5-2011

Pond-Hindsight: Applying Hindsight Optimization to Partially-Observable Markov Decision Processes

Alan Olsen
Utah State University

POND-HINDSIGHT: APPLYING HINDSIGHT OPTIMIZATION TO PARTIALLY-OBSERVABLE MARKOV DECISION PROCESSES

by

Alan Olsen

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Daniel L. Bryce Major Professor	Dr. Vicki H. Allan Committee Member
Dr. Daniel W. Watson Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2011

ABSTRACT

POND-Hindsight: Applying Hindsight Optimization to Partially-Observable Markov Decision Processes

by

Alan Olsen, Master of Science

Utah State University, 2011

Major Professor: Dr. Daniel L. Bryce
Department: Computer Science

Partially-observable Markov decision processes (POMDPs) are especially good at modeling real-world problems because they allow for sensor and effector uncertainty. Unfortunately, such uncertainty makes solving a POMDP computationally challenging. Traditional approaches, which are based on value iteration, can be slow because they find optimal actions for every possible situation. With the help of the Fast Forward (FF) planner, FF-Replan and FF-Hindsight have shown success in quickly solving fully-observable Markov decision processes (MDPs) by solving classical planning translations of the problem. This thesis extends the concept of problem-determinization to POMDPs by sampling action observations (similar to how FF-Replan samples action outcomes) and guiding the construction of policy trajectories with a conformant (as opposed to classical) planning heuristic. The resultant planner is called POND-Hindsight.

A number of technical approaches had to be employed within the planner, namely, 1) translating expected reward into a probability of goal satisfaction criterion, 2) monitoring belief states with a Rao-Blackwellized particle filter, and 3) employing Rao-Blackwellized particles in the McLUG probabilistic conformant planning graph heuristic. POND-Hindsight

is an action selection mechanism that evaluates each possible action by generating a number of lookahead samples (up to a fixed horizon) that greedily select actions based on their heuristic value and samples the actions' observation; the average goal satisfaction probability of the end horizon belief states is used as the value of each action.

POND-Hindsight was entered into the POMDP track of the 2011 International Probabilistic Planning Competition (IPPC) and performed comparable to its competitors – ranking in the middle of six planners. Benchmarks on the IPPC-2011 problems were run on a cluster of identical computers in order to evaluate computation time and plan quality. Success can be attributed to determinization of the problem, and failure can be attributed to a sometimes misleading heuristic combined with a greedy best-first lookahead algorithm.

(33 pages)

ACKNOWLEDGMENTS

I am most grateful for the love and support of my wife, Amber. I also greatly appreciate all of the guidance, instruction, and revisions and corrections from my major professor, Dr. Daniel Bryce. He has always been reachable, and has given prompt and useful feedback. Thanks go to my committee members, Dr. Vicki Allan and Dr. Daniel Watson, for patiently working with me through difficult summer scheduling. I would like to thank Myra Cook for answering my questions concerning formatting and for performing the final editing of this thesis. Special thanks go to Scott Sanner and Sungwook Yoon for organizing the 2011 International Probabilistic Planning Competition, of which this work competed in.

This work was supported by DARPA contract HRC0011-07-C-0060.

Alan D. Olsen

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Classical planning is a good start to autonomous problem solving but is not equipped to handle all real-world problems, as it does not account for non-deterministic actions and imperfect sensors. Realistically, a robot does not move precisely as intended, and it does not see the world exactly as it is. Markov decision processes (MDPs) are a step closer, as they model non-deterministic actions, but perhaps the best model for a real-world problem is a *partially*-observable MDP (POMDP), which models the uncertainty of actions and observations. With a POMDP, one is able to generate plans for autonomous agents that take into account the non-determinism and uncertainty of their actions and sensors.

Unfortunately, POMDPs are much more difficult to solve than classical planning problems. The work in this thesis is in response to the POMDP track of the 2011 International Probabilistic Planning Competition (IPPC) – a challenge to produce a quick and effective POMDP solver. Many existing approaches use some form of value iteration to predict the realization of rewards for performing any action from any state of the world. An alternate approach is to start at the initial belief state and search through the belief space, therefore only solving the parts of the problem that are truly necessary. Still, even that can be quite time consuming, when all contingencies are taken into account.

Previous success has been found in determinizing a stochastic problem so that only a select number of contingencies need to be considered. The winner of IPPC-2004 and unofficial winner of IPPC-2006, FF-Replan [20], solves MDPs with the aid of a classical planner. FF-Hindsight [22] uses similar techniques but in a more thorough way, in order to solve more challenging MDPs. An analogous approach for POMDPs could be to choose what is observed, as opposed to the outcome of an action. This would transform the problem into a conformant one, rather than a classical one. Therefore, the conformant planner,

POND [4], is used in this work to solve POMDPs. Adapting the idea of determinization to a POMDP competition setting, for IPPC-2011, presents several unique challenges: 1) POMDPs allow observation uncertainty in addition to action outcome uncertainty; 2) the POMDP instances are formulated with rewards instead of goals (as in the MDP track of past IPCs); and 3) the POMDP instances have a finite horizon and are not discounted.

The approach taken for POND-Hindsight was to formulate a goal-based belief state MDP, allowing a straight-forward application of the outcome sampling ideas made popular in FF-Replan and FF-Hindsight. In the belief state MDP, the action outcomes correspond to the POMDP observations, and the states correspond to the POMDP belief states. Thus, sampling action outcomes in the belief state MDP corresponds to sampling observations in the POMDP, and computing successor states corresponds to computing successor belief states. While sampling observations, planning is still done in the belief state space, and a reachability heuristic for belief state space is used based on the McLUG [5]. The relaxed planning heuristic extracted from the McLUG ignores observations but accounts for probabilistic action outcomes – estimating the number of actions required to support the goal with high probability under non-observability.

Rewards are translated into goals by introducing a goal proposition whose probability is increased by each action – the magnitude of the increase is proportional to the reward. In most problems, this translation leads to belief states wherein the goal proposition is true with very low probability; employing a particle filter for belief state monitoring is problematic because very few, if any, particles correspond to states wherein the goal proposition is true. To more accurately track the rewards, Rao-Blackwellized particles [7] are used wherein each particle assigns a value to each state proposition and retains a distribution over the goal proposition. The same issue of a low probability goal proposition appears in the McLUG, and so Rao-Blackwellized particles are also used in the planning graph heuristic.

POND-Hindsight varies the number of particles in its belief state monitoring N, the number of particles in its heuristic M, the number of futures sampled per action during

action selection F, and the depth of each future D. The depth of the futures is limited for each action in order to reduce search cost, and the McLUG heuristic is used to bias the search.

The following contains formal descriptions for MDPs and POMDPs, past methods for solving MDPs and POMDPs, a review of the procedures used by FF-Replan and FF-Hindsight to solve determinized translations of MDPs, a description of the POND-Hindsight planning algorithm along with a few optimizations, a discussion of results for the IPPC-2011 domains, and a conclusion with directions for future work.

CHAPTER 2

BACKGROUND

This work and related work are based on certain formulations for probabilistic problems.

2.1 Framework

Actuators provide the ability to act in, or influence the world. Sensors provide the ability to observe the world. The goal in planning is to determine how to use a set of actuators and sensors to solve a task. MDPs are able to model the uncertainty of actuators and sensors in a simple and compact way. Definitions of MDP model concepts can be found in [3, 18]. The MDP model (S, A, T, R, H) is used wherein

- S is a finite set of states (the power set over all Boolean propositions or facts)
- A is a finite set of actions
- $T(s, a, s') = Pr(s_{t+1} = s' | a_t = a, s_t = s)$ models transition between states
- $R(s, a)$ is the reward for applying a in s
- H is a finite horizon, or number of steps to solve

Each state of the world has a set of applicable actions that can be performed. There is some probability that an action transitions to a particular, new state. When each action only transitions to one state with a 1.0 probability, and the horizon is indefinite and absorbing, the problem is considered a classical planning problem. When action outcomes are uncertain but sensors are complete and deterministic, the MDP is sometimes called a fully-observable MDP (FOMDP), but generally they are simply called MDPs. When sensors are also non-deterministic, then the MDP is typically called a partially-observable MDP (POMDP). In this case, each state is replaced by a belief state – a probabilistic distribution over states –

and each action transitions to one of a set of belief states depending on the probability of receiving some sensory observation.

The POMDP model $(S, A, O, b_0, T, R, \Omega, H)$ extends the MDP model, above, with the addition that

- O is a finite set of observations
- $b_0(s) = Pr(s_0 = s)$ is an initial belief state, or distribution over states
- $\Omega(o, s, a) = Pr(o_{t+1} = o | a_t = a, s_{t+1} = s)$ is the probability of observing o after applying action a and obtaining state s

The belief state after a history, or sequence, of actions and observations is defined as

$$b_t = Pr(s_t | b_0, a_0, o_1, ..., o_{t-1}, a_{t-1}, o_t)$$

which can be computed incrementally from a prior belief state, given action a, observation o, and normalization constant α, so that

$$b_t(s') = \alpha \Omega(o, s', a) \sum_{s \in S} T(s, a, s') b_{t-1}(s)$$

The transition probability between belief states $b_t = b$ and $b_{t+1} = b_a^o$, given an action and observation is

$$T(b, a, b_a^o) = \sum_{s' \in S} \Omega(o, s', a) \sum_{s \in S} T(s, a, s') b(s)$$

The reward associated with applying an action in a belief state is

$$R(b, a) = \sum_{s \in S} R(s, a) b(s)$$

Definitions of POMDP model concepts can also be found in [3, 18].

The solution to an MDP is called a policy, π, and simply maps states to actions, $\pi : S \rightarrow A$. Policies for POMDPs are only slightly different in that they map *belief states* to actions. A policy is non-stationary if it maps a state to a particular action depending on time, $\pi : S \times t \rightarrow A$. The optimal policy maps each state to the action that gives the greatest expected end reward. A partial policy only provides a mapping for some of the states. A policy trajectory is a partial policy that leads from a source state to a destination state by following a sequence of actions and is typically, but not necessarily, non-stationary. In the case of POMDPs, a policy trajectory is a sequence of action-observation pairs.

2.2 IPC and IPPC

International Planning Competitions (IPCs) are held in conjunction with the International Conference on Automated Planning and Scheduling (ICAPS). The goal of the first IPC, in 1998, was to motivate and excite researchers in automated planning and to provide a standard for other planners to compare against [16]. IPC-1998 focused on two different translations of classical planning problems. Since then, various tracks have been included to provide a challenge for new areas of automated planning. The uncertainty/probabilistic track focuses on solving MDPs and POMDPs, and is also referred to as an International Probabilistic Planning Competition (IPPC). Six planners, including POND-Hindsight, were entered into the POMDP track of IPPC-2011 [19]. Another five planners were entered into the MDP track.

A few model parameters were fixed across all of the domains that were used in IPPC-2011. While it is possible to model and solve infinite horizon [PO]MDPs, a horizon of 40 was used. Typically, infinite-horizon problems involve a reward discount factor, γ, between 0.0 and 1.0, which decreases the value of future rewards (giving emphasis to immediate reward). Because the horizon was finite for all domains, no discount factor was used (rather, $\gamma = 1.0$ which means that future rewards were not discounted).

Probabilistic planning problems can be solved online or offline. Online planning begins with the initial belief state and alternates between selecting an action to perform and receiving a full or partial observation (transitioning to the next appropriate belief state). Offline planning generates part, or all, of a policy without any feedback. An online planner can be used to solve problems in an offline manner simply by simulating all possible contingencies. Therefore, online planning is more fundamental than offline planning. IPPCs have been taking the online approach.

The Amazon EC2 (Elastic Compute Cloud) was used to run IPPC-2011. This made each competitor's machine equal in processor power and resources. A competitor was allowed 24 hours to complete up to 80 problems (10 instances each of 8 domains). First, a competitor's client requests a problem to solve. Then, the server and the client alternate

between sending an observation and sending an action, respectively. This simulation to the 40-step horizon is repeated 30 times per problem. Rewards are then averaged over all 30 trials to obtain the score for that problem. If the client requests a problem that it has already completed, its previous score for that problem is replaced.

After the competition, each problem score is normalized between the minimum and maximum score for that problem. The maximum is determined by reviewing all competitor's scores, for that problem, as well as the scores obtained by a random policy and a no-operation (noop) policy. The minimum is either the random or noop score. In the case that a competitor's normalized problem score is below 0, it is considered a 0. The final score for each competitor, then, is the sum over all normalized problem scores for that competitor.

CHAPTER 3

RELATED WORK

Various approaches have been taken to solve probabilistic problems. Dynamic programming has been used to analytically compute the utility or value of each action in order to determine which to perform. Planning, on the other hand, searches from an initial state to a goal state. Some planners translate the probabilistic problem into a deterministic one in order to approximate the value of performing an action. POND-Hindsight is most similar to those planners that determinize the problem.

3.1 Dynamic Programming

Value iteration [2,3] is a dynamic programming procedure that can be thought of as a brute force approach to solving MDPs. The value of every state, s, is calculated for every time step, t, from 0 to the horizon, H, with the follow equations:

$$V_0^*(s) = 0 \tag{3.1}$$

$$V_t^*(s) = \max_{a \in A}\{R(s,a) + \sum_{s' \in S} T(s,a,s')V_{t-1}(s')\}, \quad 0 < t \leq H \tag{3.2}$$

The value at time, t, can be thought of as the value of being in a state if t steps will be executed. The base case in Equation 3.1 says that at time 0, every state has a value of 0. The recursive step in Equation 3.2 can be repeated up to $t = H$ in order to determine the value of being in state s given that H steps will be executed. The summation in the equation gives an expected value for performing an action, based on the states that the action will possibly lead to. The value of selecting a particular action for a state is equal to the reward of taking that action plus its expected value. The value of a state, then, is equal to the most valued action. Extracting the optimal non-stationary policy, mapping

each state to the optimal action at time t, is very straightforward. This is done simply by replacing the max with an arg max like so:

$$\pi_t^*(s) = \arg\max_{a \in A} \{R(s,a) + \sum_{s' \in S} T(s,a,s') V_{t-1}(s')\} \tag{3.3}$$

This whole process is quite computationally intensive as it ultimately iterates H times, determining the value (and inherently determining the optimal action) for every possible state in S.

The concept of value iteration can be extended to POMDPs by replacing states with probability distributions over states, called belief states. The value function, then, is no longer a real number but an $|S|$-dimensional function. That is to say, the value of a belief state is dependent on each probability of being in a particular state of S.

Point-based value iteration (PBVI) samples the belief space in order to select a finite number of belief points to use during value iteration of a POMDP [18]. This approximates value iteration by discretizing the value function and then focusing on some of the most probable beliefs.

3.2 Planning Under Uncertainty

An alternative to value iteration is planning under uncertainty. For this to be done, an initial belief state must be specified, which it is in the IPPCs. Planning under uncertainty solves [PO]MDP style problems by searching from the start state to one or more goal states.

3.2.1 Conformant Planning

Conformant planning typically solves no-observation MDPs (NOMDPs), but can also solve POMDPs if observations are ignored. A conformant plan consists of a sequence of actions that lead to the goal with at least some probability, θ, that is predetermined.

Probabilistic-FF [6] produces conformant plans for problems that have an initial belief state and uncertain action outcomes. Belief states are represented as Bayesian networks. The probability of a belief state satisfying goal conditions is calculated by translating the

Bayesian network into a propositional logic formula, ϕ, that is weighted according to conditional probability tables, and then performing weighted model counting on it. A heuristic is used to guide search and to prune dead-ends. This heuristic takes concepts from the FF relaxed planning graph heuristic, by generating a plan that ignores negative effects of actions, and then returning the length or number of actions in that plan.

POND [4] also searches through belief space. Unlike Probabilistic-FF, however, the goal satisfaction of a belief state is analytically computed and a Monte Carlo labeled uncertainty graph (McLUG) [5] is used by the heuristic to generate a relaxed plan.

MaxPlan [14] solves problems quite differently. Rather than searching through belief space, the entire problem is translated into an E-MAJSAT problem, which is basically a probabilistic Boolean satisfiability problem. Solving this translation yields the plan solution.

3.2.2 Contingency Planning

Instead of ignoring observations, contingency planning produces plans that are contingent upon observations. While all plans to this point have been defined as a sequence of actions, a contingent plan takes shape in the form of a tree or a graph of actions and observations. Each node represents an action in the plan, and each edge represents an observation that might be made during plan execution. Planning for multiple contingencies creates a more complete plan than conformant planning, but at the cost of search time.

C-MaxPlan and Zander [15] were created as extensions to MaxPlan by including observations into the translation. C-MaxPlan continues the use of E-MAJSAT, while Zander translates the problem into S-SAT instead.

C-Buridan [8] iteratively builds a solution in the form of a directed acyclic graph (DAG) until the solution can guarantee a predetermined probability of reaching the goal. Building the graph can be done by inserting actions that increase the probability of goal satisfaction, eliminate threats to goal satisfaction, or provide a branch in the graph based on some observation. DTPOP [17] is a partial-order planner (i.e., it allows the execution of multiple actions at the same time), that uses techniques from C-Buridan.

Contingent-FF [12] performs AO* search over a tree of belief states, which are represented as CNF formulas. The contingent problem is translated into a relaxed conformant problem that is used in a conformant relaxed planning heuristic similar to the one used in Probabilistic-FF.

The closed-loop greedy (CLG) planner [1] generates contingent plans by performing enforced hill climbing (EHC) on a conformant translation of the problem. A relaxed classical planning translation of the conformant problem is used as a heuristic. This technique is the most similar to POND-Hindsight, except that POND-Hindsight performs greedy best-first search to a limited depth and uses the McLUG heuristic in order to solve reward-based POMDPs.

POND is able to produce contingent plans, in addition to conformant plans, simply by applying observations to the belief states. This produces an And/Or graph, so LAO* search [9] is used. The planner can be reduced to perform conformant planning by sampling one observation or ignoring observations entirely. In such a case, LAO* would behave similar to A* and a conformant plan would be returned.

3.3 Determinizing MDPs

Work by Yoon, Fern, and Givan [20] and Yoon et al. [21] has shown that deterministic translations of probabilistic problems can be used to quickly find approximate solutions to the original problem.

3.3.1 Replan

FF-Replan generates partial policies for MDPs in an online manner [20]. First, it translates the problem into a deterministic one. The "single-outcome" approach does this by keeping only the most probabilistic outcome for each action. The "all-outcome" approach does this by generating a deterministic action for each probabilistic outcome of each action in the original problem. At each state in the online search, FF-Replan sends the deterministic translation of the problem, along with the current state, to a deterministic planner - such as FF [10] - to generate a policy trajectory. FF-Replan follows the generated policy trajectories

during online search, where possible, but whenever it is brought to a state that it has not yet solved, it replans by generating a new policy trajectory for that state.

The simplicity of FF-Replan has proven itself quite successful. It was the winner of IPPC-2004 and also would have won IPPC-2006 except that it was only an unofficial entry (one of the authors was also head of the competition). The paper [20] does mention that one weakness of FF-Replan is that it discards all probabilistic information when it determinizes the problem. This weakness was explored in [13], in which FF-Replan's success was attributed to the problems of the 2004 and 2006 IPPCs being "probabilistically uninteresting," trivial for any planner, or too large for probabilistic planners.

3.3.2 HOP or FF-Hindsight

Hindsight optimization (HOP) [21], later called FF-Hindsight [22], improves upon FF-Replan by utilizing the probabilistic information of the problem. This time, the generation of a policy trajectory obeys a "future" that is described as the mapping of an action at time, t, to a sampled outcome. This is different from the single-outcome approach of FF-Replan in two ways. First, the sampled outcome may not necessarily be the highest probability outcome. Second, because an action may have a different outcome at a different time, policy trajectories are non-stationary, meaning, a state may map to a different action depending on the time.

FF-Replan is able to choose its own future. In the single-outcome case, the future chosen is the one in which each action's outcome, at any time, is the most probable one. Sometimes, however, the goal may only be reached from a low probability outcome, in which case the determinization is a dead end. In the all-outcome case, the future chosen is the one in which each action's outcome is the one that leads to the goal the soonest, regardless of probability. It is obvious that the all-outcome approach is overly optimistic in its ability to reach the goal. By choosing a future from the uniform distribution over all futures, FF-Hindsight is able to occasionally consider low probability outcomes as well as produce policy trajectories that are not overly optimistic, but realistic.

The length of each policy trajectory produced gives a possible distance to the goal.

Averaging over the lengths of multiple policy trajectories from the same state but with different futures gives an expected distance to the goal. Of course, generating more trajectories takes more time but also helps approximate the distance to the goal better. FF-Hindsight generates an equal number of policy trajectories for each action applicable to the current state, and then averages them to determine an expected distance to the goal if that action is taken. During online search, FF-Hindsight uses the policy trajectories as a heuristic only. It chooses the action with the minimum expected distance to the goal and then discards the the policy trajectories that were generated.

FF-Hindsight was later modified in three significant ways, and called FF-Hindsight+ [22]. First, Instead of sampling an equal number of trajectories for every action, trajectories are generated from the current state in order to determine which of that state's actions show up first in any of the trajectories (called, "probabilistically helpful actions" or PHAs). An equal number of trajectories are then generated through each of these PHAs only. Second, after selecting the action with the minimum expected distance to the goal, FF-Hindsight+ keeps the longest prefix that all trajectories of that action share in common. Third, it always includes an all-outcome generated trajectory when calculating the average.

CHAPTER 4

PLANNING POMDP TRAJECTORIES

In order to solve reward-based POMDPs with POND (a goal-based planner), rewards must be translated into an analogous goal satisfaction metric. POND-Hindsight can then select actions according to a limited lookahead technique similar to FF-Hindsight.

4.1 Translation to Goal-Based POMDPs

As defined by Majercik and Littman [15], discounted expected reward POMDPs can be translated to goal-based POMDPs of the form $(S^*, A, O, b_0, T^*, \Omega)$, where

- $S^* = S \cup \{goal, sink\}$
- $T^*(s, a, goal) = R^*(s, a)$
- $T^*(s, a, sink) = (1 - \gamma) - R^*(s, a)$
- $T^*(s, a, s') = \gamma T(s, a, s')$

where the *goal* and *sink* states are absorbing and all rewards $R(s, a)$ are transformed so that $0 \leq R^*(s, a) < 1 - \gamma$. Then, the value of a belief state, following a policy, is

$$V^*_\pi(b) = \sum_{s \in S} T^*(s, \pi(b), goal) b(s) + \sum_{o \in O} T^*(b, \pi(b), b^o_{\pi(b)}) V^*_\pi(b^o_{\pi(b)})$$

4.2 POND-Hindsight

A future in POND-Hindsight samples observations as opposed to action outcomes, as in FF-Hindsight. By selecting the action outcome in an MDP, FF-Hindsight knows when it has reached a goal state with complete certainty. By only selecting what is observed and not the action outcome in a POMDP, POND-Hindsight performs lookahead over belief states, which consist of real-valued goal satisfaction. Therefore, POND-Hindsight treats all belief states on the lookahead horizon as goal states during a lookahead search.

Like FF-Hindsight, POND-Hindsight produces several policy trajectories from each immediate action to the goal, averages the quality of the set of trajectories for an action, and then selects the action with the best average quality (shown in Algorithm 1). Also, notice that an action is reused whenever a belief state is revisited. The policy is even preserved between trials, which saves greatly on computation time at the expense of introducing bias.

Algorithm 1 *GetBestAction*

Input: Belief state, b; Number of lookahead samples, F; lookahead depth, D
Output: Best action, a^*

```
if π(b) = null
  for each action a applicable to b
    for i = 1 to F
      R_i(a) ← GreedyLookahead(b, a, D)
    end for
    R̄(a) ← Σ_{i=1}^{F} R_i(a)/F
  end for
  π(b) ← arg max_a R̄(a)
end if
a* ← π(b)
```

4.2.1 Greedy Lookahead

FF-Hindsight measures quality as the distance to the goal. However, POND-Hindsight measures quality as the probability of goal satisfaction (i.e., translated reward). Because the search space to the horizon can be very large, a greedy best-first approach was taken (see Algorithm 2). While searching for the goal, the state with the minimum heuristic value is chosen first. In the case of a tie, the state with the higher goal satisfaction is chosen. Remaining ties are broken randomly.

4.2.2 Efficiency Improvements

In order to make POND competitive, the computation of belief states has been changed from an exact computation to a Rao-Blackwellized particle filter $RBPF(b, a, N)$, for applying action a to belief state b and using N particles [7]. As part of the particle filter, only one observation is sampled based on its probability of being generated after applying a to b.

Algorithm 2 *GreedyLookahead*

Input: Belief state, b; Action, a; Lookahead depth, D
Output: Probability of goal satisfaction estimate, r

```
b' ← RBPF(b, a, N)
t ← 0
closed ← ∅
open ← {⟨b, t⟩}
while open ≠ ∅
  ⟨b, t⟩ ← open item with min h(b), max Pr(goal|b)
  if t = horizon
    r ← Pr(goal|b)
    return
  end if
  closed ← closed ⋃ {⟨b, t⟩}
  for each action a' applicable to b
    b' ← RBPF(b, a', N)
    t' ← t + 1
    open ← open ⋃ {⟨b', t'⟩} \ closed
  end for
end while
```

Actions are represented as dynamic Bayesian networks (DBNs), and a successor belief state is computed by sampling the original state propositions but analytically computing the distribution over the goal and sink propositions. Because the McLUG heuristic samples outcomes, and the goal and sink states have very low probabilities, the heuristic also had to be altered to analytically compute the goal. Because belief states are approximated by a particle filter, sometimes a belief state will have no particles that agree with an observation given by the IPPC server. In such a case, the planner would be unable to continue search. To handle this, POND-Hindsight updates its belief state according to the previous action performed but ignores the conflicting observation (similar to conformant planning).

Another improvement, which is reported on in the next section, is a bound on the number of antecedents for conditional effects in the McLUG heuristic. As each action is a DBN, each row in a conditional probability table can be treated as a conditional effect. Hoffmann and Brafman [11] discovered that limiting the number of antecedents of conditional effects significantly reduces the cost of heuristic construction in conformant planning heuristics.

CHAPTER 5

EMPIRICAL RESULTS

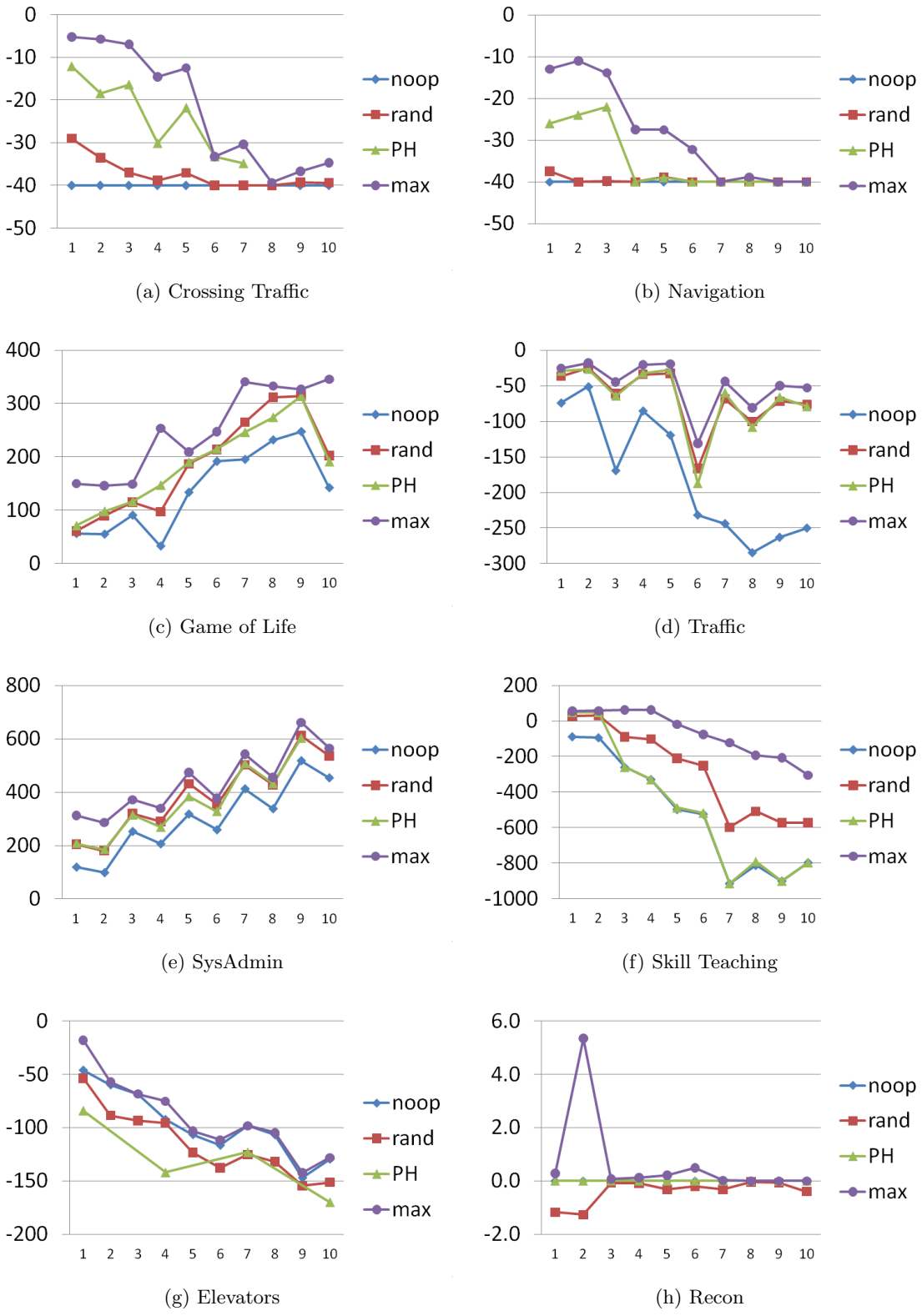
A portion of the benchmarks presented in this section are actual results from IPPC-2011. Parameters for state particle filtering, heuristic computation, and the lookahead algorithm were adjusted for the competition in order to maximize performance under a limited time frame of 24 hours, that is to say, some instances were poorly approximated for the sake of time. The planner used a single 2.66 GHz processor and was limited to roughly 7 GB of memory during the competition. The remainder of the benchmarks are runs or re-runs of instances outside of the competition with sufficient parameters. These were run on a single 1.99 GHz processor and were also limited to 7 GB of memory. Therefore, all benchmarks shown in this section represent instances that were run with sufficient parameters. Missing data corresponds to instances that took longer than 3 hours to solve.

5.1 Benchmarks on IPPC Domains

There were eight domains used for IPPC-2011: crossing traffic, navigation, traffic, game of life, sysadmin, skill teaching, elevators, and recon. Each domain has 10 problem instances associated with it. Table 5.1 shows the parameters that were used to solve each set of instances. Rewards for each instance are averaged over 30 trials. These are compared against the maximum rewards from the IPPC, an all-"noop" policy, and a random action policy, also averaged over 30 trials, and shown in Figure 5.1.

Crossing traffic resembles the game of Frogger. A straight, one-way road separates a robot from its goal. Cars randomly appear at one end of the road and deterministically travel forward. Difficulty increases with the rate that cars appear and the number of lanes of traffic. POND-Hindsight performed its best on this domain, as can be seen in Figure 5.1a.

Navigation requires a robot to cross a sort of mine field to get to its goal. As with



(a) Crossing Traffic

(b) Navigation

(c) Game of Life

(d) Traffic

(e) SysAdmin

(f) Skill Teaching

(g) Elevators

(h) Recon

Figure 5.1. Rewards for 10 instances, along the horizontal axis, of each domain (averaging over 30 trials for each instance)

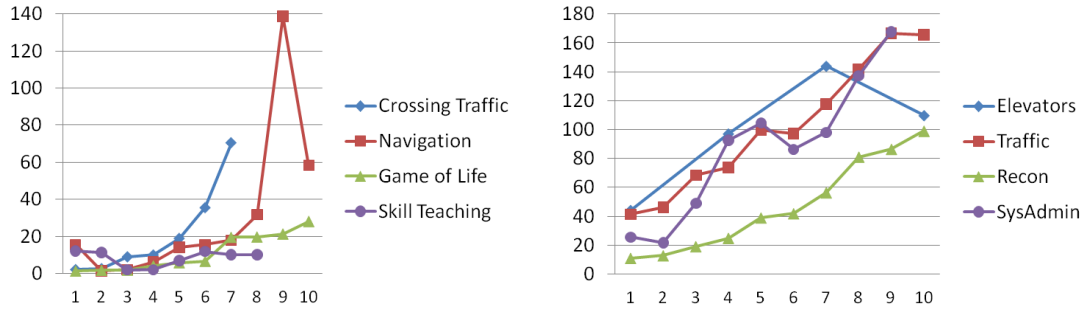
Table 5.1. Parameters Used to Solve Each Domain (Particular Instances if Specified in Parenthesis; All Instances Otherwise).

Domain / Instances	**State RBPF,** N	**McLUG Particles,** M	**McLUG Antecedents**	**Lookahead Depth,** D	**Lookahead Samples,** F
Crossing Traffic	8	4	4	4	4
Navigation (1)	64	64	64	40	1
Navigation (2-3)	1	1	1	30	1
Navigation (4-10)	32	32	32	20	2
Game of Life	2	1	1	0	1
Traffic	4	1	1	2	2
SysAdmin	32	1	1	0	2
Skill Teaching (1)	16	16	1	20	2
Skill Teaching (2)	16	16	16	20	2
Skill Teaching (3-10)	8	4	4	4	4
Elevators (1-7)	8	4	4	10	10
Elevators (8-10)	32	32	32	20	2
Recon	4	1	1	2	2

crossing traffic, robot movement is deterministic. The only exception is the possibility of the robot disappearing while traveling across the field. The probability distribution of disappearing in the field is not uniform; therefore, it is to the robot's advantage to strategically choose where to cross. The difficulty of this domain comes primarily from the size of the field. Figure 5.1b shows the success of POND-Hindsight on this domain.

Game of life consists of locations on a grid that are alive, that thrive or fail depending on locations surrounding them which the planner has control in setting as live. With the expectation that the best policy for this domain is the random action policy, a lookahead horizon of 0 and a McLUG heuristic with only 1 sample were used. In some cases, this performed better than the random action policy, in some cases worse, but for the most part very similar to random (see Figure 5.1c).

Traffic consists of cars lining up at intersections, with the planner choosing which lights to change to green. SysAdmin resembles a network of computers that periodically fail and need to be reset by the planner. Skill teaching attempts to educate students by asking questions and giving hints. All three of these domains are shown in Figures 5.1d, 5.1e,



(a) Quick Domains

(b) Slow Domains

Figure 5.2. Total time (in minutes) to solve 30 trials for each of the 10 instances of each domain

and 5.1f respectively, to have performed close to random.

Elevators requires a set of elevators to pick up passengers and then drop them off at either the top or bottom floor depending on their destination. Recon is similar to the mars rover scenario wherein a robot has the task of sampling rocks but may receive damage in the process and needs to perform repairs. POND-Hindsight never performed better than random and noop in these two cases (see Figures 5.1g and 5.1h).

5.2 Discussion

It is likely that failure to perform better than random and noop comes from the greedy nature of the lookahead algorithm. A greedy best-first search is primarily influenced by the heuristic. The McLUG heuristic was designed for goal-based problems. Although all domains are reward-based, crossing traffic and navigation are actually reward-based translations of goal-based domains. It makes sense, then, that POND-Hindsight performed better on crossing traffic and navigation than on any of the other domains.

Times are shown in Figure 5.2. The biggest factors in computation time are: 1) the complexity of the problem, 2) the parameters used to solve the problem, and 3) policy reuse among trials. Instances can be solved much quicker when most or all of the policy can be reused in subsequent trials.

Table 5.2. Competition Results for the POMDP Track of IPPC-2011 - Ordered from First to Last Place

Planner	**Score**	**Standard Error**
POMDPX NUS	0.590	± 0.098
KAIST AILAB	0.420	± 0.101
HyPlan	0.168	± 0.074
POND-Hindsight	0.152	± 0.058
Symbolic Perseus	0.117	± 0.064
McGill	0.034	± 0.031

IPPC-2011 results are shown in Table 5.2. Two planners greatly outperformed the rest. Of the remaining planners, however, POND-Hindsight is almost at the top. It is likely that a little future work would bring POND-Hindsight up to 3rd place or higher. As it stands, POND-Hindsight performs comparable to other competitive POMDP solvers to date.

CHAPTER 6

CONCLUSION

While POND-Hindsight performed well on crossing traffic and navigation, some work needs to be done to get it to succeed for the remaining domains.

The lookahead algorithm should use A* search instead of greedy best-first search. With a poor heuristic that often plateaus, this would result in more of a breadth-first search which would take significantly longer to complete than greedy best-first search. However, with a good heuristic, A* has the potential to reach the horizon almost as quickly as greedy best-first search but with a much greater chance of finding a good plan.

Some problems, such as crossing traffic and navigation, have absorbing states. Once one of these states has been reached, there is no need to perform any more online planning – a random or noop policy is just as sufficient. Because observations are sampled, it is important that POND-Hindsight performs lookahead by producing non-stationary policies, just like FF-Hindsight. This allows different actions to be performed, at later times, from the same belief state. However, by simply recognizing absorbing states as ones that have only one possible outcome, and that the outcome leads back to itself, online planning can end earlier and significantly cut down on computation time.

Extending the concept of hindsight optimization from MDPs to POMDPs is an appealing approach to applying the advances in planning heuristics and search algorithms. While the competition results are promising in a few domains, additional work is required to fulfill the potential of hindsight optimization in POMDPs.

REFERENCES

[1] Albore, A., Palacios, H., and Geffner, H. A translation-based approach to contingent planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (2009), Morgan Kaufmann Publishers Inc., pp. 1623–1628.

[2] Bellman, R. A markovian decision process. Technical report, DTIC Document, 1957.

[3] Boutilier, C., Dean, T., and Hanks, S. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research 11*, 1 (1999), 94.

[4] Bryce, D., Kambhampati, S., and Smith, D. Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research 26*, 1 (2006), 35–99.

[5] Bryce, D., Kambhampati, S., and Smith, D. Sequential monte carlo in reachability heuristics for probabilistic planning. *Artificial Intelligence 172*, 6-7 (2008), 685–715.

[6] Domshlak, C., and Hoffmann, J. Fast probabilistic planning through weighted model counting. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling* (2006), pp. 243–251.

[7] Doucet, A., De Freitas, N., Murphy, K., and Russell, S. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence* (2000), Citeseer, pp. 176–183.

[8] Draper, D., Hanks, S., and Weld, D. Probabilistic planning with information gathering and contingent execution. In *Proceedings of the 2nd International Conference on AI Planning Systems* (1994), Citeseer, pp. 31–36.

[9] Hansen, E., and Zilberstein, S. LAO: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence 129*, 1–2 (2001), 35–62.

[10] Hoffmann, J. FF: The fast-forward planning system. *AI magazine 22*, 3 (2001), 57.

[11] Hoffmann, J., and Brafman, R. Conformant planning via heuristic forward search: A new approach. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling* (2004), pp. 355–364.

[12] Hoffmann, J., and Brafman, R. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of 15th International Conference on Automated Planning and Scheduling* (2005), pp. 71–80.

[13] Little, I., and Thiébaux, S. Probabilistic planning vs replanning. 2007. http://www2.parc.com/isl/members/minhdo/icaps07_ws/papers/ICAPS06LittleI.pdf. September 2011.

[14] Majercik, S., and Littman, M. MAXPLAN: A new approach to probabilistic planning. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning* (1998), pp. 86–93.

[15] Majercik, S., and Littman, M. Contingent planning under uncertainty via stochastic satisfiability. In *Proceedings of the National Conference on Artificial intelligence* (1999), John Wiley & Sons Ltd, pp. 549–556.

[16] McDermott, D. AIPS98 planning competition results. 1998. ftp://ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html. July 2011.

[17] Peot, M. *Decision-theoretic planning*. PhD thesis, Stanford University, 1998.

[18] Pineau, J., Gordon, G., and Thrun, S. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence* (2003), vol. 18, Citeseer, pp. 1025–1032.

[19] Sanner, S., and Yoon, S. ICAPS 2011 international probabilistic planning competition (IPPC). 2011. http://users.cecs.anu.edu.au/∼ssanner/IPPC_2011. July 2011.

[20] Yoon, S., Fern, A., and Givan, R. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling* (2007), pp. 352–359.

[21] Yoon, S., Fern, A., Givan, R., and Kambhampati, S. Probabilistic planning via determinization in hindsight. In *Proceedings of the 23rd National Conference on Artificial Intelligence* (2008), vol. 2, AAAI Press, pp. 1010–1016.

[22] Yoon, S., Ruml, W., Benton, J., and Do, M. Improving determinization in hindsight for online probabilistic planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling* (2010), pp. 209–216.