal of the agent is to maximize expected total discounted reward w.r.t. the user's (unknown) reward function r less discounted query costs.[2]

It is not hard to show that elicitation and control are coupled and should be integrated. Indeed, in the worst case, the optimal policy *without querying* can be arbitrarily worse than a policy that integrates elicitation:

Example 1. Let $S = \{s_0, \ldots, s_n, s_{elicit}\}$, with the initial state drawn uniformly at random. Let R reflect that exactly one of $\{s_0, \ldots, s_n\}$ gives the user reward x (others have reward zero), with each equally likely. The agent can deterministically move from any state to any other at zero cost, except that a transition to s_{elicit} costs y. A single (zero-cost, noise-free) query reveals which s_i is rewarding, but can only be asked at s_{elicit}.

The optimal query-free policy randomly traverses the s_i, yielding value $\frac{x}{n(1-\gamma)}$.[3] If $\frac{x\gamma^2(n-1)}{n(1-\gamma)} - \gamma(y + \frac{x}{n}) \geq 0$, the optimal strategy *with queries* moves first to s_{elicit}, issues the query—note that a "plan" is needed even be able to query—and then moves to (and stays at) the rewarding state, yielding value $\frac{x}{n} - y\gamma + \frac{x\gamma^2}{1-\gamma}$.

Let $n, x, y \to \infty$, with n doing so asymptotically faster than x, and x faster than y. The value of the query-free policy approaches 0 while the query strategy approaches ∞.

In the remainder of this section, we draw connections between the EE model and several related decision-making models. Exploiting these connections is difficult in the domain we consider, but they may be valuable in other settings and are of theoretical interest. We first note that an EE instance can be formulated as a *partially-observable MDP (POMDP)* [32]. Full proofs are in the supplemental material.[4]

Theorem 1. *Any EE instance can be formulated as a POMDP.*

Proof Sketch. The fundamental idea of the proof is to embed the set of possible rewards in the POMDP state space. This transformation is related to Poupart et al.'s [22] method of solving Bayesian reinforcement learning problems by formulating them as POMDPs. State transitions act on S, but never change the reward embedded

[2]We make two assumptions for ease of exposition. First, MDP state transitions evolve more slowly than any "reasonable" number of queries. If this were not the case, we could model exogenous state transitions that occur when queries are asked or introduce time-based discounting to account for query "delays." Second, (non-query) actions provide no reward information. The extension to actions with information content is straightforward.

[3]Note that the user, not the agent, experiences the reward, so without user feedback, the agent cannot observe which state is rewarding even as it occupies it.

[4]Available at http://www.cs.toronto.edu/~perrault/EE-supplement.pdf

in the state; thus, for any distinct $r, r' \in R$, the corresponding embedded state sets $S^{(r)}$ and $S^{(r')}$ do not communicate. The reward uncertainty distribution may be discrete or continuous (necessitating a POMDP with infinite states in the latter case). The action space A is augmented by the set of queries, so the agent can ask queries or take (original) actions; queries cause no state transition. The observation function for (original) actions reflects full observability of the (original) state space (observations are the states themselves), while for queries, the observation function captures the distribution over responses.

□

Note that the POMDP belief state for an EE problem will represent the agent's posterior over R, reflecting information captured about a user's preferences by queries and responses. POMDPs have been used to model elicitation problems in the past [4, 13]. These formulations differ from ours because they require only one state for each potential reward function. Unfortunately, exactly solving even simple POMDPs is often intractable due to their doubly-exponential complexity.

There is a direct connection between EE and reinforcement learning (RL) as well.

Observation 1. *Consider an RL problem $\langle \mathcal{M} \rangle$ that consists of an MDP $\mathcal{M}$ which is known to the agent except for the reward function, and whenever the agent transitions into a state, it receives the reward for that state. This problem can be reduced to EE and the reduction requires increasing the number of states by a factor of $O(|S| \times |A|)$.*

Proof. Create an EE that retains the model details (states, actions, transitions, rewards, discount). We let reward uncertainty R be an uninformative prior. For each state s, we allow *value queries* for any state-action pair (s, a), which asks a "user" (representing the environment) for the reward for that pair, and the response function represents the RL (stochastic) reward for that pair. Query cost is zero if asking about the action just taken at the previous state, and infinite otherwise. To encode this query cost function, the EE state must include the previous state visited and action taken, which causes a state space blowup of $|S| \times |A|$.

In this EE instance, the only "available" query at a state is "free," so it is optimal to always ask it, giving an EE agent the same information as an RL agent. □

Lastly, suppose we ignore an EE agent's query strategy or, equivalently, make no queries available. The optimal control policy under reward uncertainty for a risk-neutral agent can be solved as an MDP.

Observation 2. *Given an MDP with uncertain reward R, its (risk-neutral) optimal policy is that of an MDP where each state-action reward is its expected reward under R. (This holds even if rewards are correlated under R.)*

This follows from a simple argument using linearity of expectation. We exploit this observation when defining myopic expected value of information in the next section.

4 EE WITH RELATIVE VALUE QUERIES

A *relative value query (RVQ)* asks the user to articulate the difference in value or utility between states. An RVQ $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ comprises two points in a d-dimensional state space. A response y is the user's estimate of the difference $r(\boldsymbol{x}_0) - r(\boldsymbol{x}_1)$ in their immediate rewards. We study RVQs because they naturally capture the way that users make decisions in certain domains. For example, in the HVAC setting an RVQ captures a query like "In the current conditions, what electricity cost savings would be required so that raising the temperature by one degree for one hour would be acceptable?" When a user sets their HVAC system in a given context, they are "reasoning" over the set of RVQs. In this section, we examine EE systems that use RVQs.

It is instructive to compare RVQs to several other query types [29] in the HVAC domain. *Value queries* ask the user directly for her utility for a state, e.g., "How much is it worth to hold temperature and humidity at (t, h) for the next two hours?" This is a challenging query for users in the home HVAC setting. *Demand queries* ask the user for her most preferred outcome from a set (e.g., "Which of these three configurations do you prefer?" or "Which of these four actions is best under the current conditions?" Asking a user for her preferred action requires the user to reason about action costs and benefits, exactly what an intelligent agent is supposed to *do for the user*, and also has some ambiguity w.r.t. its semantics (e.g., immediate vs. planning value). RVQs, by contrast, separate the state reward from the cost of reaching that state and do not require the user to reason about expectations. Asking for, say, preferred temperature and humidity in isolation makes it difficult to integrate cost considerations.[5]

A first key observation is that the MDP embedded in an EE instance can be solved optimally using only RVQs. Having only difference information for all state pairs means that state rewards are known only up to an additive factor. However, by the equivalence of utility functions (in this case, the MDP value function) under positive affine transformation, it immediately follows that adding a constant to the reward function does not change the optimal policy.

Observation 3. *An MDP can be solved optimally given only information about the differences in rewards between a collection of state-action pairs.*

To exploit RVQs effectively, we need a learning method that, given a set of responses to RVQs, can estimate the rewards (and uncertainty) of all states. GPs can be adapted to this purpose while maintaining the critical properties that drive their usefulness for PE, in particular, the ability to estimate the expected value of information of a query.

We show that the GP posterior update has a closed form in the case of RVQs. Since an RVQ represents the difference of two Gaussian random variables (RVs) with known covariance, the affine property of multivariate Gaussian RVs ensures they are closed under linear transformations.

Observation 4. *Let $\mathcal{X}$ be an RV distributed as $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, $\boldsymbol{c}$ be a constant M-dimensional vector, and $\boldsymbol{B}$ a constant $M \times N$ matrix. Then $\boldsymbol{c} + \boldsymbol{B}\mathcal{X} \sim \mathcal{N}(\boldsymbol{c} + \boldsymbol{B}\boldsymbol{\mu}, \boldsymbol{B}\Sigma\boldsymbol{B}^T)$.*

[5]Binary comparisons [5] are even easier than RVQs to answer, but they require more sophisticated inference techniques (e.g., variational or MCMC).

Suppose our training data consists of RVQ responses.[6] Our n training points have form $(\boldsymbol{x}_0, \boldsymbol{x}_1, y)$, where $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ is the RVQ (two states) and y its (noisy) response, i.e., estimation of reward difference. Let $\boldsymbol{X}_0$ (resp., $\boldsymbol{X}_1$) be the matrix resulting from stacking the $\boldsymbol{x}_0$ (resp., $\boldsymbol{x}_1$) vectors, and let $\boldsymbol{X}_{\text{test}}$ be the set of m test points, arranged as a $m \times d$ matrix. To compute the posterior of the GP at $\boldsymbol{X}_{\text{test}}$ (states at which we wish to predict reward), we construct an affine transformation matrix $\boldsymbol{B}$. Let $\boldsymbol{X}$ be the data points stacked as $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_{\text{test}} \\ \boldsymbol{X}_0 \\ \boldsymbol{X}_1 \end{bmatrix}$, a matrix of size $(m + 2n) \times d$. Define

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I}_{m,m} & \boldsymbol{0}_{m,2n} \\ \boldsymbol{0}_{n,m} & \boldsymbol{B}_{\text{diff}} \end{bmatrix} \tag{4}$$

where $\boldsymbol{B}_{\text{diff}}$ is $[\boldsymbol{I}_{n,n}\ -\boldsymbol{I}_{n,n}]$, $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{0}$ is the zero matrix. ($\boldsymbol{B}_{\text{diff}}$ is $n \times 2n$ and $\boldsymbol{B}$ is $(m + n) \times (m + 2n)$.) $\boldsymbol{B}$ transforms the training rows into differences without affecting the test points.

In the absence of observations, the distribution of $\mathcal{X}$—the RV that represents the distribution over $\boldsymbol{X}$—is just the zero-mean, covariance Σ Gaussian prior, where Σ results from our chosen kernel. Applying Obs. 4 using $\boldsymbol{B}$ yields an RV that relates the training and test points: a Gaussian with covariance $\boldsymbol{B}\Sigma\boldsymbol{B}^T$ and mean $\boldsymbol{B}\boldsymbol{0} = \boldsymbol{0}$. This Gaussian has dimension $m+n$ (cf. the original $m+2n$)—we now have a *single* dimension for each training example. The conditional GP (Eqs. 2, 3) in the transformed space yields the posterior at the test points.

As in a standard GP model, we can also incorporate observation noise. Suppose that each observation y has i.i.d. Gaussian noise with variance σ^2. We model this with $\frac{\sigma^2}{2}$ noise on the underlying Gaussian, which results in observation noise of σ^2 on the difference. Non-i.i.d. noise is handled by replacing $\sigma^2 I$ with our chosen noise matrix.

Using this procedure for posterior updates, we can estimate *(myopic) EVOI* of a query using a GP, adapted to the MDP setting. The estimate is myopic because it does not account for future queries. Myopic EVOI is the amount our expected reward increases as a result of asking a query w.r.t. the current preference predictions of the GP. In our experiments, we use EVOI to determine if a query increases our discounted reward more than the cost of the query.

Suppose we want to estimate the EVOI of an RVQ q. Let R be our current reward uncertainty and $R_{(q,y)}$ be the posterior after asking q and receiving response y. Let Y be an RV representing the RVQ response according to R. Applying Obs. 2, let V^* be the optimal policy value under belief R and let $V^*_{q,y}$ be the optimal policy value under belief $R_{(q,y)}$.

Definition 1. The *(myopic) (EVOI)* of a query q in EE is $\mathbb{E}_Y[V^*_{(q,y)} - V^*]$.

In practice, we compute the expectation by sampling query responses from R and averaging the changes in policy value. We use the following procedure:

(1) Sample n query responses for q from the GP posterior.
(2) For each query response y:
   (a) Add y to the set of query responses (i.e., treat it as if it were a query that we actually asked the user).
   (b) Calculate the value of the optimal policy under the new set of query responses and save it.
   (c) Remove y from the set of query responses.
(3) Return the optimal policy value, averaged over all the responses, minus the original optimal policy value.

Random forest model: As a point of comparison, we also define a *random forest (RF)* model [7, 11] for EE with RVQs. We choose an RF because of its excellent performance with small amounts of training data. We train the RF using each training example twice, $(\boldsymbol{x}_0, \boldsymbol{x}_1, y)$ and its "reversed" RVQ as $(\boldsymbol{x}_1, \boldsymbol{x}_0, -y)$, so the model is informed of the "opposite" prediction. At test time, we predict a response to q' by averaging the model response to q' and the negative response to the reversal of q'. To evaluate reward at a set of states, we formulate each as an RVQ that compares it to the state most observed in the training data. Unlike the GP, this model is incapable of evaluating EVOI of queries because the RF does not estimate the uncertainty of its predictions.

5 QUERY RESPONSE AND COST MODEL

We now develop a model for users' responses to queries, i.e., how accurate and how costly such responses are. We are motivated by the HVAC setting, where RVQs compare states representing the internal and external temperature, humidity, time, etc. Our response and cost models should satisfy the following principles:

(1) They should align with the concept of *just noticeable difference (JND)* [12]. An RVQ comparing two states that differ only in internal temperature is more difficult to answer if the difference is small than if the difference is large. Greater "difficulty" means higher cognitive cost and more response noise.
(2) It is more difficult to compare states that differ in more ways rather than fewer; e.g., it is easier to compare states that differ only in internal temperature than states that differ in internal/external temperature/humidity.
(3) It is easier to answer queries about states that are similar to the current state, or to one that was visited (i.e., experienced) recently. For example, it is easier to compare two "summer states" in the summer than it is in the winter.
(4) The mere act of answering a query is expensive because it is disruptive. Thus, there is a lower bound on query response costs.

Query response model: We use the following query response model:

$$D(\boldsymbol{x}_{q0}, \boldsymbol{x}_{q1}) = \mathcal{N}(r(\boldsymbol{x}_{q0}) - r(\boldsymbol{x}_{q1}), \sigma_{noise}) \tag{5}$$

$$\sigma_{\text{noise}} = c_{\text{fn}} + c_{\text{dn}}(||\boldsymbol{x}_{q0} - \boldsymbol{x}_{q1}||_1 + \min_{0 \le i \le t} (1+\delta)^{t-i}(||\boldsymbol{x}_{q0} - \boldsymbol{x}_i||_1 + ||\boldsymbol{x}_{q1} - \boldsymbol{x}_i||_1)) \tag{6}$$

where $\boldsymbol{x}_{q0}$ and $\boldsymbol{x}_{q1}$ are the queried states, $\boldsymbol{x}_i$ is the sequence of past/visited states and δ is the user's discount factor.

Modern psychophysics [12] assumes that sensor noise is the cause of JND. The **f**ixed **n**oise constant $c_{\text{f\_n}}$ ensures that noise is non-zero. The second and third desiderata are satisfied by using the L_1-distances between states, which captures both the number of (state feature) differences and their magnitude, weighted by the **d**istance **n**oise constant $c_{\text{d\_n}}$. We capture the temporal component

[6]This method can be modified to accommodate any combination of training points as long as each represents the result of applying a linear transformation. This includes ordinary value queries.

(i.e., less noise when queried states are close to recently visited states) by discounting distances to previous states and using the state with least discounted distance. The literature generally embraces the notion that sensors are well-calibrated, thus, we use an unbiased response model.

Our query cost model has the same form, with one addition. We augment the model to reflect that queries near the JND are more costly by adding a term that decreases with the distance between $\boldsymbol{x}_{q0}$ and $\boldsymbol{x}_{q1}$:

$$C(\boldsymbol{x}_{q0}, \boldsymbol{x}_{q1}) = c_{\text{fc}} + c_{\text{dc}}(||\boldsymbol{x}_{q0} - \boldsymbol{x}_{q1}||_1 + \min_{0 \le i \le t} (1+\delta)^{t-i}(||\boldsymbol{x}_{q0} - \boldsymbol{x}_i||_1 + ||\boldsymbol{x}_{q1} - \boldsymbol{x}_i||_1)) + \frac{c_{\text{JND}}}{1 + e^{||\boldsymbol{x}_{q0} - \boldsymbol{x}_{q1}||_1}} \quad (7)$$

6 EXPERIMENTS

Our experiments assess: (i) how the GP-based approach to EE compares to several natural baselines; and (ii) how filtering prospective queries by EVOI affects both the quality of the learned policy and overall utility accrued. We first describe our experimental setup. Our MDP model is derived from HVAC and temperature data from Pecan Street Inc. [27]. The states of the MDP contain the time, date, exterior and interior temperature. We discretize time into hours and temperatures into 20 buckets between 5 and 45°C. There are 3.5 million states. There are 10 HVAC control actions (5 heating, 5 cooling). Control response depends on properties of the HVAC system and thermal properties of the house in question, which are selected randomly from a range of realistic values. Action costs are determined by electricity prices in Austin, TX (where the data was collected). The user's reward for state s has the form:

$$r(s) = \exp\left[-h\frac{(\text{INT}(s) - (\text{INT}^* + \text{BIAS}(\text{EXT}(s) - \text{INT}^*)))^2}{w}\right] \quad (8)$$

where INT^* is the user's most preferred internal temperature, $\text{INT}(s)$ and $\text{EXT}(s)$ are the internal and external temperatures in state s, and BIAS represents the degree to which a user's preference for internal temperature is affected by external temperature. Parameters h and w control the height (strength) and width (breadth) of a user's utility function. INT^*, w, h and BIAS are drawn from $\mathcal{U}(18, 22)$, $\mathcal{U}(0, 20)$, $\mathcal{U}(0, 3)$, and $\mathcal{U}(0, 0.4)$, respectively. All users discount exponentially at a rate of 1% per period (hour). Query response/cost models use: $c_{\text{fn}} = c_{\text{fc}} = c_{\text{dc}} = c_{\text{dn}} = 0.05$ and $c_{\text{JND}} = 2.5$.

We run two sets of experiments. In the first, we randomly generate a user and sample weather for 1000 consecutive hours from the Pecan Street data. The second studies the effect of changing context explicitly over 1500 hours. The first 500 hours are consecutive starting in either Dec., Jan. or Feb. The next 500 are consecutive in Jun., Jul. or Aug. The final 500 return to the same time steps as the first. In Austin, TX. the average temperature difference between the two contexts (summer and winter) is around 15 °C.

While performance is affected by the choice of user utility function, the model can accommodate any form of utility, as long as it depends only on attributes in the state vector. This stands in contrast with previous work on learning HVAC utilities (e.g., [30]).

We select the action for the current time step by solving an MDP, representing the next 24 hours, using value iteration. For EVOI estimation of queries, we use 10 response samples and a longer lookahead of 50 hours. We scale up the EVOI estimate by the time horizon of the experiment divided by the evaluation horizon, taking discounting into account: $\frac{\sum_{i=0}^{\text{true\_horizon}}(1-\gamma)^i}{\sum_{i=0}^{\text{sample\_horizon}}(1-\gamma)^i}$. This will tend to overestimate EVOI because a query response is likely more valuable in the near-term, before the context changes. Note that we do not consider horizon effects in EVOI estimates.

Query strategies: In each query strategy, a single candidate query is proposed at each time step, and then the strategy decides whether or not to ask the candidate query. The strategies differ in how they decide whether to ask the candidate query and how they integrate responses. The candidate query compares (i) a sampled *next state* from transitions induced by the current best action, and (ii) a sampled next state using the second best action. If the states are identical, we re-sample. This meta-strategy is effective because (i) it provides information that is immediately relevant, and (ii) it asks queries that tend to be close to the current state. More complex methods could increase performance at the cost of additional computation.[7]

We compare the performance of seven query strategies: three that are GP-based, two RF-based, and two baselines:

- GP_ALWAYS uses a GP-based preference prediction model. It does not calculate EVOI and always asks the proposed query.
- GP_ALWAYS_LIMIT is the same, but limited to 200 (250 in the setting with length 1500) queries.[8]
- GP_EVOI also uses a GP-based model, but only asks the proposed query if its EVOI is greater than its cost.
- RF_ALWAYS uses a random forest and always asks the proposed query (it does not use EVOI).
- RF_ALWAYS_LIMIT is the same, but limited to 200 (250) queries.
- OMNI is optimally omniscient, and takes the action of the optimal policy in the underlying (known-reward) MDP without querying. It provides an upper bound on performance, which cannot be realized practically.
- NULL always takes the null action, i.e., does nothing.

Both GP models use the squared exponential kernel with lengthscale 0.1. The random forest uses 25 decision trees.

Computing: We run all experiments on Intel Broadwell CPUs at 2.1 GHz. At time 1500—when all methods have their largest set of responses (i.e., maximal demands on GP computation)—all non-baseline strategies take 0.1s (on average) to select the next control action. GP_ALWAYS and RF_ALWAYS (and their query-limited versions) do not estimate EVOI and take 0.3s to select a query, whereas GP_EVOI takes 2.0s. The strategies also differ in the time required to evaluate rewards given the current query set: GP_ALWAYS takes 1.2s, RF_ALWAYS 0.6s, and GP_EVOI and the query-limited strategies 0.1s—they are more efficient since they ask fewer queries. All methods support real-time response.

Results (1000 steps): We first analyze total reward, including query cost, accrued by each strategy. Fig. 1 shows (average) reward accrued over time for five of the strategies. Because each user has different utility height and width, we normalize results w.r.t. the maximum policy value, max(OMNI), observed in that instance

[7]In theory, we could use EVOI to determine which query has highest value, but given the vast space of RVQs, we use this heuristic to limit query scope.

[8]We choose these limits because they are approximately the same as the average number of queries asked by GP_EVOI.

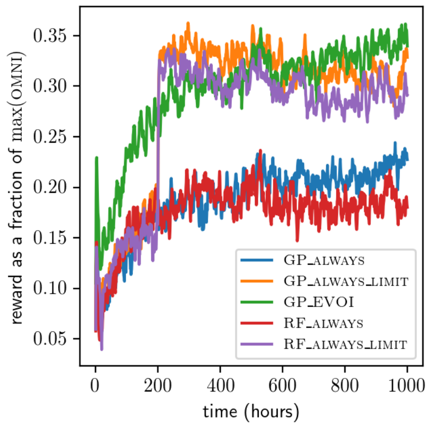

Figure 1: Reward accrued vs. time.

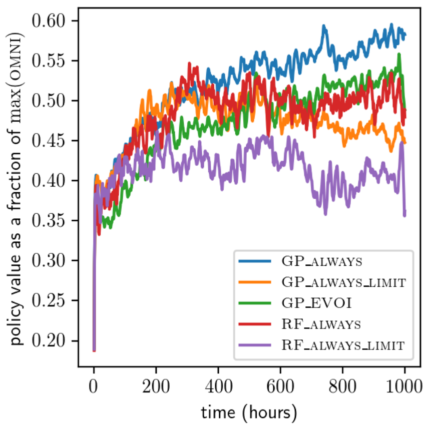

Figure 2: Policy value vs. time.

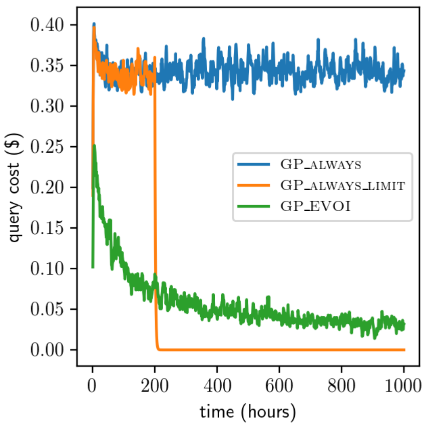

Figure 3: Query cost vs. time.

Query strategy	Mean total reward ($)	Mean discounted reward ($)
GP_ALWAYS	663.9	48.0
GP_ALWAYS_LIMIT	861.0	55.0
GP_EVOI	892.1	64.6
RF_ALWAYS	640.4	50.4
RF_ALWAYS_LIMIT	820.9	55.7
OMNI	1248.4	124.6
NULL	514.2	57.3

Figure 4: Mean total reward and discounted reward.[9]

by the OMNI baseline. This ensures users with large utility values do not dominate the data.

OMNI and NULL perform at a constant level of 44.6% and 18.8%, respectively, of max(OMNI), and are not shown in Fig. 1. GP_EVOI outperforms GP_ALWAYS and RF_ALWAYS at each point (each performs similarly to NULL). The differences are statistically significant from iteration 90 onward (with $p < 0.05$). We show below that GP_EVOI's success is due to its ability to achieve control policy quality similar to the other methods, while incurring much lower query cost. This illustrates that the ability to estimate query EVOI is invaluable in this domain.

The query-limited algorithms (GP_ALWAYS_LIMIT and RF_ALWAYS_LIMIT) represent a more traditional PE approach: they initially ask many queries to learn a high quality policy and exploit that policy, receiving high reward for the remainder of the instance. Nevertheless, these algorithms perform poorly from the perspective of *discounted* reward. Fig. 4 shows the total discounted and non-discounted rewards earned by the seven strategies. While the query-limited algorithms have comparable performance without discounting, GP_EVOI outperforms GP_ALWAYS_LIMIT by 19% and RF_ALWAYS_LIMIT by 18% with discounting,[10] and the query-limited strategies fail to outperform NULL. This illustrates a key advantage of the EE approach: deferring queries may increase discounted utility.

[9]Reflecting instance variability, total reward std. is around \$780 for all strategies and discounted reward std. around \$75.
[10]The difference is statistically significant at $p < 0.01$ using a paired t-test.

We next analyze the quality of the policy learned by each strategy. Fig. 2 shows (ground truth) expected value of the current (optimal) control policy (as specified by Obs. 2, assuming no more queries are asked) evaluated under the *true reward function*. OMNI and NULL (not shown) perform at a constant level of 80% and 1.7%, respectively, of max(OMNI). All non-baseline strategies perform similarly w.r.t. policy value, with GP_ALWAYS having a slight advantage over GP_EVOI and the query-limited algorithms falling behind their unlimited counterparts. The policy quality of the query-limited strategies increases until they stop querying at time 200 and declines slowly thereafter. Because the instance is relatively short, the quality does not decline much: 10% on average for GP_ALWAYS_LIMIT from time 200 (when querying stops) to time 1000. GP_EVOI falls behind GP_ALWAYS because it asks many fewer queries: 43.8 ($\sigma = 9.7$) queries on average by time 100, 121.1 ($\sigma = 25.8$) by time 500 and 179.6 ($\sigma = 39.4$) by time 1000.

The query numbers above lead to significant query cost savings for GP_EVOI. Fig. 3 shows the query cost accrued by GP_ALWAYS, GP_ALWAYS_LIMIT, and GP_EVOI (RF_ALWAYS and RF_ALWAYS_LIMIT are not shown since their query costs are similar to GP_ALWAYS and GP_ALWAYS_LIMIT, respectively; OMNI and NULL are not shown since they ask no queries). Low query cost is a major factor in GP_EVOI's strong performance. By time 100, GP_EVOI has incurred total query cost of \$15.4 vs. \$35 for GP_ALWAYS. By time 500 and 1000, GP_EVOI's costs are \$40.2 and \$58.4, resp. (vs. \$173 and \$348 for GP_ALWAYS). Average cost per (asked) query for GP_EVOI is similar to that of GP_ALWAYS, \$0.32 vs. \$0.35, showing that GP_EVOI's advantage lies largely in asking queries with higher EVOI rather than lower cost.

Results (1500 steps, two contexts): Our second set of experiments studies (i) how the strategies handle changing context; and (ii) whether GP_EVOI recognizes that queries in a previous context have low EVOI after spending many steps in a different context.

Figs. 5, 6 and 7 show the effects of changing context. The reward accrued by the strategies (Fig. 5) is quite similar to the original experiments. The query-limited strategies suffer heavily in the second context because they lack up-to-date reward information. This illustrates a key challenge for traditional preference elicitation techniques in a setting with evolving context—the need to detect that

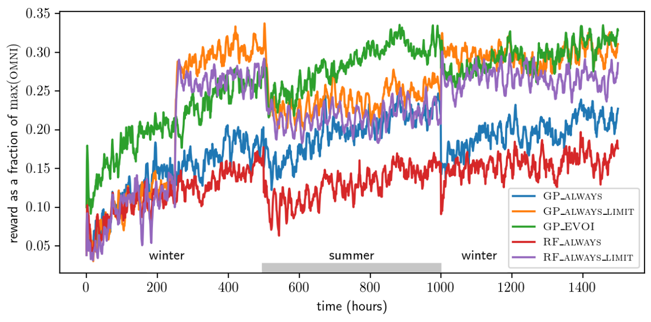

Figure 5: Reward accrued vs. time.

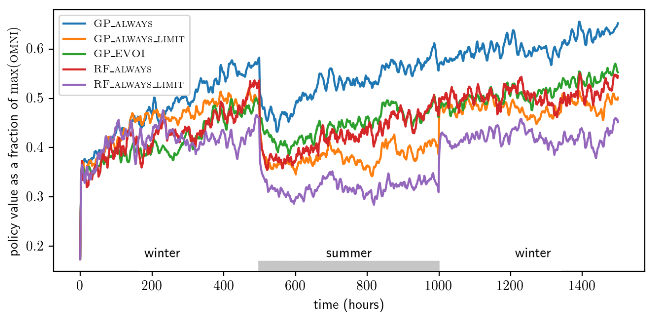

Figure 6: Policy value vs. time.

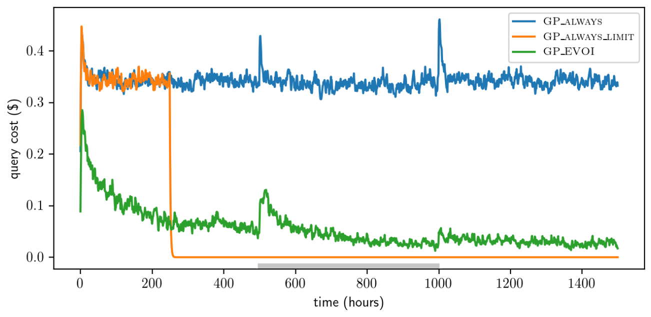

Figure 7: Query cost vs. time.

their information is out-of-date and reinitiate querying. In contrast, Fig. 7 shows that GP_EVOI begins to query more aggressively after the shift occurs at time 500 (transition to summer). Note that RF_ALWAYS and GP_ALWAYS also experience a spike in query cost at time 500, but this is due to the lack of a useful state history at the time of transition, resulting in higher query costs and lower accuracy. We can tell that GP_EVOI's increase is not solely due to a less useful state history because there is no corresponding increase for GP_EVOI at the second context change at time 1000 (return to winter)—GP_EVOI recognizes that it already has a high quality model for the new context and does not increase its querying.

In policy quality terms (Fig. 6), the first context change results in a drop of around 17% on average, which is similar across the non-baseline strategies. The drop occurs because the strategies lack information about the reward function in the new context and not because the optimal policy has a different value: OMNI achieves an average policy quality of 76% and 74% of max(OMNI) in the winter and summer contexts, respectively. As in the previous experiments, GP_EVOI lags behind GP_ALWAYS in policy quality, but asks many fewer queries, and thus achieves a higher reward.

The context change slightly increases the number of queries asked by GP_EVOI: 46.4 ($\sigma = 9.8$) by time 100, 132.9 ($\sigma = 19$) by time 500, 204 ($\sigma = 23.9$) by time 1000 and 254 by time 1500 ($\sigma = 36.5$).

The difference in how GP_EVOI handles the context changes compared to the other strategies indicates that the benefit of query EVOI estimation increases with a rapidly changing context. For example, in a model where the context rotated every 10 steps, GP_EVOI could concentrate its queries in the later periods of each context, when more state history is available. Meanwhile, the strategies without EVOI estimation would constantly suffer higher query costs due to the lack of relevant state history.

7 CONCLUSION

We have introduced *experiential elicitation*, a framework for preference elicitation in settings where a user's ability to (easily) answer queries is context dependent. Our model of experiential elicitation has tight connections to POMDPs, RL and uncertain-reward MDPs. We studied a new query type for GPs, the *relative value* query, that is well-suited to the home HVAC domain. We showed that GPs naturally accommodate RVQs and offer effective EVOI computation, which is critical for trading off query cost vs. value. Our experiments show that GP-based elicitation using EVOI outperforms other natural baselines in the HVAC setting.

Interesting future directions for EE remain. Most obvious is the significant gap between the performance of our approach and that of the optimal omniscient algorithm. While this gap cannot be closed entirely, a more sophisticated query strategy may have significant effect. In particular, the model allows multiple queries per time step, which our strategy does not use. With additional computational resources, we could use heuristics to suggest a list of potential queries, evaluate myopic EVOI for each and then ask the query (or queries) that offers the most EVOI net of query cost.

General EE problems may be much harder than the well-behaved HVAC problem, where there is limited incentive to deviate from a simple control strategy to seek out better queries. We showed (Example 1) that in the worst case, the EE system may be required to make a deviation of unbounded cost from the optimal policy to receive any reward at all, and the approach we take in the HVAC domain would perform poorly in such a setting. It remains to be seen whether such difficult instances arise in practice.

In the HVAC domain, it may be desirable to integrate budget constraints, i.e., the optimal policy in the MDP is subject to a bound on expected spending. Our approach may be extensible to this setting through the use of budgeted constrained MDP techniques [1].

The EE approach is natural for (mobile or other) personal assistants that need to learn the user's preferences about types and timing of notifications, reminders, recommendations, etc.

ACKNOWLEDGMENTS

Perrault was supported by OGS. We gratefully acknowledge the support of NSERC.

REFERENCES

[1] Eitan Altman. 1999. *Constrained Markov Decision Processes.* Chapman and Hall, London.

[2] Eyal Beigman and Rakesh Vohra. 2006. Learning from Revealed Preference. In *Proceedings of the Seventh ACM Conference on Electronic Commerce (EC'06).* Ann Arbor, 36–42.

[3] Edwin V. Bonilla, Shengbo Guo, and Scott Sanner. 2010. Gaussian Process Preference Elicitation. In *Advances in Neural Information Processing Systems 23 (NIPS-10).* Vancouver.

[4] Craig Boutilier. 2002. A POMDP Formulation of Preference Elicitation Problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02).* Edmonton, 239–246.

[5] Craig Boutilier. 2013. Computational Decision Support: Regret-based Models for Optimization and Preference Elicitation. In *Comparative Decision Making: Analysis and Support Across Disciplines and Applications*, P. H. Crowley and T. R. Zentall (Eds.). Oxford University Press, Oxford, 423–453.

[6] Darius Braziunas and Craig Boutilier. 2010. Assessing Regret-based Preference Elicitation with the UTPREF Recommendation System. In *Proceedings of the Eleventh ACM Conference on Electronic Commerce (EC'10).* Cambridge, MA, 219–228.

[7] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[8] Urszula Chajewska, Daphne Koller, and Ronald Parr. 2000. Making Rational Decisions Using Adaptive Utility Elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00).* Austin, TX, 363–369.

[9] Li Chen and Pearl Pu. 2004. *Survey of preference elicitation methods.* Technical Report 52659. EPFL.

[10] Wei Chu and Zoubin Ghahramani. 2005. Preference learning with Gaussian processes. In *Proceedings of the Twenty-second International Conference on Machine Learning (ICML-05).* Bonn, 137–144.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2009. *The elements of statistical learning.* Springer-Verlag New York.

[12] George A. Gescheider. 2013. *Psychophysics: the fundamentals.* Psychology Press.

[13] Hillary A. Holloway and Chelsea C. White, III. 2003. Question Selection for Multiattribute Decision-aiding. *European Journal of Operational Research* 148 (2003), 525–543.

[14] Bowen Hui and Craig Boutilier. 2008. Toward Experiential Utility Elicitation for Interface Customization. In *Proceedings of the Twenty-fourth Conference on Uncertainty in Artificial Intelligence (UAI-08).* Helsinki, 298–305.

[15] Kshitij Judah, Alan Paul Fern, Prasad Tadepalli, and Robby Goetschalckx. 2014. Imitation Learning with Demonstrations and Shaping Rewards. In *Proceedings of the Twenty-eighth AAAI Conference on Artificial Intelligence (AAAI-14).* Quebec City, 1890–1896.

[16] Daniel Kahneman. 2011. *Thinking, fast and slow.* Macmillan.

[17] Tiep Le, Atena M Tabakhi, Long Tran-Thanh, William Yeoh, and Tran Cao Son. 2018. Preference elicitation with interdependency and user bother cost. 1459–1467.

[18] Jonas Mockus. 1989. *Bayesian approach to global optimization: theory and applications.* Vol. 37. Springer Netherlands.

[19] Andrew Ng and Stuart Russell. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00).* Stanford, CA, 663–670.

[20] Phillip Odom and Sriraam Natarajan. 2016. Active advice seeking for inverse reinforcement learning. In *Proceedings of the Fifteenth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-16).* Singapore, 512–520.

[21] Pollara. 2013. BMO Psychology of House Hunting Report. https://newsroom.bmo.com/2013-05-02-BMO-Psychology-of-House-Hunting-Report-Home-Buyers-Visited-an-Average-of-10-Homes-Before-Buying. (2013). Accessed: 2018-05-08.

[22] Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. 2006. An Analytic Solution to Discrete Bayesian Reinforcement Learning. In *Proceedings of the Twenty-third International Conference on Machine Learning (ICML-06).* Pittsburgh, 697–704.

[23] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, New York.

[24] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning.* The MIT Press.

[25] Kevin Regan and Craig Boutilier. 2009. Regret-based Reward Elicitation for Markov Decision Processes. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI-09).* Montreal, 454–451.

[26] Kevin Regan and Craig Boutilier. 2011. Eliciting Additive Reward Functions for Markov Decision Processes. In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI-11).* Barcelona, 2159–2164.

[27] Joshua D. Rhodes, Charles R. Upshaw, Chioke B. Harris, Colin M. Meehan, David A. Walling, Paul A. Navrátil, Ariane L. Beck, Kazunori Nagasawa, Robert L. Fares, Wesley J. Cole, et al. 2014. Experimental and data collection methods for a large-scale smart grid deployment: Methods and first results. *Energy* 65 (2014), 462–471.

[28] Paul A. Samuelson. 1948. Consumption Theory in Terms of Revealed Preference. *Economica* 15, 60 (1948), 243–253.

[29] Tuomas Sandholm and Craig Boutilier. 2006. Preference Elicitation in Combinatorial Auctions. In *Combinatorial Auctions*, P. Crampton, Y. Shoham, and R. Steinberg (Eds.). MIT Press, Cambridge, MA, 233–264.

[30] Mike Shann and Sven Seuken. 2013. An Active Learning Approach to Home Heating in the Smart Grid. In *Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI-13).* Beijing, 2892–2899.

[31] Mike Shann and Sven Seuken. 2014. Adaptive home heating under weather and price uncertainty using GPs and MDPs. In *Proceedings of the Thirteenth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-14).* Paris, 821–828.

[32] Richard D. Smallwood and Edward J. Sondik. 1973. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Operations Research* 21 (1973), 1071–1088.

[33] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS-12).* Harrahs and Harveys, Lake Tahoe, 2951–2959.

[34] Ngoc Cuong Truong, Tim Baarslag, Gopal Ramchurn, and Long Tran-Thanh. 2016. Interactive scheduling of appliance usage in the home. In *Proceedings of the Twenty-five International Joint Conference on Artificial Intelligence (IJCAI-16).* 869–875.

[35] Huan Xu and Shie Mannor. 2009. Parametric Regret in Uncertain Markov Decision Processes. In *48th IEEE Conference on Decision and Control.* Shanghai, 3606–3613.