# Planning Delayed-Response Queries and Transient Policies under Reward Uncertainty

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

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

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

## ABSTRACT

We address situations in which an agent with uncertainty in rewards can selectively query another agent/human to improve its knowledge of rewards and thus its policy. When there is a time delay between posing the query and receiving the response, the agent must determine how to behave in the *transient* phase while waiting for the response. Thus, in order to act optimally the agent must jointly optimize its transient policy along with its query. In this paper, we formalize the aforementioned joint optimization problem and provide a new algorithm called JQTP for optimizing the Joint Query and Transient Policy. In addition, we provide a clustering technique that can be used in JQTP to flexibly trade performance for reduced computation. We illustrate our algorithms on a machine configuration task.

## Keywords

Human-robot/agent interaction, Reward structures for learning, Single agent Learning

## 1. INTRODUCTION

The work described in this paper addresses a problem that arises when one agent (which we will refer to as the robot) is acting on behalf of another agent (which we will refer to as the (human) operator). The robot should act so as to bring about states of the world that satisfy the preferences of the operator, but in complex environments the effort required of the operator to express preferences that account for every possible circumstance is prohibitive. In this paper, we assume that the operator's preferences are expressed as a reward function. Thus, in all but very simple domains the robot is inherently uncertain about some aspects of the operator's true reward function. Should the world the robot encounters veer into regions where its uncertainty over the operator's true rewards is high, the robot's only way to reduce its uncertainty is to ask the operator one or more queries.

Such a robot is semi-autonomous: it can behave autonomously for considerable intervals of time (in more routine circumstances where the operator's preferences are unambiguously clear) but might periodically need human assistance. Because its human operator is only called on to help infrequently, the operator's valuable cognitive capabilities will be shared with other tasks (such as operating other robots). Thus, we assume that the operator is not necessarily paying attention to the robot at all times. When the robot asks for assistance, the operator might require some (variable) time to complete or suspend her current cognitive activities, to understand what exactly the robot is asking, to introspect to discern an appropriate answer, and to articulate the response. Delays in communication channels can further extend the time lapse between when the robot asks a query and when it receives a response.

Because getting a response from the operator incurs cost (distracting the operator from other possibly valuable cognitive tasks), the robot must choose with care when to query and what to query about. Furthermore, because receiving a response might take some time, the robot needs to decide what to do while waiting. In fact, because the value of a query response often depends on what state the robot is in when that response arrives, the decisions about what to ask and what to do while waiting for the response are intertwined.

We focus in this paper on the formulation and solution of this joint sequential decision problem that the robot faces. We should note that in the treatment we provide here, the other agent in the system—the human operator—is simply assumed to provide a (truthful) response to the posed query after some (stochastic) delay. Thus, the sequential decision problem we consider here is multiagent only in a degenerate sense, in that the query response policy of the operator is assumed fixed. Nevertheless, solving the robot's intertwined problems of deciding on a query and on a transient policy to follow while awaiting the response is itself challenging, and is influenced by the operator's responsiveness.

In the remainder of this paper, we formally define this problem, and then identify aspects of the problem structure that can be exploited as part of a greedy myopic approach. Finally, we describe a technique for reducing computation in a flexible way by clustering together hypothetical future reward function beliefs arising from potential responses to queries.

## 2. PROBLEM DEFINITION

In this section, we define the two interacting processes that constitute the overall problem faced by the (robot) agent, the decision-making process that models the agent's interaction with the environment and then the reward-knowledge process which models the agent's interaction with the operator.

### Decision-Making Process

First we define the two elements that form the decision-making process faced by the agent, namely controllable Markov processes and uncertainty over reward functions.

In a Controlled Markov Process (CMP), at time $t$ the agent oc-

*The Seventh Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty* (*MSDM-2012*), held in conjunction with *AAMAS*, June 2012, in Valencia, Spain.

cupies state $s_t \in S$, executes an action $a_t \in A$, and stochastically transitions to state $s_{t+1}$ with probability governed by transition function $T(s_{t+1}|s_t, a_t)$. Given any reward function $R : S \times A \rightarrow \Re$ that maps states and actions to reals, the value function of a policy $\pi \in \Pi$, where $\pi : S \rightarrow A$, is $V_R^\pi(s) \triangleq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s, \pi]$, the expected discounted sum of rewards achieved when the agent starts in state $s$ and behaves according to policy $\pi$. Thus, a reward function $R$ induces a partial ordering over policies for a CMP. The optimal policy $\pi_R^*$ satisfies $V_R^{\pi_R^*} \geq V_R^\pi\ \forall \pi \in \Pi$ and is guaranteed to exist; its value function $V_R^{\pi_R^*}$ is termed the optimal value function and denoted $V_R^*$. The optimal value function and corresponding optimal policy can be computed by algorithms such as value iteration and policy iteration.

CMPs are identical to MDPs except that, unlike MDPs, CMPs do not assume that the rewards are an observation from the environment. Instead, CMPs treat rewards as an expression of a (operator's) preference ordering over policies.

In this paper, we assume that an agent knows the CMP, but has uncertainty over which reward function is the operator's *true* reward function. The uncertainty is expressed as a distribution $\psi \in \Psi$ over a finite set of reward functions $\{R_1, R_2, \cdots, R_n\}$ (we will interchangeably use $\psi$ as a distribution and as a knowledge-state). The expected value function with respect to a reward function distribution $\psi$ that never changes is defined as follows:

$$\begin{aligned} V_\psi^\pi(s) &\triangleq \mathbb{E}_{R\sim\psi}[V_R^\pi(s)] \\ &= \mathbb{E}_{R\sim\psi}\Big[\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s, \pi]\Big] \\ &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \overline{R}_\psi(s_t, a_t)|s_0 = s, \pi], \end{aligned} \tag{1}$$

where $R \sim \psi$ denotes reward function $R$ drawn with probability governed by $\psi$, and where $\overline{R}_\psi(s, a) \triangleq \sum_{i=1}^n \psi(i) R_i(s, a)$ is the **mean-reward** for state-action pair $s, a$ under distribution $\psi$ ($\psi(i)$ denotes the probability assigned to reward function $R_i$ by $\psi$). The *Bayes-optimal* policy $\pi_\psi^* \triangleq \arg\max_\pi V_\psi^\pi$ and the corresponding Bayes-optimal value function is denoted $V_\psi^*$. Note that solving for the Bayes-optimal policy with uncertainty only over rewards as above is no harder than solving a traditional MDP [8], so long as the mean-reward function can be calculated efficiently.

In this paper we consider situations in which an agent's decision-making problem is a CMP with uncertain rewards, but where the uncertainty over reward functions is not fixed for all time because the agent has the capability of actively querying its operator to acquire information about the distribution over rewards.

## Reward-Knowledge Process

Here we consider CMPs with reward uncertainty in which the agent has the ability to query its operator about the reward function, where the operator in turn responds to these queries with some stochastic temporal delay. The response to a query is informative in some way about the true reward function, and thus potentially leads to an update to the distribution over rewards, but otherwise has no effect on the CMP. We will assume that the agent can only have one outstanding query at a time, and that each query has an associated cost. We now formalize the reward-knowledge process as a controlled *semi*-Markov process (CsMP).

The state of the reward-knowledge process is the distribution $\psi$ over rewards $\{R_1, R_2, \cdots, R_n\}$ and the number of time steps since the last querying action was taken. Let the initial distribution or state be denoted $\psi_0$. The actions of this reward-knowledge process are queries from set $Q = \{q_1, q_2, \cdots, q_m\}$. Examples of the form of queries include asking about the reward for some state-action pair, and about the optimal action in some state. For the purposes of developing this formal framework, the form of the query $q$ as well as the form of the response $o$ is irrelevant. What matters is at what time step the query returns a response, and what changes to the distribution over reward functions are caused by the query response. Specifically, let $F_q(\tau)$ denote the probability that query $q$ returns $\tau$ time steps after it is asked; we will assume that all queries return within $\tau_{max}$ time steps of being asked.

Let $H_q(\psi'|\psi)$ represent the probability that the distribution over reward functions is $\psi'$ given that query $q$ was asked in distribution $\psi$; each possible response to a query can potentially lead to a different next distribution $\psi'$. Denote the support of $H_q(\psi'|\psi)$, i.e., the set of posterior beliefs possible given $q$ is asked in belief state $\psi$, as $\Psi(\psi, q)$. This set can be computed by considering all possible responses to $q$. For example, in the experiments in the paper, we consider action-queries, which query for the optimal action given some state. The response can then be used to update the agent's reward function distribution, and so $\Psi(\psi, q)$ can be computed by performing a Bayes update for each action possible in the state being queried (see [4],[8] for details).

We emphasize the semi-Markov nature of this controlled process; the state does not change and no query-action choice is available until the previous query returns. Finally, the cost of asking query $q$ is denoted $c(q)$, where $c(q) \leq 0$. The states and actions of the decision-making process will be distinguished from the corresponding quantities in the reward-knowledge process by calling the latter *knowledge-states* and *queries* respectively.

## Joint Reward-Query CMP

Above we defined two factored processes: a decision-making process that is modeled as a CMP, and a reward-knowledge process that is modeled as a CsMP. Given fixed policies in each of the two processes, their dynamics, i.e., their evolution of state, are completely uninfluenced by each other. Thus the only way these processes interact is that each influences what the agent *should* do in the other process. Specifically, the reward-knowledge process defines the evolving mean-reward function for the decision-making process. Therefore, the best choice of query in the reward-knowledge process depends on the likely state of the agent in the decision-making process when the query returns, and the best behavior of the agent in the decision-making process whilst waiting for a query to return depends on the query and its likely responses and temporal delay. Planning in domains with this highly structured interaction is at the heart of this paper. Hereafter we will refer to the joint reward-knowledge and decision-making processes as the Reward-Query CMP or RQCMP.

## 3. JOINT QUERY AND TRANSIENT-POLICY PLANNING

The general problem of planning jointly optimal query and action policies in RQCMPs presents a challenging optimization problem, which in this paper we will approximate by repeated myopic planning. Specifically, we consider what query an agent should select and how it should act whilst waiting for the query to return, ignoring the possible effects of future queries. It is repeated myopic planning because we repeat this myopic joint optimization whenever a query returns.

**Query+Transient-Policy Value Function.** Consider the expected discounted summed value of mean-rewards as a function of state $s$, query $q$, and transient non-stationary policy $\pi \in \Pi^{\tau_{max}}$ (where $\Pi^{\tau_{max}}$ is the set of non-stationary policies defined from the current time step to $\tau_{max}$ additional time steps). We emphasize that $\pi$ is transient because it terminates when the query returns, and non-stationary because in general the policy maps states and "time since the start of the policy" to actions. This query+transient-policy value function can be decomposed into three components by exploiting the factored structure of RQCMPs. Formally, when the knowledge-state is $\psi$ and there is no outstanding query (recall that only one outstanding query is allowed at a time),

$$\begin{aligned} Q_\psi(s,q,\pi) &= c(q) + r^q_\psi(s,\pi) \\ &+ \sum_{s'} T^q(s'|s,\pi) \sum_{\psi' \in \Psi(\psi,q)} H_q(\psi'|\psi) V^*_{\psi'}(s') \end{aligned} \tag{2}$$

(See Appendix for a derivation.) The first term, $c(q)$, is the cost of asking the query. The second term is the expected value of the discounted sum of mean-reward obtained by the transient policy until the query returns; the expectation here is over both the random state transitions in the decision-making process as well as the randomness over return times. More formally, the second term is defined as

$$r^q_\psi(s,\pi) \triangleq \sum_{\tau=0}^{\tau_{\max}} F_q(\tau) \mathbb{E}\Big[\sum_{k=0}^{\tau-1} \gamma^k \overline{R}_\psi(s_k,a_k)|s_0 = s,\pi\Big].$$

The third term is the expected value of the state at the time the query returns; here the expectation is over the random state of the decision-making process when the query returns. Note that the $V^*_{\psi'}$ is the optimal posterior value that is computed assuming no further queries; hence the myopic nature of the query+transient-policy value function. More formally,

$$T^q(s'|s,\pi) \triangleq \sum_{\tau=1}^{\tau_{\max}} F_q(\tau)\gamma^\tau P(s'|s,\pi,\tau)$$

where $P(s'|s,\pi,\tau)$ is the probability that the state in the decision making process $\tau$ steps after the non-stationary transient policy $\pi$ starts in state $s$ is $s'$. Furthermore note that $T^q$ takes into account the effect of discounting because of the delay in query response. In summary, Equation 3 defines the expected discounted sum of mean-rewards obtained by an agent if it asks query $q$ in state $s$ and then behaves according to policy $\pi$ until the query returns, after which the agent behaves optimally forever with respect to the posterior distribution over rewards (without asking any additional queries).

Given the definition of the query+transient-policy value function above, we can define three optimization problems (we will use the first and second in our empirical results below; the third is included here for completeness):

**Joint Query and Transient-Policy Optimization**

$$\langle q^*, \pi^* \rangle | s, \psi = \arg \max_{q \in Q, \pi \in \Pi^{\tau_{\max}}} Q_\psi(s,q,\pi) \tag{3}$$

where $\Pi^{\tau_{\max}}$ is the set of non-stationary policies of length $\tau_{\max}$.

**Transient-Policy Optimization**

$$\pi^* | s, q, \psi = \arg \max_{\pi \in \Pi^{\tau_{\max}}} Q_\psi(s,q,\pi) \tag{4}$$

**Query Optimization**

$$q^* | s, \psi, \pi = \arg \max_{q \in Q} Q_\psi(s,q,\pi)$$

**Relationship to Options.** The form of the query+transient-policy value function in Equation 3 resembles the option-value functions constructed in the options literature [13], where algorithms have been developed to solve for the option-value function. However, our form has two key differences. First, we have a joint optimization over the query space and transient-policy space, rather than an optimization over some option space. The transient-policy space is much larger than the usual case of a small number of options usually considered. Second, we have an additional inner sum over knowledge-states that takes into account possible changes caused by the query and its response.

## Algorithm Components

We now present our **J**oint **Q**uery and **T**ransient **P**olicy (JQTP) algorithm, which solves the Joint Query and Transient-Policy Optimization problem (Equation 3) for a given $s$ and $\psi$. JQTP solves the transient-policy optimization problem for each possible query by solving a nested optimization problem as described below, and then picks the best query along with its paired optimal transient policy. We rewrite the transient-policy optimization problem below for reference (noting that $c(q)$ does not depend on the transient policy):

$$\begin{aligned} \pi^*|s,q,\psi = \arg \max_{\pi \in \Pi^{\tau_{\max}}} \Big\{ r^q_\psi(s,\pi) \\ + \sum_{s'} T^q(s'|s,\pi) \sum_{\psi' \in \Psi(\psi,q)} H_q(\psi'|\psi) V^*_{\psi'}(s') \Big\}. \end{aligned}$$

We can solve this transient-policy optimization problem by defining an induced terminal-reward problem in which the value for a particular non-stationary $\pi$ depends on the summed discounted rewards $r^q_\psi(s,\pi)$ it receives until the query is returned, at which point it receives terminal reward $\sum_{\psi' \in \Psi(\psi,q)} H_q(\psi'|\psi) V^*_{\psi'}(s')$ depending on which state $s'$ it occupies. Note that both of the terms in the terminal reward, $H_q(\psi'|\psi)$ and $V^*_{\psi'}(s')$ are independent of $\pi$ (the distribution over $s'$ is a function of $\pi$, which links the reward-knowledge and decision-making processes). Thus, this terminal-reward process can be represented as a non-stationary CMP, solvable by standard MDP solving techniques.

However, constructing this transient process requires the expensive computation of $\sum_{\psi' \in \Psi(\psi,q)} H_q(\psi'|\psi) V^*_{\psi'}(s')$ for every $s'$ in which the agent might receive the response to its query, each of which requires solving the mean-reward MDP associated with each possible $\psi'$. Thus, as specified, JQTP's asymptotic complexity depends linearly on both the number of states reachable within $\tau_{max}$ time steps, and the number of possible posteriors considering all possible queries (the latter of which amounts to $\sum_{q \in Q} |\Psi(\psi,q)|$ value calculations). The latter issue precludes JQTP as stated from being applied to problems with large query spaces without further approximations. In the next section, we present such an approximation in the form of a clustering technique.

## Posterior Belief Clustering

In this section, we propose reducing the number of different optimal posterior value ($V^*_{\psi'}(\cdot)$) calculations in Equation 3 by clustering posterior beliefs, and calculating the posterior value only once for each cluster (sharing that value among each member of the cluster). This idea is founded on the principle that similar posteriors should share similar values. We now show that this principle has theoretical justification by bounding the maximal difference between the optimal posterior value functions for posteriors which induce similar mean reward functions. The following Lemma, a result similar to those proved in [10] and [15], will be useful in proving our bound:

LEMMA 1. *If* $||R_1 - R_2||_\infty \leq \epsilon$, *then for all* $\pi$,
$||V^\pi_{R_1} - V^\pi_{R_2}||_\infty \leq \frac{\epsilon}{1-\gamma}$,
*where* $||x||_\infty = \max_i x_i$.

PROOF. Using the fact that $V^\pi_{R_1} = (I - \gamma P^\pi)^{-1} R_1$, and $V^\pi_{R_2} = (I - \gamma P^\pi)^{-1} R_2$,

$$||V^\pi_{R_1} - V^\pi_{R_2}||_\infty = ||(I - \gamma P^\pi)^{-1}(R_1 - R_2)||_\infty,$$

where $(I - \gamma P^\pi)^{-1}(R_1 - R_2)$ is the value function of policy $\pi$ for the reward function $R' = R_1 - R_2$. By assumption $||R'||_\infty \leq \epsilon$ and thus with reward function $R'$ the maximum value for any state would be $\frac{\epsilon}{1-\gamma}$ and the minimum value for any state would be $-\frac{\epsilon}{1-\gamma}$. □

THEOREM 1. *Given two reward functions* $R_1$ *and* $R_2$, *if* $||R_1 - R_2||_\infty \leq \epsilon$, *then* $||V^*_{R_1} - V^*_{R_2}||_\infty \leq \frac{\epsilon}{1-\gamma}$.

PROOF. Let $\pi^*_1$ be optimal w.r.t. $R_1$ and $\pi^*_2$ be optimal w.r.t. $R_2$. Then, we know that $\forall s$

$$\begin{aligned} V^{\pi^*_1}_{R_1}(s) &\leq V^{\pi^*_1}_{R_2}(s) + \frac{\epsilon}{1-\gamma} \text{(By Lemma 1)} \\ &\leq V^{\pi^*_2}_{R_2}(s) + \frac{\epsilon}{1-\gamma} \end{aligned}$$

and

$$\begin{aligned} V^{\pi^*_1}_{R_1}(s) &\geq V^{\pi^*_2}_{R_1}(s) \\ &\geq V^{\pi^*_2}_{R_2}(s) - \frac{\epsilon}{1-\gamma} \text{(By Lemma 1)} \end{aligned}$$

Thus $\forall s$
$V^{\pi^*_2}_{R_2}(s) - \frac{\epsilon}{1-\gamma} \leq V^{\pi^*_1}_{R_1}(s) \leq V^{\pi^*_2}_{R_2}(s) + \frac{\epsilon}{1-\gamma}$. □

We now show that when the agent has uncertainty only in rewards, the maximal difference in optimal value between two posteriors can be bounded by the maximal difference between the mean reward functions induced by the two posteriors:

THEOREM 2. *Given distributions* $\psi_1$ *and* $\psi_2$ *over reward functions,*
$||\mathbb{E}_{R_1 \sim \psi_1}[V^*_{R_1}] - \mathbb{E}_{R_2 \sim \psi_2}[V^*_{R_2}]||_\infty \leq \frac{1}{1-\gamma}||\overline{R}_{\psi_1} - \overline{R}_{\psi_2}||_\infty$.

PROOF. $||\mathbb{E}_{R_1 \sim \psi_1}[V^*_{R_1}] - \mathbb{E}_{R_2 \sim \psi_2}[V^*_{R_2}]||_\infty$
$= ||V^*_{\overline{R}_{\psi_1}} - V^*_{\overline{R}_{\psi_2}}||_\infty$
$\leq \frac{1}{1-\gamma}||\overline{R}_{\psi_1} - \overline{R}_{\psi_2}||_\infty$. □

We now present our clustering algorithm, which given the agent's current state $s$ and belief $\psi$, groups all possible posterior beliefs ($\psi' \in \cup_{q \in Q} \Psi(\psi, q)$).

**K-means Clustering of Posterior Beliefs.** Theorem 2 above shows that if the mean reward functions induced by two posteriors are close, the difference in expected value for a particular state given the two posteriors is small. We make use of this fact to save computation within JQTP in calculating $\sum_{\psi' \in \Psi(\psi,q)} H_q(\psi'|\psi) V^*_{\psi'}(s')$ for each $q$ and state $s'$ reachable (from $s$) as follows.

Given the agent's current state is $s$, we cluster all posteriors $\psi_1$, $\psi_2$ in $\cup_{q \in Q} \Psi(\psi, q)$ using K-means with the following distance function:

$$D(\psi_1, \psi_2) \triangleq |\overline{R}_{\psi_1} - \overline{R}_{\psi_2}|_1,$$

where each posterior $\psi'$ is assigned to cluster $J(\psi')$. We can then use the clustering to compute the needed terms just for the cluster center and to use them as approximations to the terms for all the posteriors that fall into that cluster. Specifically, for each $s'$ the agent could occupy when the query is returned and cluster $j$, we compute the value given the posterior mean reward function $\overline{\psi}_j$ represented by the center of $j$,

$$\tilde{V}^*_j(s') \triangleq V^*_{\overline{\psi}_j}(s).$$

Then, for each $s'$ and $q$, we estimate

$$\sum_{\psi' \in \Psi(\psi,q)} H_q(\psi'|\psi) V^*_{\psi'}(s') \approx \sum_{\psi' \in \Psi(\psi,q)} H_q(\psi'|\psi) \tilde{V}^*_{J(\psi')}(s').$$

We term the JQTP algorithm that utilizes this clustering estimation technique JQTP_Clustering. Note that the number of clusters is an input parameter to the K-means algorithm and hence to the JQTP_Clustering algorithm; this will determine the amount of computational savings achieved by clustering as well as the quality of the approximation. In Section 4 we will show empirical results that demonstrate the significant tradeoffs in using JQTP_Clustering compared to JQTP.

## 4. EXPERIMENT: MACHINE CONFIGURATION

In Section 3 we introduced JQTP, an algorithm that, at a particular decision point, finds an approximately optimal joint query and action policy as prescribed by Equation 3. The approximation is that JQTP is myopic, assuming that only one query can be asked and can only be asked at the current time step. In addition, we presented a method for clustering posterior beliefs that can be used to reduce the computation required by JQTP, and proved a worst-case bound on the difference in value given two posterior beliefs. In this section, we give a preliminary empirical evaluation of JQTP, comparing its performance and computational needs to alternative algorithms for computing a joint query and transient policy. In addition, we examine the degree to which clustering to reduce computation reduces the quality of the joint query and transient policy found.

**Domain Description.** We conduct our evaluation in the Machine Configuration domain, in which the agent's task is to build a machine for its user consisting of $N$ components $\{c_1, c_2, ..., c_N\}$. There are $M$ different types $\{t^i_1, t^i_2, ..., t^i_M\}$ that can be assigned to each

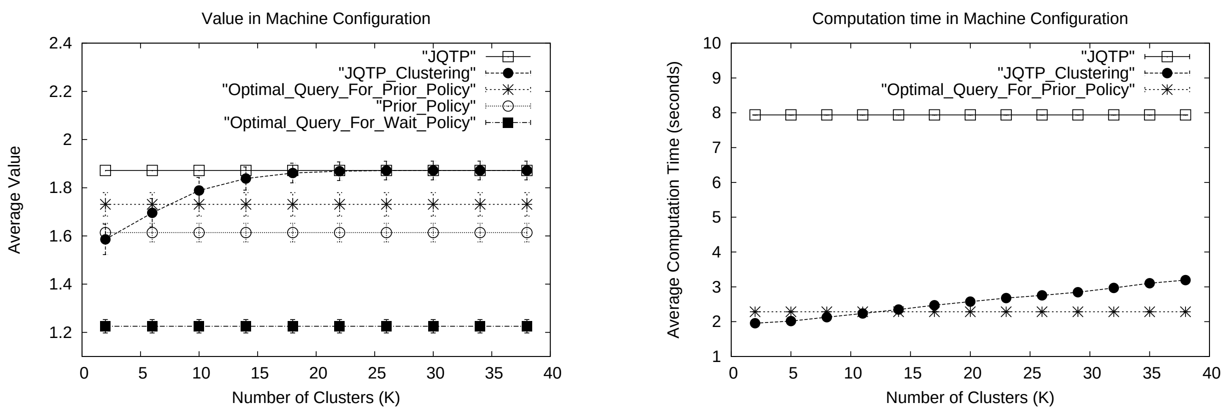

**Figure 1: (a) Expected value of JQTP with clustering as a function of number of clusters. Expected value of other non-clustering methods shown for comparison. Error bars are confidence intervals with p-value $0.05$. (b) Average computation time to select query and transient policy in seconds. Values were computed via Value Iteration, and value functions for each candidate reward function were precomputed to allow for fast Bayes updates. Computation for Optimal_Query_For_Wait_Policy not shown but similar to Optimal_Query_For_Prior_Policy as they both compute one value per potential query, and computation for Prior_Policy is on the order of miliseconds. Error bars are negligible. Experiments were run on a 3.07GHz Intel ®Core $^{\text{TM}}$i7 processor with 8GB of RAM.**

component $c_i$. We refer to the type assignment for component $c_i$ as $t(c_i)$. The configuration agent has several actions at its disposal. It can select a next component to assign a type to. If it has already selected a component, it can assign a type to it. Hence, configuring a component is a two-step process (selecting the component and then assigning it a type). The agent also has an action of waiting, which leaves a partial configuration unchanged for the time step. However, at each time step until the configuration is completed and given to the user, the agent receives a penalty of $-0.2$; hence, there is a sense of urgency in completing the task, as waiting is costly.

The user has preferences over the assignment to each component that can be modeled in terms of reward function $R(t(c_i))$, and the reward for a complete configuration is the sum of the component reward functions $\sum_i^N R(t(c_i))$. The configuration agent, however, may not know the user's component reward functions, and represents its uncertainty as a probabilistic distribution over the components' possible reward functions. To reduce its uncertainty, the agent can pose to the user an *action*-query [4], which asks the user "What would you do?" given a partially specified machine, where the question may only be posed as asking which type to specify for a particular component. The user's response, which for our experiments comes after a deterministic delay of 2 time units, allows the configuration agent to improve its probabilistic model of the user's true component reward functions. Here we assume the user's response is deterministic in that the user is sure of the optimal action; we refer to our earlier work [4] for handling noise in the user's responses. Note that we have not included every possible state in the query space, as queries asking which part to configure next are useless since the user is indifferent as to in which order the parts are configured.

Hence, the configuration agent faces a decision about what query (if any) to pose, and what policy to follow while waiting for the response. Intuitively, the agent might ask about a component whose reward function it is least certain about. While awaiting a response, it might assign components about which it is reasonably certain. In the extreme case where it is sufficiently uncertain about any components, and the possible reward values are high enough, the agent might justifiably choose to wait for a response before making any assignment. (Note: We assume that once a component is assigned it cannot be changed.) Or, when query costs are high enough, the agent might make assignments as best as it can without asking any queries.

**Experiment Setup.** We compare five algorithms in the Machine Configuration domain. As a baseline, the first algorithm (Prior_Policy) does no querying, and simply follows the optimal policy given the agent's current model of the possible reward functions. The second algorithm (Optimal_Query_For_Prior_Policy) incorporates JQTP's technique of iterating through the query space to find a (myopically) optimal query given a transient policy, but again assumes the agent follows the prior policy (corresponding to acting according to Equation 4 with $\pi$=$\pi^*_\psi$). That is, Optimal_Query_For_Prior_Policy follows a transient policy that has not been formulated to complement the query that is being asked, but has a chance to update its policy according to the response of its query before the machine has been completely configured (since a minimum of 6 actions must be taken in total to complete the task). The third algorithm, Optimal_Query_For_Wait_Policy, is at the other extreme: it follows the "always wait" transient policy and optimizes the query for it. The fourth algorithm, JQTP, is the full implementation of JQTP as described in Section 3, and finds a joint query and transient policy that together are guaranteed to maximize expected (myopic) value (acting according to Equation 3). Finally, the fifth algorithm (JQTP_Clustering) is JQTP with posterior belief clustering, where we vary the $K$ parameter used in K-means to explore tradeoffs between computational cost and solution quality.

We use a small version of the Machine Configuration domain with query cost set to 0 for every query (varying query costs did not produce any interesting additional trends), discount set to 0.9, and $N$ and $M$ both set to 3. The algorithms were required to select a query at the start state (except for Prior_Policy), and were not allowed any additional queries. We measured the expected performance in terms of value accrued by each algorithm, where the expectation was estimated over 180 trials: trials differed by the priors used by the agents (though identical across the agents for a given trial). On each trial, 10 candidate settings of $R(t(c_i))$ were sampled, where the reward for each possible component type was sampled uniformly from $[0, 3]$, and each agent's prior was uniform over these 10 candidate settings. For each trial, values were measured as an expectation with respect to the trial's prior. Even though the candidate reward functions were sampled uniformly, the small number of samples biased the agent's priors towards expecting larger or smaller rewards for particular component types.

**Results.** Figure 1a shows our results for the above experiment. As expected, JQTP performs best since it is optimal in this setting where only one query may be asked and must be asked at the beginning. Somewhat surprisingly, Optimal_Query_For_Wait_Policy performs worst, even worse than electing not to query at all. This is due to the fact that taking two steps longer to configure the machine penalizes the agent more than it gains by asking the best query which will be received before committing to any component types. Optimal_Query_For_Prior_Policy performs better than Prior_Policy but worse than JQTP. This is because, as intuitively expected, JQTP queries to determine the user's preferences about components it is less certain about, and while waiting assigns components that it is more certain about. In contrast, Optimal_Query_For_Prior_Policy assigns components in a rigid order (unaffected by querying) and so decides on a query around this order, meaning that sometimes it asks a query that is less useful because an answer to a more useful query would be received after the corresponding component would already be assigned. These results show that in this domain, optimal transient behavior (paired with its optimal query) lies somewhere in between the two extremes of always waiting for a response before taking irreversible actions and never waiting.

Lastly, JQTP_Clustering displays improvement as the number of clusters used in K-means is increased (as it makes finer distinctions between queries' results), approaching the performance of JQTP after $K = 20$. This means it is able to perform nearly as well as JQTP by evaluating only 20 posterior beliefs per reachable state, as opposed to JQTP's evaluation of all (382) possible posterior beliefs per reachable state. The computational implications are shown in Figure 1b: in this domain, on average, JQTP_Clustering can perform nearly as well as JQTP without clustering with only approximately $30\%$ of the computation. Note that although the computation time shown includes that of performing the clustering step, its impact is minimal.

These positive results can be explained by noticing that the optimal number of clusters in this problem is 10: there are 10 distinct posteriors possible after asking a single query (3 distinct posteriors are possible for each query asking about one of the three components, and 1 posterior equaling the prior is possible if the user were to respond with "wait"). Clustering allows JQTP to avoid computing redundant values, and further shows a graceful degradation in performance as less similar posteriors are grouped. Overall, these results suggest that clustering posterior beliefs shows promise as an effective means for flexibly balancing performance and computation in domains with time-delayed query responses.

## 5. RELATED WORK

Our previous work ([5], [4]) and similar work [6] deal with planning under different forms of query and environment uncertainty. These methods can be seen as bridging Bayesian Reinforcement Learning [12] and POMDP Planning [7] whereby the methods exploit structured uncertainty and observations. However, they do not consider the temporal dynamics and concurrency induced by delays in receiving query responses as we do in this work.

Temporal abstraction and concurrency are issues that have been addressed together before in Hierarchical Reinforcement Learning [2] and other similar work [14], but these methods do not consider environment model uncertainty explicitly, and thus do not exploit the factored relationship between physical state and knowledge state with respect to queries and actions, as JQTP does.

Our setting can be viewed as a degenerate multiagent system in which one agent (the operator, which we represented by the reward-knowledge process) has a fixed policy (i.e., when the agent queries $q$, from the agent's viewpoint the operator responds according to $H_q(\psi'|\psi)$ after a stochastic delay). This structured interaction allowed us to develop a more scalable solution than solutions developed for similar problems explored in the rich multiagent communication literature, where in the typical case agents may communicate their observation histories (or portions of them). In particular, Spaan et al. [11] analyzed multiagent communication under state uncertainty with stochastic communication delays, but assumed that communication involves exchanging entire histories of observations, eliminating the *what* to query question that we address. Further, their approach assumes that communication is either delayed by one time step or fails entirely, while our approach allows for any bounded communication delay distribution. Roth et al. [9] provide an algorithm for the *what* to query problem in a similar setting to Spaan et al. [11], but do not account for stochastic communication delays. Thus, our setting can be roughly viewed as the intersection of these settings generalized to allow for arbitrary communication effects, but specialized to the case in which uncertainty is only in rewards, and only two agents are present, with the query-answering agent being a degenerate agent whose state merely consists of the duration of time since the currently outstanding query was asked, allowing for a more scalable solution.

A potentially interesting alternative way of framing our problem in the special case of single time step response delay is as a transition-independent Dec-MDP [3], in which one agent controls the CMP and the other agent controls the reward-knowledge process, and the joint reward function depends on the states of both. Whether algorithms for solving transition independent Dec-MDPs can be fruitfully adapted to our setting is a subject we leave for future work.

Lastly, [1] have explored clustering observations in POMDPs. In their work, the aim is to reduce the size of a large observation set offline, which they accomplish by clustering observations according to their emission probabilities across all possible states. Our clustering approach, while related in that we aim to reduce the number of possible posterior beliefs that the agent reasons over, instead clusters possible posterior beliefs the agent might encounter online during the planning stage. Further, our setting is more specialized in that the agent's uncertainty is in the reward function rather than an arbitrary state space, which we exploit by clustering posterior beliefs according to the distance between their induced mean reward functions.

## 6. CONCLUSION AND FUTURE WORK

In this paper we investigated the problem of how an agent can act optimally when it has reward uncertainty but can query its human operator to receive a response (after a stochastic delay) that reduces its uncertainty. We formulated a solution to the problem as a joint optimization over the space of queries and transient policies, and provided a Joint Query and Transient Policy (JQTP) algorithm that optimally solves the myopic approximation of this problem by exploiting factoring between the agent's knowledge state and physical state with respect to queries asked and actions taken. We then proposed a clustering algorithm for flexibly reducing the computation required by JQTP in exchange for performance loss, and presented empirical results showing potential benefits of JQTP and that the clustering algorithm can effectively reduce computation while incurring only small performance loss relative to JQTP.

Our ongoing work is in two directions. First, we are investigating the utility of leveraging further domain structure in order to more effectively cluster query-response pairs, using ideas of influence-based abstractions [16]. Second, we are looking to apply our method to a physical robot and human operator situation. Our long-term efforts are to extend our framework to multiagent sequential decision problems involving more querying agents (vying for the attention of a single human/agent responder), and hence where the policy of the responder (over how to prioritize queries) should be optimized jointly with the policies of the querying agents. Lastly, it would be interesting to explore other forms of uncertainty (such as in the transition function) in the context of delayed-response queries.

*Acknowledgments.*

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

## 7. REFERENCES

[1] A. Atrash. Efficient planning and tracking in pomdps with large observation spaces.

[2] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. 13:2003, 2003.

[3] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-Independent Decentralized Markov Decision Processes. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 41–48, Melbourne, Australia, July 2003. ACM Press.

[4] R. Cohn, E. H. Durfee, and S. Singh. Comparing action-query strategies in semi-autonomous agents. In *AAAI'11*, pages –1–1, 2011.

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

[6] F. Doshi and J. Pineau. Reinforcement learning with limited reinforcement: Using bayes risk for active learning in pomdps. In *ICML*, 2008.

[7] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large pomdps. *Journal of Artificial Intelligence Research*, 27:2006, 2006.

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

[9] M. Roth, R. Simmons, and M. Veloso. What to communicate? execution-time decision in multi-agent pomdps. In *The 8th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2006.

[10] S. P. Singh and R. C. Yee. An upper bound on the loss from approximate optimal-value functions. In *Machine Learning*, pages 227–233, 1994.

[11] M. T. J. Spaan, F. A. Oliehoek, and N. Vlassis. Multiagent planning under uncertainty with stochastic communication delays. In *Proc. of Int. Conf. on Automated Planning and Scheduling*, pages 338–345, 2008.

[12] M. Strens. A bayesian framework for reinforcement learning. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 943–950. ICML, 2000.

[13] R. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[14] D. S. Weld. Planning with durative actions in stochastic domains.

[15] R. Williams and L. C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, 1993.

[16] S. J. Witwicki and E. H. Durfee. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *ICAPS-2010*, pages 185–192, Toronto, Canada, May 2010.

## APPENDIX

Here we show the derivation of Equation 3, which represents the expected discounted sum of mean-rewards of the agent which asks query $q$, follows transient policy $\pi$, and then acts according to its new knowledge state without asking any more queries. Let the random time at which the query returns be $\tau$. Let $\{s_t, a_t\}$ be the infinite random sequence of state-action pairs generated by following transient policy $\pi$ until the query returns and following the optimal decision-making policy thereafter, and let $\{\psi_t\}$ be the infinite random sequence of knowledge states the agent encounters given that it queries $q$ and then asks no more queries. Note that $\{\psi_t\}$ is $\psi$ until the (random) time step at which the query returns after which point it is some (random) $\psi'$. Then,

$$\begin{aligned}
&Q_\psi(s,q,\pi)\\
&\triangleq c(q)\\
&+\mathbb{E}_{\{s_t,a_t\},\{\psi_t\}}[\sum_{t=0}^{\infty}\gamma^t\overline{R}_{\psi_t}(s_t,a_t)|s_0=s,\psi_0=\psi,q,\pi]\\
&=c(q)\\
&+\sum_{\tau=1}^{\tau_{\max}}F_q(\tau)\mathbb{E}_{\{s_t,a_t\}}[\sum_{t=0}^{\tau-1}\gamma^t\overline{R}_{\psi_t}(s_t,a_t)|s_0=s,\psi_0=\psi,q,\pi]\\
&+\sum_{\tau=1}^{\tau_{\max}}F_q(\tau)\mathbb{E}_{\{s_t,a_t\},\psi'}[\sum_{t=\tau}^{\infty}\gamma^t\overline{R}_{\psi_t}(s_t,a_t)|s_0=s,\psi_0=\psi,q,\pi^*_{\psi'}].
\end{aligned}$$

The second term is $r^q_\pi(s)$, defined in the main text. The third term represents the rewards collected by the agent after $q$ has been returned, at which point it updates its knowledge state to $\psi'$, never to be updated again (the myopic assumption), and begins selecting actions and queries according to $\pi^*_{\psi'}$:

$$\sum_{\tau=1}^{\tau_{\max}} F_q(\tau)\mathbb{E}_{\{s_t,a_t\},\psi'}\Big[\sum_{t=\tau}^{\infty}\gamma^t \overline{R}_{\psi'}(s_t,a_t)|s_0=s,q,\psi_0=\psi,\pi^*_{\psi'}\Big]$$
$$= \sum_{\tau=1}^{\tau_{\max}} F_q(\tau) \sum_{\psi'\in\Psi(\psi,q)} H_q(\psi'|\psi)\mathbb{E}_{\{s_t,a_t\}}$$
$$\Big[\sum_{t=\tau}^{\infty}\gamma^t \overline{R}_{\psi_t}(s_t,a_t)|s_0=s,\pi^*_{\psi'}\Big]$$
$$= \sum_{s'\in S} T^q(s'|s,\pi) \sum_{\psi'\in\Psi(\psi,q)} H_q(\psi'|\psi)\mathbb{E}_{\{s_t,a_t\}}$$
$$\Big[\sum_{t=\tau}^{\infty}\gamma^{t-\tau} \overline{R}_{\psi'}(s_t,a_t)|s_\tau=s',\pi^*_{\psi'}\Big]$$

where $T^q(s'|s,\pi) \triangleq \sum_{\tau=1}^{\tau_{\max}} F_q(\tau)\gamma^\tau P(s'|s,\pi,\tau)$,

and where $P(s'|s,\pi,\tau)$is the probability that the state $\tau$ steps after the non-stationary policy $\pi$ starts in state $s$ is $s'$. Now,

$$\sum_{s'\in S} T^q(s'|s,\pi) \sum_{\psi'\in\Psi(\psi,q)} H_q(\psi'|\psi)\mathbb{E}_{\{s_t,a_t\}}$$
$$\Big[\sum_{t=\tau}^{\infty}\gamma^{t-\tau} \overline{R}_{\psi'}(s_t,a_t)|s_\tau=s',\pi^*_{\psi'}\Big]$$
$$= \sum_{s'\in S} T^q(s'|s,\pi) \sum_{\psi'\in\Psi(\psi,q)} H_q(\psi'|\psi)V^*_{\psi'}(s'),$$

and so finally we have

$$Q_\psi(s,q,\pi) = c(q) + r^q_\psi(s,\pi) +$$
$$\sum_{s'\in S} T^q(s'|s,\pi) \sum_{\psi'\in\Psi(\psi,q)} H_q(\psi'|\psi)V^*_{\psi'}(s').$$