# Incremental methods for computing bounds in partially observable Markov decision processes

**Milos Hauskrecht**
MIT Laboratory for Computer Science, NE43-421
545 Technology Square
Cambridge, MA 02139
*milos@medg.lcs.mit.edu*

## Abstract

Partially observable Markov decision processes (POMDPs) allow one to model complex dynamic decision or control problems that include both action outcome uncertainty and imperfect observability. The control problem is formulated as a dynamic optimization problem with a value function combining costs or rewards from multiple steps. In this paper we propose, analyse and test various incremental methods for computing bounds on the value function for control problems with infinite discounted horizon criteria. The methods described and tested include novel incremental versions of grid-based linear interpolation method and simple lower bound method with Sondik's updates. Both of these can work with arbitrary points of the belief space and can be enhanced by various heuristic point selection strategies. Also introduced is a new method for computing an initial upper bound – the fast informed bound method. This method is able to improve significantly on the standard and commonly used upper bound computed by the MDP-based method. The quality of resulting bounds are tested on a maze navigation problem with 20 states, 6 actions and 8 observations.

## Introduction

Many real-world control or dynamic decision problems must deal with two sources of uncertainty. The first is uncertainty of the outcomes of actions, and the second is the control agent's imperfect ability to observe the underlying controlled process. In such problems there are usually more ways to explore the underlying process by means of various investigative actions and their outcomes. These can be more or less informative and can come with different costs. Thus in order to decide about the best action one needs to consider costs and benefits associated with both control and investigative actions. Problems with such characteristics include robot navigation (Littman, Cassandra and Kaelbling 95a), disease treatment (Hauskrecht 96) and various fault repair problems.

A framework that can model both sources of uncertainty and can represent both investigative and control actions and their associated costs is the partially observable Markov decision process (Astrom 65; Smallwood and Sondik 73; Lovejoy 91; Cassandra 94). A control or decision making problem within the framework is expressed as a dynamic optimization problem with a value function combining costs or rewards from multiple steps. Unfortunately the increased expressivity of the modelling framework is paid for by high computational demands for optimization. Although the problem of finding optimal control is computationally intractable, it is often possible to solve problems faster using various shortcuts or approximations. These methods can often benefit from the availability of good value function bounds.

We first describe the partially observable MDP framework, then describe and analyze incremental methods for computing upper and lower value function bounds. We illustrate the quality of these bounds on a robot maze navigation problem.

## Partially observable Markov decision process

A *partially observable Markov decision process (POMDP)* describes a stochastic control process with partially observable states and formally corresponds to a 6-tuple $(S, A, \Theta, T, O, R)$ where $S$ is a set of states; $A$ is a set of actions; $\Theta$ is a set of observations; $T : S \times A \times S \rightarrow [0,1]$ is a set of transition probabilities between states that describe the dynamic behavior of the modeled environment; $O : S \times A \times \Theta \rightarrow [0,1]$ stand for a set of observation probabilities that describe the relationship among observations, states and actions; and $R : S \times A \times S \rightarrow \mathcal{R}$ denotes a reward model that assigns rewards to state transitions and models payoffs associated with such transitions.

The *decision (or planning) problem* in the context of POMDP requires one to find an action or a sequence of actions for one or more information states that optimizes the objective reward function. An *information state* represents all information available to the agent at the decision time that is relevant to the selection of the optimal action. The information state consists of either a complete history of actions and observations or corresponding sufficient statistics ensuring the Markov property of the information process. A *value function* represents (quantifies) control objectives by combining rewards incurred over time using various kinds of models. Typically, the value function is additive over time and based on expectations, e.g. the function often uses a finite horizon model $\max E(\sum_{t=0}^{n} r_t)$, maximizing expected rewards for the next $n$ steps or an infinite discounted horizon $\max E(\sum_{t=0}^{\infty} \gamma^t r_t)$, with a discount factor $0 \leq \gamma < 1$.

We will focus on problems in which: 1. the value function uses the infinite discounted horizon criterion, 2. information state is sufficiently modeled by a *belief state*, which assigns a probability to every possible state, 3. the objective is to find a stationary control for all information states (policy problem).

## Solving the control problem

The optimal value function for the infinite horizon discounted problem and a belief state satisfies the standard fixed point formula:

$$V^*(b) = \max_{a \in A} \rho(b,a) + \gamma \sum_{o \in \Theta^+} P(o|b,a) V^*(\tau(b,o,a)) \quad (1)$$

where $V^*(.)$ is the optimal value function; $b$ denotes an $|S|$ dimensional belief state; $\rho(b,a)$ is the expected transition reward from state $b$ under action $a$ and can be computed as:

$$\rho(b,a) = \sum_{s \in S} \rho(s,a) b(s) = \sum_{s \in S} \sum_{s' \in S} R(s,a,s') P(s'|s,a) b(s);$$

$\Theta^+$ is a set of observations that are available in the next step; $\gamma$ is a discount factor; and $\tau$ is a transition function that maps the information state $b$, action and observation to the next belief state:

$$b^+(s') = \tau(b,o,a)(s') = (1/\beta) P(o|s',a) \sum_{s \in S} P(s'|s,a) b(s)$$

with $\beta$ normalizing the belief vector.

Once the optimal value function is known the optimal control function $\mu^*$ can be computed easily as:

$$\mu^*(b) = \text{argmax}_{a \in A} \rho(b,a) + \gamma \sum_{o \in \Theta^+} P(o|b,a) V^*(\tau(b,o,a)) \quad (2)$$

The equation 1 for an optimal value function can be rewritten by means of a value function mapping $H : B \to B$ with $B$ standing for bounded real valued functions, as: $HV^* = V^*$. The mapping $H$ is isotone and for $0 \leq \gamma < 1$, it is a contraction under the supremum norm. The major consequences of $H$ being a contraction are that by the Banach theorem there is a unique fixed point solution $V^*$ and that one can construct an iteration method with a step $V^{i+1} = HV^i$ that converges to it. Therefore the use of the standard iteration method would theoretically allow us to compute arbitarily close approximation of $V^*$ by performing a sufficient number of iterations. Such an approximation can then be used in equation 2 instead of the optimal value function.

The major problem with the iteration method is in computing value function updates within one iteration step for all possible belief states. Although hard and likely impossible in general (e.g., for models with time lags), it has turned out that when the information state is sufficiently modeled by a belief state, and when the value function $V^i$ is described by a finite, piecewise linear and convex function then $V^{i+1} = HV^i$ is computable and corresponds also to a finite, piecewise linear and convex function. This is based on the result in (Smallwood and Sondik 73) showing that the value function for a belief state can be computed as:

$$V^{i+1}(b) = \max_{a \in A} \sum_{s \in S} b(s) [\rho(s,a) + \sum_{o \in \Theta^+} \sum_{s' \in S} P(s',o|s,a) \alpha_i^{\iota(b,a,o)}(s')]$$

where $\iota(b,a,o)$ indexes a linear vector $\alpha_i$ in a set of linear vectors $\Gamma^i$ that maximizes:

$$\sum_{s' \in S} [\sum_{s \in S} P(s',o|s,a) b(s)] \alpha_i(s')$$

for fixed $b, a, o$. The linear vector that maximizes $V^{i+1}$ at $b$ then is:

$$\alpha_b^{i+1} = \text{argmax}_{\alpha_{b,a}^{i+1}} \sum_{s \in S} \alpha_{b,a}^{i+1}(s) b(s) \quad (3)$$

where $\alpha_{b,a}^{i+1}$ is the optimizing linear vector for action $a$:

$$\alpha_{b,a}^{i+1}(s) = \rho(s,a) + \sum_{o \in \Theta^+} \sum_{s' \in S} P(s',o|s,a) \alpha_i^{\iota(b,a,o)}(s')$$

Though the value function update is computable, the number of linear vectors describing the new improved value function can grow exponentially with the number of iterations, making the whole problem intractable. Moreover, the computation of a set of all useful linear segments of $V^{i+1}$ can be solved efficiently only when $RP = NP$ (Littman, Cassandra and Kaelbling 95b).

## The role of value function bounds

The computational inefficiency of exact value function updates as well as general $\epsilon$-optimal solutions leads naturally to the exploration of various approximations and shortcuts that can speed up exact methods or provide good control solutions with less computation. These are often built on the availability of good value function bounds or the ability to compute them fast.

Value function bounds and methods for computing them can be used within the POMDP framework in several ways. Bound methods can provide a good initial value function for the exact version of the value iteration algorithm, can be combined or interleaved with steps of exact methods for computing optimal or $\epsilon$-optimal solutions. For example bounds can be used to speed up the on-line decision (control) methods that compute an optimal or $\epsilon$-optimal control action for a specific belief state via forward expansion of the decision tree (Hauskrecht 96). Good value function bounds often reduce the size of the tree explored via branch

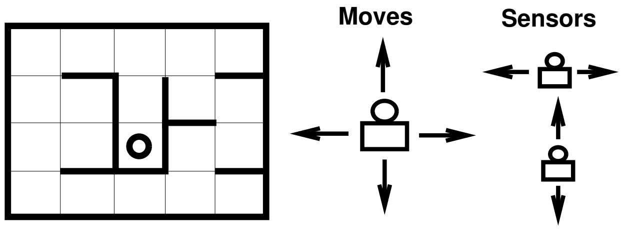


Figure 1: The robot navigation problem: Maze20

and bound strategies and allow one to cut the suboptimal action branches from the active tree.

Finally value function bounds can be used as good approximations of an optimal value function. Especially important with regard to control are suboptimal bounds that can guarantee minimal expected reward. However, the evaluation of the quality of various bound functions for approximating control and their comparison to alternative methods is a subject of another paper (Hauskrecht 97b).

## Test example: Maze20

For the purpose of testing and illustrating results, we have built a toy robot maze navigation problem with 20 states, 6 actions and 8 observations. The maze (figure 1) consists of 20 partially connected rooms (states) in which a robot functions and collects rewards. The robot moves in 4 directions (North, South, East and West) and checks for the presence of walls using sensors. Neither "move" actions nor sensor inputs are perfect. The robot moves in an unintended direction with probability of 0.3 (0.15 for each of the neighboring directions). A move into the wall keeps the robot on the same position. Investigative actions help the robot navigate by activating sensor inputs. There are 2 investigative actions: checking the presence of a wall in the North-South directions and checking of walls in the East-West directions. Sensor accuracy in detecting walls is 0.75 for a two wall case, 0.8 for a one wall case and 0.89 for a no wall case, with smaller probabilities for wrong perceptions.

The control objective is to maximize the expected discounted rewards with a discount factor of 0.9. A small reward is given for every action not leading to bumping into the wall (4 points for a move and 2 points for an investigative action), and one big reward (150 points) is given for achieving the special target room (shown as a circle on the figure). After reaching the goal state, the robot is placed with some probability into one of the 'initial" rooms.

## Computing upper bound

A standard method for computing an upper bound combines point interpolation techniques and value iteration strategy (see (Lovejoy 91) (Lovejoy 93)). In this method a value function is represented nonparametrically using a set of grid points together with their values and an interpolation rule that estimates the value at non-grid points with a convex combination of $|S|$ grid points. The values at grid points are updated (iterated) using the approximate update formula, i.e. any grid point $b$ defining $\widehat{V}^{i+1}$ is computed as:

$$\widehat{V}^{i+1}(b) = \max_{a \in A} \rho(b,a) + \gamma \sum_{o \in \Theta+} P(o|b,a)\widehat{V}^{i}(\tau(b,o,a))$$

The update of the complete function with a fixed grid can be expressed using value function mapping $H_{inter}$ as: $\widehat{V}^{i+1} = H_{inter}\widehat{V}^{i}$. It can be shown (see (Lovejoy 91)) that $H_{inter}$ is a contraction mapping that preserves the upper bound, i.e. starting from an initial upper bound every new value function in the iteration method is guaranteed to be an upper bound. As a consequence, value iteration with $H_{inter}$ mapping converges to a fixed point solution $\widehat{V}^{*} \geq V^{*}$.

The efficiency of the value function update depends strongly on the efficiency of the point interpolation rule. The interpolation rule must first select a set of $|S|$ points from the grid $G$ suitable for interpolation. In general there can be $\binom{|S|}{|G|}$ possible sets and finding the best interpolating set can be time-consuming. One possible solution to this is to use regular grids (Lovejoy 91), that evenly partition the belief space and allow one to choose interpolating set efficiently. However such grids must use a specific number of points and any increase in the resolution of a grid is paid for by an exponential increase in the grid size. For example for the Maze20 problem with 20 states the sequence of regular grids consists of $20, 210, 1540, 8855, 42504, \cdots$ grid points, which makes the method practically usable only for grids with small resolutions.

To provide more flexibility in grid selection, we designed a simple point interpolation method that allows arbitrary grids and is guaranteed to run in time linear in the size of the grid. The designed interpolation rule builds upon the fact that any point $b$ of the belief space of dimension $|S|$ and can be easily interpolated with a set of grid points that consists of an arbitary point $b' \in G$ and $|S|-1$ critical points of the belief simplex (critical points correspond to $(1,0,0,\cdots),(0,1,0,\cdots)$, etc.). In other words, for any grid point $b' \in G$ there is a simple interpolating set that allows one to compute a linear interpolation $\widehat{V}^{i}_{b'}(b)$ at an arbitrary point $b$. As any interpolation over the values of a convex function guarantees an upper bound, the tightest possible bound value for a set of grid points can be chosen, i.e.:

$$\widehat{V}^{i}(b) = \min_{b' \in G} \widehat{V}^{i}_{b'}(b).$$

The proposed interpolation rule can be computed in $O(|G||S|)$ time, which is linear in the size of grid. Although the method does not implement the optimal interpolation, we believe that its simplicity allows it to make use of a larger number of grid points. Moreover any increase in the grid size is very easy, and can be a basis for various efficient incremental strategies that gradually improve the upper bound. A simple incremental improvement algorithm is illustrated bellow. The algorithm starts from the initial upper bound $\widehat{V}_{init}$, expands the grid gradually in $k$ point increments, and uses Gauss-Seidel updates for points in the active grid. As the grid size is bounded by linear growth, the algorithm is guaranteed to run efficiently for a fixed number of iterations.

**Incremental upper bound** $(k, \widehat{V}_{init})$
- **select** an initial grid of points $G$
- **for every** point $b \in G$
  - **compute** $\widehat{V}_{init}(b)$ and store it in $\widehat{V}_G$ definition
- **repeat** until the stopping criterion is satisfied
  - **repeat** until the grid expansion criterion is met
    - **for every** point $b$ in $G$
      - **compute** new update $\widehat{V}(b)$ and
      - **update** the value in $\widehat{V}_G$
  - **select** a set of $k$ points $G_{EXP}$ to expand $G$
  - **for every** $b \in G_{EXP}$
    - **add** $b$ to $G$ and $\widehat{V}(b)$ to $\widehat{V}_G$

**return** $\widehat{V}_G$

The standard way to compute an initial bound $\widehat{V}_{init}$ is to use an MDP-based approximation. The method utilizes the optimal value function $V^*_{MDP}$ obtained for the given problem under the assumption of perfect observability. An upper bound on partially observable $V^*$ is then computed as:

$$\widehat{V}(b) = \sum_{s \in S} b(s) V^*_{MDP}(s)$$

## Selection of grid points

In general the quality of bounds produced by the incremental method is strongly influenced by the grid selection strategy. The advantage of our interpolation rule is that it does not rely on a specific grid (unlike regular grids). Thus it can be easily combined with an arbitrary selection method, which may include various heuristic strategies.

A heuristic method for selecting grid points we have designed, implemented and tested attempts to maximize improvements in bound values using stochastic simulations. The method builds on the fact that every grid suitable for interpolation must include critical points (otherwise the interpolation cannot be guaranteed). A value at any grid point $b$ improves more when more precise values are used for its successor belief states, i.e. belief states that correspond to $\tau(b, a, o)$ for an optimizing action $a$ and observation $o$. Incorporating such points into the grid would then increase the chance of larger improvement of values associated with critical points. Naturally one can proceed with selection further, by incorporating succesor points for the first level successors into the grid set as well, and so on. The stochastic simulation method tries to sample likely successor points by: 1. selecting an action $a$ that is optimal for $b$ given the current upper bound value function; 2. selecting the next observation randomly according to the probability distribution $p(o|b, a)$. In the context of POMDPs, a similar simulation method was used in (Parr and Russell 95; Littman, Cassandra and Kaelbling 95a)

| grid size | bound score | | | |
|---|---|---|---|---|
| | random | regular | heuristic-MDP | heuristic |
| MDP | 130.11 | 130.11 | 130.11 | 130.11 |
| 40 | 129.61 | - | 110.92 | 110.92 |
| 80 | 129.55 | - | 89.58 | 89.59 |
| 120 | 129.50 | - | 87.81 | 86.38 |
| 160 | 129.44 | - | 87.63 | 84.49 |
| 200 | 129.40 | 98.91* | 87.55 | 83.84 |
| 240 | 129.37 | - | 87.47 | 83.06 |
| 280 | 129.31 | - | 87.44 | 82.32 |
| 320 | 129.29 | - | 87.41 | 81.84 |
| 360 | 129.13 | - | 87.39 | 81.35 |
| 400 | 129.09 | - | 87.38 | 80.98 |

Table 1: Quality of upper bounds.

Model-based sampling schemes that target belief points with largest improvement potential can be designed by repeating simulations from critical points. Table 1 compares results one can achieve using such methods and other grid selection strategies for the Maze20 problem and grid sizes of 40 - 400 points. Methods tested are: random grid method; regular grid method; and two versions of heuristic grid method, one with model-based sampling that uses a fixed initial MDP-based bound, the other in which new points are always sampled using the value function bound acquired in the previous step. The quality of every bound is measured by a score that represents an average value for all critical points and a fixed set of 2500 randomly generated belief points. All but the regular grid method use the simple interpolation rule described above and are tested on grids with 40 point increments starting from the initial MDP-based bound. The regular grid method has been tested on the grid of size 210 that falls into the tested range. For every grid, the inner approximate value iteration was run using the relative stopping criterion with a maximum allowable improvement of 0.1 for every grid point.

The worst results for the Maze20 problem were achieved for randomly generated grids. This is mostly because the transitions in Maze20 model are local. This means that from any critical point one can get only to belief states that lie on the boundary of the belief simplex, i.e. those that contain lots of zeros. Contrary to this, random sampling is more likely to produce a belief point with nonzero probabilities. As any boundary point can be interpolated using only points on the same boundary, the internal points of the belief simplex has no effect on their interpolation and thus there is a very slim chance that critical points will get updated by randomly generated grids. On the other hand, a regular grid (with the resolution of 210 points) consists only of points on the boundary. This fact is also reflected by a significantly better bound score.

Both model-based sampling entries beat random as well as regular grid approaches. Between the two the better score was achieved by the method that samples new grid points using the most recent bound function. The difference in results also illustrates a potential problem with large grid increments in the context of model-based sampling. The reason for this is that large grid increments would likely lead to a large number of grid points with small bound improvement effect.

### Improving initial upper bound

The MDP-based initial bound can be significantly improved by a new method – the fast informed bound method. The method is iterative and does not use grids, but rather tries to utilize information from the POMDP model directly. In this method a value function $\widehat{V}^i$ is represented as a piecewise linear convex function with at most $|A|$ different linear vectors in $\Gamma^i$, each corresponding to one action. Linear vectors $\alpha_a^{i+1} \in \Gamma^{i+1}$ are updated using the following formula:

$$\alpha_a^{i+1}(s') = \rho(s', a) + \gamma \sum_{o \in \Theta^+} \max_{\alpha_k^i \in \Gamma^i} \sum_{s \in S} P(s, o|s', a)\alpha_k^i(s)$$

The function update can be described via function mapping $H_{FBM}$. $H_{FBM}$ is a contraction with a fixed point solution $\widehat{V}^* \geq V^*$, thus the method converges to the upper bound of the optimal value function. The major advantage of the fast bound method is that there are at most $|A|$ different $\alpha_a^{i+1}$s. This guarantees an efficient update with regard to $|S|, |A|, |\Theta|$. Testing the fast bound method on the Maze20 problem yielded the bound score of 102.48 comparing to 130.11 for the MDP-approximation, thus leading to a significant bound improvement.

## Computing lower bounds

A lower bound on the optimal value function for infinite horizon discounted problem can be acquired using a simple method that updates derivatives (linear vectors) for belief points using equation 3 (Smallwood and Sondik 73). Assume that $\widehat{V}^i \leq V^*$ is a convex piecewise linear lower bound on the optimal value function, defined by a a linear vector set $\Gamma_i$, and let $\alpha_b$ be a linear vector for a point $b$ that is computed from $\widehat{V}^i$ by the Sondik's method. As it holds that $\widehat{V}^i(b) \leq \sum_s b(s)\alpha_b(s) \leq V^*(b)$, one can easily construct a new value function $\widehat{V}^{i+1} \geq \widehat{V}^i$ by simply updating $\Gamma_i$ to: $\Gamma_{i+1} = \Gamma_i \cup \alpha_b$ (Hauskrecht 96). Such a value function update guarantees that the new lower bound is improved. Note that after adding new linear vector to $\Gamma_i$ some of the previous linear vectors can become redundant. To fix that the update step can be combined with various redundancy check procedures.

The new lower bound update rule can be turned directly into various iterative algorithms with the incremental bound improvement property. Unfortunately the new update rule also causes the size of $\Gamma^i$ to grow with every iteration. However, assuming that only a single optimizing linear vector for every point is selected, the growth is linear in the number of steps. This together with the efficient update procedure guarantees efficient running time of such an algorithm for a fixed number of iteration steps. A simple incremental lower bound improvement algorithm is shown below. The algorithm starts from the initial lower bound $\widehat{V}_{init}$ (with linear vector set $\Gamma_{init}$), selects a belief point and updates an existing lower bound with a new linear vector.

**Incremental lower bound** $(m, \widehat{V}_{init})$
    **set** $\Gamma$ defining current bound $\widehat{V}$ to $\Gamma_{init}$
    **repeat** until the stopping criterion is met
        **select** a belief point $b$
        **compute** new update $\alpha_b$ for $b$
        **add** the $\alpha_b$ to $\Gamma$
**return** $\widehat{V}$

The initial lower bound $\widehat{V}_{init}$ can be computed via the blind policy method (Hauskrecht 96). The main idea of the method is to compute value functions for all "one-action" (or blind) policies. These correspond to simple linear value functions that lower bound the optimal one and can be easily computed within the fully observable MDP framework. The linear value functions for every action in $A$ can be then combined to a lower bound piecewise linear function as:

$$\widehat{V}(b) = \max_{a \in A} \sum_{s \in S} b(s) V_{MDP}^a(s)$$

### Selecting points for update

The update phase of the incremental lower bound method is not limited to any specific point. One may combine it with arbitrary point selection strategies, including various heuristics. The above incremental method can in principle lead also to the $\epsilon$-optimal solution. However in this case the method must use some systematic way of chosing points to be updated next.

With an objective to speed up the improvement of the bound we have designed and implemented a relatively simple two tier heuristic point selection strategy that tries to optimize updates of a bound value function at critical belief points. The top level strategy attempts to order critical belief points. It builds upon the fact that states with higher expected rewards (e.g., a goal state in Maze20) backpropagate their effects locally. Therefore it is desirable that states in the neighbourhood of the highest reward state are updated first, and distant ones later. The strategy uses the current value function to identify the highest expected reward states and the POMDP model to determine local dependencies and order neighboring states. The lower

| updates | bound score | | |
|---|---|---|---|
| | fixed-random | random | heuristic |
| blind-policy | 33.10 | 33.10 | 33.10 |
| 40 | 43.80 | 44.65 | 45.16 |
| 80 | 46.20 | 47.93 | 48.17 |
| 120 | 47.10 | 49.28 | 49.60 |
| 160 | 47.89 | 50.12 | 51.19 |
| 200 | 48.23 | 50.43 | 52.64 |
| 240 | 48.60 | 50.70 | 53.04 |
| 280 | 48.89 | 51.18 | 53.58 |
| 320 | 49.06 | 52.02 | 53.88 |
| 360 | 49.38 | 52.40 | 53.96 |
| 400 | 49.67 | 52.84 | 54.13 |

Table 2: Quality of lower bounds

level strategy uses the idea of stochastic simulation, similar to the one used in the upper bound method, to generate a sequence of belief points that can result from a given critical point. These points are then used in reverse order to update the current value function.

Both partial strategies try to sequence belief points to increase the benefit of updates from other points. The complete selection cycle can be repeated once all critical points became updated. The drawback of this approach can be that after a few cycles the strategy will sample points from a rather restricted set of points and that can lead to smaller and smaller improvements. A possible solution would be to combine a heuristic strategy with random sampling of points that tends to spread new points evenly over the belief space.

The quality of a constructed heuristic method with the simulation sequence of 5 steps was compared with strategies that used a fixed set of 40 points consisting of all critical points and 20 randomly selected belief points, and a strategy that generated points randomly for every update. The quality of bounds achieved were compared after 40 updates using the same bound quality measure as in the upper bound case.

The results (table 2) showed that the best bound quality was achieved by the designed heuristic method. However the differences between strategies were not very large. The following migth be reasons for this: the initial lower bound is not far from the optimal solution and improvements are hard to make; the new linear vector added by the method influences a larger portion of the belief space and thus changes are easier to propagate; we did not use a very good heuristic and better heuristics or combinations can be constructed.

## Conclusion

In this paper we have proposed simple incremental methods for computing upper and lower bounds on the optimal value function. Both are based on the value iteration approach and allow one to use arbitrary belief points to either refine the grid (upper bound) or update a piecewise linear convex function (lower bound). This feature provides room for various heuristic strategies to be used in belief point selections that can lead to tighter and faster bounds. The upper bound incremental methods can also benefit from the newly introduced fast informed bound method, that can outperform standard initial MDP-based bounds.

The heuristic strategies we have designed and tested are based mostly on the idea of stochastic simulation. This approach tries to increase the chance of bound improvement by exploiting model dependencies. We have found that the simulation strategy works extremely well when combined with the upper bound method on the test maze navigation problem. Contrary to this, only a small improvement compared to randomized strategies has been achieved for the lower bound method. These differences may also turn out to be problem specific and further study of methods on a spectrum of other problems is needed.


## Acknowledgements

This research was supported by the grant 1T15LM07092 from the National Library of Medicine. Peter Szolovits has provided valuable comments on early versions of the paper.


## References


Astrom, K.J. 1965. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications* 10: 174-205.

Cassandra, A.R. 1994. Optimal policies for partially observable Markov decision processes. Technical report CS-94-14, Brown University.

Hauskrecht, M. 1996. Planning and control in stochastic domains with imperfect information. PhD thesis proposal, EECS, MIT.

Hauskrecht, M. 1997a. Dynamic decision making in stochastic partially observable medical domains: Ischemic heart disease example. In Procedings of AIME-97.

Hauskrecht, M. 1997b. Approximation methods for solving control problems in partially observable Markov decision processes. Technical Memo. MIT-LCS-TM-565.

Littman, M.L.; Cassandra, A.R.; Kaelbling, L.P. 1995a. Learning policies for partially observable environmets: scaling up. In Proceedings of the 12-th international conference on Machine Learning.

Littman, M.L.; Cassandra, A.R.; Kaelbling, L.P. 1995b. Efficient dynamic programming updates in partially observable Markov decision processes. submitted to *Operations Research*.

Lovejoy, W.S. 1991. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research* 39(1):192-175.

Lovejoy, W.S. 1993. Suboptimal policies with bounds for parameter adaptive decision processes. *Operations Research* 41(3):583-599.

Parr, R.; Russell, S. 1995. Approximating optimal policies for partially observable stochastic domains. In Proceedings of IJCAI-97, 1088-1094.

Smallwood, R.D.; Sondik, E.J. 1973. The optimal control of Partially observable processes over a finite horizon. *Operations Research* 21:1071-1088.