Greedy Algorithms for Sequential Sensing Decisions

Hannaneh Hajishirzi **Afsaneh Shirazi** **Jaesik Choi** **Eyal Amir**
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{hajishir, hajiamin, jaesik, eyal}@illinois.edu

Abstract

In many real-world situations we are charged with detecting change as soon as possible. Important examples include detecting medical conditions, detecting security breaches, and updating caches of distributed databases. In those situations, sensing can be expensive, but it is also important to detect change in a timely manner.

In this paper we present tractable greedy algorithms and prove that they solve this decision problem either optimally or approximate the optimal solution in many cases. Our problem model is a POMDP that includes a cost for *sensing*, a cost for *delayed* detection, a reward for *successful detection*, and no-cost partial observations. Making optimal decisions is difficult in general. We show that our tractable greedy approach finds optimal policies for sensing both a single variable and multiple correlated variables. Further, we provide approximations for the optimal solution to multiple hidden or observed variables per step. Our algorithms outperform previous algorithms in experiments over simulated data and live Wikipedia WWW pages.

1 Introduction

Humans and software systems may need to *detect change* as soon as it occurs. Often timely detection is critical in dynamic, uncertain environments because it enables focused and timely action. Unfortunately, many times information comes with a cost, so *sensing* (testing, checking, or retrieving information) at every step is not feasible. They must decide what information to sense and when. An example that motivates this research is the problem of detecting the changes of web pages. For example, some pages include information about a group of companies. This information may affect the stock value of the corresponding companies.

In this paper we formalize the problem of deciding when and what to sense and provide efficient greedy solutions for it. We assume that we are given a dynamic model formulated as a Dynamic Bayesian Network (DBN) [Jensen *et al.*, 1990]or an HMM [Rabiner, 1989], and also a cost model for sensing variables. We show that this problem can be modeled with Partially Observable Markov Decision Processes (POMDPs) [Littman, 1996]. POMDPs model partially observable stochastic systems, and include cost and rewards for actions and states. Unfortunately, finding optimal policies in POMDPs is difficult [Kaelbling *et al.*, 1998]. Exact algorithms for solving POMDPs take exponential time in the number of time steps considered by the system [Kearns *et al.*, 2000; Littman, 1996; Even-Dar *et al.*, 2005], or super-exponential time in the size of the space [Jaakkola *et al.*, 1994]. Approximate algorithms [Murphy, 2000] are also usually intractable for large state spaces (but see point-based algorithms [Spaan and Vlassis, 2005]). In this paper we provide constant time (after some offline precomputation) greedy algorithms to find the optimal policy for single variable and an approximation to the optimal policy for multiple hidden or observable variables per step.

The main contribution of this paper is to show that our decision problem is solvable by a greedy approach in important cases. We provide tractable algorithms (that find optimal or near-optimal solutions to such POMDPs) and an empirical evaluation of their speed and accuracy. First (Section 3), we present tractable optimal algorithms for the single-variable sensing problem. Second (Section 4), we generalize this to having multiple hidden or observed variables in each state. Finally (Section 5), we generalize these scenarios to multiple sensing-decision variables (e.g., this includes *factorial HMMs* in which we can sense part of the hidden state for a cost). There, we show that a more general greedy approach is optimal among policy chains which are a subclass of all policies. We then show our algorithm has a better performance than two leading POMDP algorithms for both simulated and real world data (Wikipedia WWW pages).

Variants of our problem were discussed before. In the WWW literature, efficient scheduling for search engine crawlers has received considerable attention (e.g. [Cho, 2001; Pandey *et al.*, 2004; Cho and Garcia-Molina, 2003]). They assume that web page changes are independent, and do not take observations into account.

In monitoring anytime algorithms [Hansen and Zilberstein, 2001; Finkelstein and Markovitch, 2001], the goal is to determine an optimal sensing schedule that minimizes execution time and there is a cost associated with the time it takes to interrupt the algorithm and check whether the solution quality is acceptable. Again, these approaches do not allow additional observations, and it is assumed that only one variable

is monitored at all times.

Finally, the multi-armed bandit problem [Robbins, 1952; Auer *et al.*, 2000] is used to model exploration decisions in stateless domains. There, an agent tries to maximize its overall reward by playing one of n different slot machines at every time step, with the initial payoffs unknown. The problem is *learning* the payoffs vs. trying to maximize rewards for machines where the payoffs have been estimated already. In our setting, we assume that dynamics (reward and transition model) are known and focus on devising optimal sensing schedules in a partially observed dynamic domain.

2 Sequential Sensing Decisions

In this section, we model a general sensing decision problem using a POMDP. Details of the formulation for specific scenarios are given in later sections. In general, we assume a discrete-time environment which evolves stochastically with time. At each time step t the agent decides to sense (or not) a sensing object and receives a reward that depends on the agent's action and the current state of the world. We model this decision problem with a POMDP because it allows modeling the effect of sensing on knowledge and the effect of knowledge on decisions. We use this specialized POMDP because it emphasizes our problem's special structure and makes change explicit. Later, we take advantage of the features of this POMDP and introduce efficient algorithms to find optimal solutions. Thus, part of our contribution here is the POMDP structure introduced for this type of problems.

In general, every POMDP is a tuple $\mathcal{M} = \langle \mathbf{X}, A, T, O, R \rangle$, which includes a state space, $\mathbf{X}$, a set of actions, A, a probabilistic transition model, T, an observation model, O, and a reward function, R. At time step $t-1$, the agent executes a chosen action, a^{t-1}, and goes from state s^{t-1} to s^t according to the transition model. Then, it receives an observation o^t and a reward $R(s^{t-1}, a^{t-1})$. The agent cannot observe s^t directly, but can maintain a *belief state* $b^t(s^t) = P(s^t | o^{1:t}, a^{1:t-1}, b^0)$, where b^0 is the belief state at time 0. The agent's goal is to devise a policy that maximizes the discounted sum of expected rewards.

We define POMDP components that model our problem. The environment of interest is represented by a state space $\mathbf{X}$ which is the union of a set of random variables. The variables defining the state space are factored into four disjoint sets $\mathbf{X}_h, \mathbf{X}_o, \mathbf{X}_s$, where

- $\mathbf{X}_h$ is a set of *completely hidden variables* whose true values can never be observed by the agent. They can represent some properties of the system.
- $\mathbf{X}_o$ is a set of *completely observable variables* whose true values are always observed by the agent.
- $\mathbf{X}_s$ is a set of *sensing variables* whose true value can be observed depending on the action. Every sensing variable C is boolean, representing whether or not a certain property changed since the last time we observed it.

The full set of possible deterministic actions is $\mathcal{A} = \mathcal{A}_s \cup \{idle\}$ in which $\mathcal{A}_s$ includes all sensing actions. Action $idle$ does nothing. Each $a_i \in \mathcal{A}_s$ is associated with a set of variables $\mathbf{X}_i \subseteq \mathbf{X}_s$, that are *sensed* by a_i (observed in the result of performing a_i).

The state of the environment evolves by a stationary Markovian transition model $P(\mathbf{X}^t | \mathbf{X}^{t-1}, a^{t-1})$. The observation model is deterministic, i.e., for action *idle* the agent always observes $\mathbf{X}_o$, and for sensing action a_i it observes $\mathbf{X}_i \cup \mathbf{X}_o$.

The reward function, R, enforces our goal to detect change as quickly as possible with a minimal number of costly sensing actions. If the agent executes a sensing action, it receives a positive reward when change has occurred and a negative reward otherwise. The agent also receives a penalty (negative reward) for staying idle while the change has occurred.

Usually, a policy is represented indirectly in terms of a *value function* $V(b)$ which represents how desirable it is for the agent to be in a particular belief state. The optimal action in belief state $b(s)$ is then chosen as the one that maximizes $V(b)$ which is the sum of immediate rewards plus expected future rewards.

Given this formulation, we can solve the sensing decision problem using general POMDP algorithms, e.g., [Cassandra *et al.*, 1994; Spaan and Vlassis, 2005]. Unfortunately, POMDP algorithms are intractable for environments larger than a few dozen states and a large time horizon (see [Littman, 1996] for a summary of complexity results, and [Murphy, 2000] for an overview of approximate solutions). Recently, point-based methods (e.g., [Spaan and Vlassis, 2005], [Poupart, 2005]) provide good approximation for POMDPs optimal policy using sampling. In this paper, we look for efficient algorithms that take advantage of the special structure of our problem. In what follows, we present greedy algorithms to find either the optimal policy or an approximation to the optimal policy and list some cases for which these greedy algorithms work.

3 Sequential Decisions for a Single Object

In this section we provide an efficient algorithm for detecting change of a single sensing object. Our algorithm uses a greedy method to determine whether to sense the object or not at each time step. Every sensing action on the sensing object determines whether a change occurs in the value of the object. We prove that the greedy algorithm provides an optimal policy in our POMDP.

The POMDP model constructed for a single sensing object has two variables in its state space $\mathbf{X}$. (1) A binary variable $G \in \mathbf{X}_h$ (completely hidden variable) whose value indicates whether a change has occurred at that time or not. (2) A binary sensing variable $C \in \mathbf{X}_s$ (sensing with a cost) whose value indicates whether a change has occurred since the last sensing action. The graphical structure of our POMDP model for a single variable is represented with a Dynamic Decision Network (DDN) [Russell and Norvig, 2003] in Figure 1, *left*. The probability of change of the sensing object is modeled with $P(G^t)$. G^t is independent and identically distributed for $t \geq 0$. The value of C is a deterministic function of the value of G and the set of actions performed up to time t (the value of C becomes 1 the first time the change occurs and it becomes 0 when we sense the object). $P(C^t)$ is calculated as follows:

$$\begin{aligned} &P(C^t = v | a^{t-1} = sense, C^{t-1} = v', G^t = v) = 1 \\ &P(C^t = 1 | a^{t-1} = idle, C^{t-1} = 1, G^t = v) = 1 \\ &P(C^t = v | a^{t-1} = idle, C^{t-1} = 0, G^t = v) = 1 \end{aligned}$$

where, v and v' are either 0 or 1.

The reward function $R(s,a)$ of executing action a in state $s = \langle C = c, G = g\rangle$ is defined as follows. Note that $r > 0$ represents the *reward* of sensing the change, $e < 0$ represents the *cost* of sensing while there is no change and $p < 0$ represents the *penalty* of sensing late while there is a change.

$$R(\langle C = c, G = g\rangle, a) = \begin{cases} r & c = 1, a = sense; \\ e & c = 0, a = sense; \\ p & c = 1, a = idle; \\ 0 & c = 0, a = idle. \end{cases} \quad (1)$$

We calculate the value function of a belief state b by using the Bellman equation in an infinite horizon with discounted rewards. Note that a belief state $b(C = c, G = g)$ is defined as the probability of being in state $s = \langle C = c, G = g\rangle$.

$$V(b) = \max_a \left(\sum_{s\in S} b(s)R(s,a) + \gamma \sum_{b'} T(b,a,b')V(b') \right) \quad (2)$$

where $0 \leq \gamma < 1$ is the discount factor and $T(b,a,b')$ is the transition between belief states after executing action a. In this paper, V always refers to the optimal value function.

We use an equivalent formulation for the value function suitable to our problem. We unwind the recursive term of Formula (2) until the first action *sense* is executed, i.e., we compute the rewards iteratively up to the first action sense (time x). For simplicity of representation, we define an auxiliary variable ch, where $P(ch = t + x) = P(C^t = 0, \ldots, C^{t+x-1} = 0, C^{t+x} = 1)$ (i.e. ch denotes the time at which C switches from 0 to 1). Also, we define P_b of any belief state b and any event E as: $P_b(E) = \sum_s P(E|s)b(s)$.

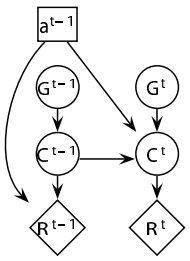
Proposition 3.1 *The value function is as follows:*

$$\begin{aligned} V(b) \;=\; & \max_x [P_b(C^t = 1)\ (\gamma^x r + (1 + \gamma \ldots + \gamma^{x-1})p) \\ & + P_b(t < ch \leq t + x)(\gamma^x r + \\ & \quad E(\gamma^{ch-t} + \ldots + \gamma^{x-1} | t < ch \leq t + x)\ p) \\ & + P_b(t + x < ch)\ (\gamma^x e) \\ & + \gamma^{x+1} V(b^*)] \;=\; \max_x f(x) \end{aligned} \quad (3)$$

where b^* is the belief state of the system after executing action sense i.e., $b^*(C = v, G = v')$ is equal to $P(G = v')$ if $v = v'$ and 0 otherwise.

We interpret the formula in Proposition 3.1 as follows: The first term of this formula states that a change has occurred before current time and we get a penalty for sensing it late in addition to positive reward r. The second term is the expected reward when the change occurs after the current time but before sensing. The third term corresponds to when change occurs after sensing so we have to pay the cost for making a sensing action in error. Finally the last term is the recursive value for actions that we perform after time $t + x$.

The algorithm (Multiple Sense Multiple Change (MSMC)) for finding the optimal policy is sketched in Figure 1. The algorithm checks at each time step if there is a decrease in f function by evaluating $d = f(1) - f(0)$. It then senses the object if $d < 0$. Notice that the algorithm replaces expensive evaluation of $V(b)$ with a much cheaper evaluation of d.



PROCEDURE MSMC($P_b(C^t)$)
$P_b(C^t)$ part of the current belief state, $P_b(C^{t+1})$ computed by prediction.
1. $d \leftarrow f(1) - f(0)$ (Formula (4))
2. **if** $d < 0$ **then** return *sense*
3. **else** return *idle*

Figure 1: *(left)* DDN representing the transition model for one sensing variable C. G indicates whether a change occurs at this time. C represents whether a change has occurred since the last sensing actions. a is an action node, and R is the reward. *(right)* Algorithm for deciding when to sense in case of a single sensing object.

PROCEDURE FPBSTAR()
1. $v_0 \leftarrow 0, v_1 \leftarrow$ VALUE(v_0)
2. **while** ($|v_1 - v_0| > \epsilon$)
   (a) $v_0 \leftarrow v1, v_1 \leftarrow$ VALUE(v_0)
3. return v_1

PROCEDURE VALUE(v_0)
1. $f \leftarrow -\infty, x \leftarrow 0$
2. **do**
   (a) $prevF \leftarrow f$
   (b) $f \leftarrow P(G = 1)\ (\gamma^x r + (1 + \gamma \ldots + \gamma^{x-1})p)$
   $+ \sum_{j=1}^{x} P(G = 0)^j P(G = 1)(\gamma^x r +$
   $(\gamma^j \ldots + \gamma^{x-1})p)$
   $+ (1 - \sum_{j=0}^{x} P(G = 0)^j P(G = 1))\gamma^x e + \gamma^{x+1} v_0$
   (c) $x \leftarrow x + 1$
   while ($f > prevF$)
3. return $prevF$

Figure 2: Algorithm for computing $V(b^*)$, the fixed point of the value function for belief b^*.

Computing $V(b)$ requires an x-step progression while computing d just requires a 1-step progression as shown in the following:

$$\begin{aligned} f(1) - f(0) = & P_b(C^t = 1)\ ((\gamma - 1)r + p) \\ & + P_b(C^t = 0, C^{t+1} = 1)(\gamma r - e) \\ & + P_b(C^t = 0, C^{t+1} = 0)\ (\gamma - 1)e \\ & + \gamma(\gamma - 1)V(b^*) \end{aligned} \quad (4)$$

We compute $V(b^*)$ of Formula (4) as a preprocessing step using a simple value iteration algorithm (Figure 2). This way, we compute $V(b^*)$ once and use it in making decision for every time step. Note that b^* is a very special belief state (sensing action has just occurred), the sensing object changes with prior probability $P(G = 1)$. Essentially, computing $V(b^*)$ is like performing value iteration in MDPs as we do not consider the effect of observations here. Moreover, the inner loop of computing $V(b^*)$ in Figure 2 is polynomial.

The rest of this section verifies the correctness of procedure MSMC (Figure 1, *right*). First, we show that the probability of $ch = t + x$ is a decreasing function of x. Using this we prove that $f(x)$ has just one local maximum. Finally we prove that, to obtain the optimal policy, we decide whether to sense or not at each time step based on just one comparison.

Lemma 3.2 *Let $P_b(ch = t+x)$ be the probability of change at time $t+x$, while the current time step is t. Then, $P_b(ch = t+x) \leq P_b(ch = t+y)$ for $x \geq y$.*

PROOF c^t and $\neg c^t$ stand for $C^t = 1$ and $C^t = 0$ respectively.

$$P_b(ch = t+x) = P_b(\neg c^t, \ldots, \neg c^{t+x-1}, c^{t+x}) = P_b(\neg c^t)\, P(\neg g^t)^{x-1}\, P(g^t)$$

Since $x \geq y$, $P_b(ch = t+x) \leq P_b(ch = t+y)$. ■

Theorem 3.3 *Let $V(b) = \max_x f(x)$ be the value function for the optimal policy. If $f(1) \leq f(0)$ then $f(x) \leq f(0)$ for $x \geq 0$.*

PROOF See appendix for the proof. ■

Corollary 3.4 *Let $V(b) = \max_x f(x)$ be the value function of the optimal policy. The optimal policy is to sense now, if and only if $f(0) \geq f(1)$.*

PROOF

1. Forward direction: if $f(0) < f(1)$, we just postpone sensing until the next time step.
2. Backward direction: if $f(0) \geq f(1)$, by Theorem 3.3, sensing at t has the greatest value. ■

4 Single Object with Observed and Hidden Variables

In this section we provide a similar greedy algorithm when there is a single sensing object and many hidden or completely observed variables in the domain. For example, there are multiple web pages that include finance-related information about companies. Updates to the information about one company may affect other companies (*e.g.*, an article about Blockbuster may indicate that there might be a change in the page about Netflix as well). Later, we show the application in Wikipedia where the update in the "Democratic party" page indicates a potential update in the "Barak Obama" page. We assume that observing "Democratic party" page has no cost while "Barak Obama" page is a sensing object and sensing that page is costly.

In this model, there is a single sensing variable $C \in \mathbf{X}_s$ and an arbitrary number of completely observed ($\mathbf{X}_o$) and hidden ($\mathbf{X}_h$) variables. The only constraint is that C in $\mathbf{X}_s$ does not have any parents in $\mathbf{X}_h$ and $\mathbf{X}_o$ except G as in Figure 3, *left*. A simple example of this model is an HMM.

A *policy tree* of a POMDP is a tree of depth t that specifies a complete t-step policy. Nodes are actions, the top node determines the immediate action to be taken. Edges are the resulting observation. We restrict our search to policy trees in which all the actions at the same depth level are the same regardless of the observations of that time step and call them *restricted policy trees*. In restricted policy trees the selection of the immediate action is independent of the effect of future observations and just depends on the current belief state. Note that the belief state is being updated at each time step based on the previous observations. Therefore, the selection of immediate actions at time t depends on the belief state given the observations up to time t.

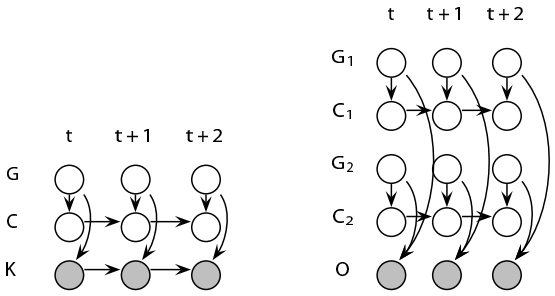


Figure 3: *(left)* Transition model for one sensing variable, C, with the existence of observations. K is a vector of both observed and hidden nodes. *(right)* Transition model for two sensing variables. C_1 and C_2 are sensing variables. O is the observation node.

The formulation for computing the value function $V(b)$ and $f(1) - f(0)$ are the same as in the previous section except that all the probabilities are posteriors given the observations up to time t. Belief state b is also the probability distribution over different assignments to state variables conditioned on the observations $o^{0:t} \in \mathbf{X}_o$.The optimal policy is the one that maximizes the value function (among restricted policy trees). Optimal restricted policy tree is a good approximation for the optimal policy as we show in our experiments (Section 6).

The following theorem shows that the probability of ch decreases as time passes. Therefore, just one comparison at each time is required to find the optimal restricted policy tree.

Theorem 4.1 *Let $P_b(ch = t+x|o^{0:t})$ be the probability of change at time $t+x$, while the current time step is t. Then, $P_b(ch = t+x|o^{0:t}) \leq P_b(ch = t+y|o^{0:t})$ for $x \geq y$.*

PROOF See appendix for the proof. ■

Same theorems as in the previous section exist for this case. Note that all the probabilities in this section are conditioned on the observations $o^{0:t} \in \mathbf{X}_o$.

5 Multiple Sensing Objects

This section provides a greedy algorithm for multiple sensing objects. There is one sensing variable C_i (can be observed by cost) and one hidden variable G_i for each sensing object (as in Section 3). There is no restriction on the number of observed variables in $\mathbf{X}_o$ (observed by no cost). The only constraint on the transition model is that each variable C_i^t has only two parents, C_i^{t-1} and G_i^t. The belief state is a distribution over different assignments to sensing variables C_i and G_i for $i \in \{0 \ldots n\}$ given the observations up to time t. Figure 3, *right* shows a graphical representation of this model.

One application of this model is checking the stock prices in the stock market to get information about participating companies. A sudden increase or decrease in the value of a stock of a company indicates the release of an important piece of information about the company. The stock value is used as an observation for the change in the web pages about the companies.

Our experiment is on Wikipedia (Section 6.2). We consider "Democratic party" page as the observation (sensing with no

PROCEDURE MOCHA($P_b(c^t|o^{0:t})$)
$P_b(c^t|o^{0:t})$ part of current belief state, n number of objects,
1. for all $0 \leq i \leq n$
   (a) $d_i \leftarrow f_i(1) - f_i(0)$
   (b) **if** $d_i < 0$ **then** return *sense* variable i

Figure 4: Algorithm for detecting change of multiple objects

cost) and two pages "Barak Obama" and "Hillary Clinton" as sensing objects (sensing with cost).

At each time step, the agent can choose a subset of objects to sense. We refer to this kind of actions as *composite actions*. Each composite action $a_{0:n}$ is a vector which represents that action $a_i \in \{sense, idle\}$ has been performed on the ith object. We assume that the reward function of a composite action $a_{0:n}$, $R(\langle c_{0:n}, g_{0:n}\rangle, a_{0:n})$, is the sum of the rewards of executing single actions on the corresponding sensing variables: $R(\langle c_{0:n}, g_{0:n}\rangle, a_{0:n}) = \sum_i R_i(\langle c_i, g_i\rangle, a_i)$, where $R_i(\langle c_i, g_i\rangle, a_i)$ is the reward function for single sensing variable with parameters r_i, e_i and p_i, as in Formula (1).

Again, the optimal restricted policy tree is the one that has the highest value among different restricted policy trees. Still, our algorithm finds the optimal restricted policy tree for each of the sensing variables, and then merges the results to find the optimal restricted composite policy. The algorithm is sketched in Figure 4. The rest of the section verifies the correctness of this algorithm.

The following development is presented with two sensing variables C_0 and C_1 for simplicity and clarity, but we can extend the results to more sensing variables easily. We use auxiliary variables ch_0 and ch_1 to show the time of the change for sensing variables C_0 and C_1 respectively.

Proposition 5.1 *The value function for belief state b is:*

$$
\begin{aligned}
V(b) = \max_{x,i}[& \\
& P_b(C_i^t = 1|o^{0:t})\ (\gamma^x r_i + (1+\gamma \ldots + \gamma^{x-1})p_i) && (5)\\
& + P_b(t < ch_i \leq t+x|o^{0:t})(\gamma^x r_i + && (6)\\
& \quad E(\gamma^{ch_i - t} + \ldots + \gamma^{x-1}|(t < ch_i \leq t+x|o^{0:t}))\ p_i) \\
& + P_b(t+x < ch_i|o^{0:t})\ (\gamma^x e_i) && (7)\\
& + P_b(C_{1-i}^t = 1|o^{0:t})\ (1+\gamma \ldots + \gamma^x)p_{1-i} && (8)\\
& + P_b(t < ch_{1-i} \leq t+x|o^{0:t}) && (9)\\
& \quad E(\gamma^{ch_{1-i} - t} + \ldots + \gamma^x|(t < ch_{1-i} \leq t+x|o^{0:t}))\ p_{1-i} \\
& + \gamma^{x+1} V(b^{x+1,i})] && (10)
\end{aligned}
$$

where $b^{x+1,i}(C_0^{t+x+1}, G_0^{t+x+1}, C_1^{t+x+1}, G_1^{t+x+1}) = P_b(C_0^{t+x+1}, G_0^{t+x+1}, C_1^{t+x+1}, G_1^{t+x+1}|a_i^{t+x} = sense, o^{0:t})$.

In this formula only one sensing object can be sensed at each time, but optimal policy may enforce sensing more than one object at a time. We can sense both objects (or any number of objects) by replacing (8), (9) with (5), (6), and (7) over variable ch_{1-i}. Then, in the recursive part the belief state is $V(b^{x+1,0,1})$. All the results remain the same.

Below we show that the value function for composite policies is the sum of value functions for single policies.

Theorem 5.2 *Let b be a probability distribution over C_0, G_0, C_1, G_1; let b_i ($i \in \{0,1\}$) be a probability distribution over C_i, G_i; and let $V_i(b_i)$ be the value function for single optimal policy given belief b_i; also let $V(b)$ be the value function for the optimal composite policy (Proposition 5.1). Then, $V(b) = V_0(b_0) + V_1(b_1)$.*

PROOF See appendix for the proof. ■

Like in previous sections, if we prove that $P_b(ch_i)$ is decreasing, then the value function for single policies has just one maximum which can be found by the greedy algorithm.

Lemma 5.3 *Let $P_b(ch_i = t+x|o^{0:t})$ be the probability that object i has changed for the first time after previous sensing at time $t+x$, while all the observations up to time t have been perceived. Then, $P_b(ch_i = t+x|o^{0:t}) \leq P_b(ch_i = t+y|o^{0:t})$ for $x \geq y$.*

PROOF See appendix for the proof. ■

Theorem 5.4 *Let $V_i(b_i) = max_x f_i(x)$be the value function for single policy in the case of multiple objects. The best policy is to sense variable i at time t iff $f_i(0) \geq f_i(1)$.*

PROOF By Theorem 5.2: $V(b) = V_0(b_0) + V_1(b_1)$. This value function achieves its maximum when both of its terms are at their maximum. Therefore, the best policy is to sense them at their unique maximum. Previous results show that if the value function for single sensing variable decreases, the function is at its maximum. Consequently, the best policy is to sense that variable at that time. ■

Like before, to avoid the direct computation of the value function, we calculate $f_i(1) - f_i(0)$ in Formula 11.

$$
\begin{aligned}
f_i(1) - f_i(0) = & P_b(C_i^t = 1|o^{0:t})\ ((\gamma-1)r + p) \quad (11)\\
& + P_b(C_i^t = 0, C_i^{t+1} = 1|o^{0:t})(\gamma r - e)\\
& + P_b(C_i^t = 0, C_i^{t+1} = 0|o^{0:t})(\gamma - 1)e\\
& + \gamma(\gamma-1)\ V(b_i^*)
\end{aligned}
$$

where $b_i^*(C_i, G_i) = P_b(C_i^t, G_i^t|a_i^{t-1} = sense)$. The function Multiple Objects CHAnge detection (MOCHA, Figure 4) greedily compares $f_i(1)$ with $f_i(0)$ and decides whether to sense the object or not.

6 Empirical Results

We evaluate our final algorithm (MOCHA) and compare it with two POMDP algorithms, Witness [Littman, 1996] and a point-based algorithm, Perseus[1] [Spaan and Vlassis, 2005]. We compare the efficiency and accuracy of these algorithms using simulated data in Section 6.1. We apply MOCHA to Wikipedia WWW pages and compare its accuracy to the other algorithms to illustrate their behavior on real-world data.

6.1 Simulation

We implement MOCHA (Figure 4) and test it on several randomly generated examples for each number of sensing objects. We also randomly generate an observation model, reward functions and a prior distribution, and build a DDN (Figure 3, *left*) for each example. We generate a sequence of size

[1] http://staff.science.uva.nl/ mtjspaan/software/approx/

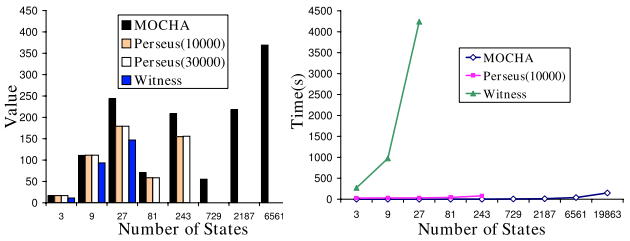


Figure 5: *(left)* Value (discounted sum of rewards) of the optimal policy vs. number of states. *(right)* Running time vs. number of states. *conclusion:* Our algorithm, MOCHA, returns higher value and is faster than Perseus (10000, 30000 samples), and Witness.

10^5 time steps for each experiment and calculate the value of the policy returned by MOCHA. We have 1 to 8 boolean variables (3 to 6561 states), 1 to 8 sensing actions, 1 idle action per time step.

We compare MOCHA with Witness and Perseus on this simulated data (transforming input DDNs into POMDPs). Witness is a classic POMDP algorithm and does not scale up to large state spaces. Perseus is a point-based algorithm which provides a good approximation for the optimal policy using sampling.

We report the accuracy and running time of our experiments over random examples in Figure 5. This figure shows that in all the experiments with more than 9 states the returned value for our method is always higher than values returned by Witness and Perseus. For each experiment we compute the value (discounted sum of rewards) of the policy returned (Perseus has experiments with 10000 and 30000 samples). Each bar in Figure 5, *left* displays the average value over all examples with the same number of states.

Notice that Witness is optimal for our problem when allowed an infinite tree depth. However, in practice we must limit Witness to a tractable number of steps so it is natural that the outcome is sub-optimal. In our case, we developed better formulas for our specific scenario so computation of those formulas is more precise and easier to estimate given the same time horizon. Also, in our experiments we run Witness only once per problem, not re-computing the policy tree after observations are presented. For that reason, in effect Witness's policy tree is shorter than ours after making an observation. This is a caveat in our evaluation method, and we can re-run the evaluation with a deeper tree for Witness, repeating the run of Witness after every observation. We limited the number of executions of witness for obvious reasons because it took a very long time to execute witness even for few states (1000 sec for 9 states).

Perseus returns the same value as ours for small problems with 3 and 9 states but it runs out of memory (uses more than 1Gb of memory) for state spaces larger than 243 states and crashes. MOCHA, on the other hand, requires less than 1Mb even for the largest state space (19863 states).

Figure 5, *right* shows the running time for computing the optimal policy for 10^5 time steps. It shows that MOCHA is faster than Witness and Perseus. Our preprocessing (computing $V(b^*)$) takes 0.01 seconds. Our algorithm takes less time than Witness for the same tree depth because our tree is pruned very aggressively on one side. Many subtrees are merged because they are all rooted in either $b^* = (G = 1, C = 1)$ or $b^* = (C = 0, G = 0)$. This way we save computation time whereas witness explores the entire tree.

This suggests that our approximation is better and faster than the state of the art POMDP algorithms.

6.2 Monitoring Wikipedia Pages

We use our algorithm to detect changes of Wikipedia pages while minimizing sensing effort and the penalty of delayed updates. Our approach is general and can be applied to any set of WWW pages. We compare our algorithm with both Witness and Perseus. Moreover, we compare the accuracy of these algorithms with the ground truth of changes of WWW pages.

In general, we build a set of factored-HMMs from a graph of nodes (*e.g.,* WWW pages), if the graph satisfies the following three conditions: (1) a set of nodes (VC) are always sensible with no additional cost; (2) VC is a vertex cover of the graph (vertex cover of a graph: a subset S of vertices such that each edge has at least one endpoint in S); and (3) the conditional probability of each node in VC can be represented by its adjacent nodes which are not in VC. Once the conditions are satisfied, each node in VC becomes a child node, and its adjacent nodes become parents of the node in the factored-HMM as shown in Figure 6, *left*. Thus, we can still solve the detect-change problem of the set of factored-HMMs obtained from a generic graph of WWW pages.

We gathered log data of 700 days for three Wikipedia pages ('Barack Obama', [2], 'Hilary Clinton' [3] and 'Democratic party' [4]). Since each page may change at any point of time, we need to discretize time first. We choose a one hour time step. We use the Democratic party page as an observation to estimate changes of the Obama page and the Clinton page. In this case, the Democratic party page is a vertex cover of the graph of three pages. We build a conditional probability table for the Democratic party page given the other two pages. The priors and conditionals for the Wikipedia pages are trained by counting of events.

Figure 6, *right* compares the value of MOCHA, Witness and Perseus for this domain. We evaluate the discounted sum of exact rewards achieved by these algorithms (r, p, e are $62, -4, -10$). We used these parameters for training to test in both cases. However, we believe that changing the parameters does not affect the result much.

Our algorithm outperforms both Witness and Perseus. In addition, we display the discounted sum of rewards of the ground truth of changes of WWW pages which knows the change of each page (call it oracle). The reward achieved by oracle is the maximum possible reward for the data set. Note that the results are not sensitive to input parameters (r, p, e). If we give different rewards, the overall rewards easily become negative numbers where our algorithm gave higher number than the others, though.

[2] http://en.wikipedia.org/wiki/Barack_Obama
[3] http://en.wikipedia.org/wiki/HillaryRodhamClinton
[4] http://en.wikipedia.org/wiki/Democratic_Party_United_States

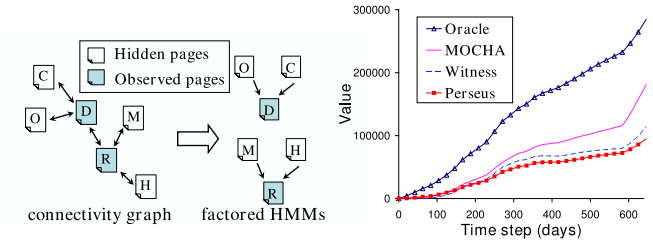


Figure 6: *(left)* Transforming the graph between WWW pages to factored HMMs for our algorithm. *(right)* Value vs. time of Obama and Clinton pages. MOCHA outperforms POMDP algorithms.

7 Conclusion & Future Work

We formalized the problem of detecting change in dynamic systems and showed how it can be cast as a POMDP. We suggested greedy algorithms for sensing decision in one or more variables. These algorithms use the structure of the problem to their advantage to compute a solution in time polynomial in the time horizon.

Our model that includes 0-cost sensing in addition to costly sensing is suitable for WWW sensing where there are WWW pages that are sensed automatically by designer decision. Such pages are, for example, news pages (NY Times, e.g.) and streaming media from websites and newsgroups. Other examples include pages with RSS feeds whose changes are detectable with very low cost. Our assumption about perfect sensing, while restrictive, is not far from many real-world systems. WWW pages and images captured by stationary cameras are good examples of those. In other cases, the perfect change-observation assumption is a good approximation, e.g. CT imaging (for detecting growth of tumors or pneumonia condition) outputs a 3D image which can be easily compared to previous images with only little noise.

There are two important limitation in our current approach. First, we do not know how far an approximation our method is from the oracle (sense at the exact same time that change occurs). Second, we cannot include actions that change the world (e.g., moving actions) in the model yet. The immediate direction for the future work is to fix these two limitations. Solving the problem of multiple objects when there is not any constraint on the transition relation between sensing variables can also be a future work. Another direction is to learn the model. The update frequencies (i.e. $P(G)$) and the observation model need to be learned. Choosing a relevant observation node is also an important part of the learning.

Acknowledgements

We would like to thank Anhai Doan and Benjamin Liebald for the insightful discussions and their help throughout the progress of this project. We also thank David Forsyth and the anonymous reviewers for their helpful comments. This work was supported by DARPA SRI 27-001253 (PLATO project), NSF CAREER 05-46663, and UIUC/NCSA AESIS 251024 grants.

References

[Auer *et al.*, 2000] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. *ECCC*, 7(68), 2000.

[Cassandra *et al.*, 1994] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, pages 1023–1028, 1994.

[Cho and Garcia-Molina, 2003] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Trans. Database Syst.*, 28(4), 2003.

[Cho, 2001] J. Cho. *Crawling the Web: Discovery and Maintenance of Large-Scale Web Data*. PhD thesis, 2001.

[Even-Dar *et al.*, 2005] E. Even-Dar, S. M. Kakade, and Y. Mansour. Reinforcement learning in POMDPs. In *IJCAI '05*, 2005.

[Finkelstein and Markovitch, 2001] L. Finkelstein and S. Markovitch. Optimal schedules for monitoring anytime algorithms. *Artif. Intell.*, 126(1-2):63–108, 2001.

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

[Jaakkola *et al.*, 1994] T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *NIPS*, volume 7, 1994.

[Jensen *et al.*, 1990] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

[Kaelbling *et al.*, 1998] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *AIJ*, 101:99–134, 1998.

[Kearns *et al.*, 2000] M. Kearns, Y. Mansour, and A. Y. Ng. Approximate planning in large pomdps via reusable trajectories. In *Proc. NIPS'99*, 2000.

[Littman, 1996] M. L. Littman. *Algorithms for sequential decision making*. PhD thesis, Brown U., 1996.

[Murphy, 2000] K. Murphy. A Survey of POMDP Solution Techniques. Unpublished Work, 2000.

[Pandey *et al.*, 2004] S. Pandey, K. Dhamdhere, and C. Olston. Wic: A general-purpose algorithm for monitoring web information sources. In *VLDB*, pages 360–371, 2004.

[Poupart, 2005] Pascal Poupart. *Exploiting Structure to Efficiently Solve Large Scale POMDPs*. PhD thesis, University of Toronto, 2005.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[Robbins, 1952] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 55, 1952.

[Russell and Norvig, 2003] S. Russell and P. Norvig. *Artificial Intelligence: Modern Approach Second Edition*. Prentice Hall, 2003.

[Spaan and Vlassis, 2005] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized point based value iteration for pomdps. *JAIR*, 24:195–220, 2005.

8 Appendix

Proof of Theorem 3.3
We prove by induction over x: Inductive assumption: $f(i) \leq f(0)$ for $i \leq x-1$. So once we prove $f(x) \leq f(x-1)$, we are done.

$$\begin{aligned} f(x) - f(x-1) = {} & P_b(ch \leq t+x-1)\, \gamma^{x-1}((\gamma-1)\kappa+\beta) \\ & + P_b(ch = t+x)\gamma^{x-1}(\gamma\kappa-\alpha) \\ & + P_b(t+x < ch)\, \gamma^{x-1}(\gamma-1)\alpha \\ & + \gamma^x(\gamma-1)\, V(b^*) \end{aligned}$$

By the assumption we know: $f(1) - f(0) \leq 0$.

$$\begin{aligned} f(1) - f(0) = {} & P_b(C^t = 1)\, ((\gamma-1)\kappa+\beta) \\ & + P_b(ch = t+1)(\gamma\kappa-\alpha) \\ & + P_b(t+1 < ch)\, (\gamma-1)\alpha \\ & + \gamma(\gamma-1)\, V(b^*) \leq 0 \end{aligned}$$

By Lemma 3.2, $P_b(ch = t+1) \geq P_b(ch = t+x)$. Also $P_b(t+x < ch) < P_b(t+1 < ch)$. $\gamma\kappa - \alpha$ and $(\gamma-1)\alpha$ are positive. So, the following inequivalency holds.

$$\begin{aligned} & f(x) - f(x-1) \\ & \leq P_b(t < ch \leq t+x-1)\, \gamma^{x-1}((\gamma-1)\kappa+\beta) \\ & \quad + P_b(ch \leq t)\, \gamma^{x-1}((\gamma-1)\kappa+\beta) \\ & \quad + P_b(ch = t+1)\gamma^{x-1}(\gamma\kappa-\alpha) \\ & \quad + P_b(t+1 < ch)\, \gamma^{x-1}(\gamma-1)\alpha \\ & \quad + \gamma^x(\gamma-1)\, V(b^*) \\ & = P_b(t < ch \leq t+x-1)\, \gamma^{x-1}((\gamma-1)\kappa+\beta) \\ & \quad + \gamma^{x-1}(f(1) - f(0)) \leq 0 \end{aligned}$$

Because both the first term and $f(1) - f(0)$ are negative numbers, and the proof is done. ■

Proof of Theorem 4.1
We use c^t and $\neg c^t$ as $C^t = 1$ and $C^t = 0$ respectively. In the following formulas k^t is the vector of all variables in $\mathbf{X}_h$ and $\mathbf{X}_e$ except G^t. According to Figure 3:

$$\begin{aligned} & P_b(ch = t+x, o^{0:t}) = P_b(\neg c^t, \ldots, \neg c^{t+x-1}, c^{t+x}, o^{0:t}) \\ & = \sum_{k^t \setminus o^t, k^{t+1:t+x}} P_b(k^t \setminus o^t, \neg c^t, o^{0:t}) \\ & \quad P_b(k^{t+1} | \neg c^{t+1}, \neg c^t, k^t \setminus o^t, o^t) P(\neg g^{t+1}) \ldots \\ & \quad P_b(k^{t+x} | c^{t+x}, \neg c^{t+x-1}, k^{t+x-1}) P(g^{t+x}) \end{aligned}$$

After a sequence of variable eliminations on $k^{t+1:t+x}$:

$$\begin{aligned} & P_b(ch = t+x, o^{0:t}) = P(g^{t+x}) P(\neg g^{t+x-1}) \ldots P(\neg g^{t+1}) \\ & \quad \sum_{k^t \setminus o^t} P_b(k^t \setminus o^t, \neg c^t, o^{0:t}) \end{aligned}$$

So $P_b(ch = t+x | o^{0:t}) \leq P_b(ch = t+y | o^{0:t})$ for $x \geq y$. ■

Proof of Theorem 5.2
Define $V^k(b)$ as the expected sum of future rewards of the optimal k-step policy (i.e., one in which we stop after k steps). $V_0^k(b_0)$ and $V_1^k(b_1)$ can be defined in the same way. Now we prove by induction that

$$V^k(b) = V_0^k(b_0) + V_1^k(b_1)$$

Induction basis: for $k = 0$ the formula holds because the expected sum of future rewards of the optimal 0-step policy is 0.
Now, assume by induction that $V^k(b) = V_0^k(b_0) + V_1^k(b_1)$ for $k \leq n$. Without loss of generality assume that $V^n(b)$ is maximized when we have to sense object 0 first. Using the induction hypothesis ($k - x - 1 < n$) we rewrite the recursive term as the sum of value functions of single policies:

$$V^{k-x-1}(b^{x+1,0}) = V_0^{k-x-1}(b_0^{x+1,0}) + V_1^{k-x-1}(b_1^{x+1,0})$$

Replacing the belief states as following yields to the proof:

$$\begin{aligned} b_0^{x+1,0}(s) &= P_b(C_0^{t+x+1} = v, G_0^{t+x+1} = v' | \\ &\qquad a_0^{t+x} = sense, o^{0:t}) \\ &= b_0^*(s) \\ b_1^{x+1,0}(s) &= P_b(C_1^{t+x+1}, G_1^{t+x+1} | a_0^{t+x} = sense, o^{0:t}) \\ &= P_b(C_1^{t+x+1}, G_1^{t+x+1} | o^{0:t}) \end{aligned}$$

$V^k(b) = \max$(expected sum of rewards of k-step policies). $V(b) = \max$(expected sum of rewards of k-step policies + $\gamma^k V(b')$).
There exists an M s.t. $|V(b')| < M$, so for any $0 < \varepsilon < 1$ there is a k s.t. $\gamma^k M < \varepsilon$. With the same argument there is a k s.t. $|V(b) - V^k(b)| < \varepsilon$, $|V_0(b) - V_0^k(b)| < \varepsilon$, and $|V_1(b) - V_1^k(b)| < \varepsilon$. Proof of $V^k(b) = V_0^k(b_0) + V_1^k(b_1)$ yields to the proof of the theorem by showing that: $|V(b) - [V_0(b_0) + V_1(b_1)]| < 3\varepsilon$. ■

Proof of Lemma 5.3
We use c^t and $\neg c^t$ in place of $C^t = 1$ and $C^t = 0$ respectively. According to Figure 3,

$$\begin{aligned} & P_b(ch_0 = t+x, o^{0:t}) = P_b(\neg c_0^t, \ldots, \neg c_0^{t+x-1}, c_0^{t+x}, o^{0:t}) \\ & = \sum_{C_1^{t:t+x}, G_1^{t+1:t+x}, o^{t+1:t+x}} P_b(C_1^t, \neg c_0^t, o^{0:t}) \\ & \quad P(o^{t+1} | G_1^{t+1}, \neg g_0^{t+1}) P(G_1^{t+1}) \\ & \quad P(\neg g_0^{t+1}) P(C_1^{t+1} | C_1^t, G_1^{t+1}) \ldots \\ & \quad P(o^{t+x} | G_1^{t+x}, g_0^{t+x}) P(G_1^{t+x}) \\ & \quad P(g_0^{t+x}) P(C_1^{t+x} | C_1^{t+x-1}, G_1^{t+x}) \end{aligned}$$

After performing a sequence of variable elimination:

$$\begin{aligned} & P_b(ch_0 = t+x, o^{0:t}) = \\ & \quad P(g_0^{t+x}) P(\neg g_0^{t+x-1}) \ldots P(\neg g_0^{t+1}) \sum_{C_1^t} P_b(C_1^t, \neg c_0^t, o^{0:t}) \end{aligned}$$

So $P_b(ch_0 = t+x | o^{0:t}) \leq P_b(ch_0 = t+y | o^{0:t})$ for $x \geq y$. ■