POND-Hindsight: Applying Hindsight Optimization to POMDPs

Alan Olsen and **Daniel Bryce**
alan@olsen.org, daniel.bryce@usu.edu
Utah State University
Logan, UT

Abstract

We present the POND-Hindsight entry in the POMDP track of the 2011 IPPC. Similar to successful past entrants (such as FF-Replan and FF-Hindsight) in the MDP tracks of the IPPC, we sample action observations (similar to how FF-Replan samples action outcomes) and guide the construction of policy trajectories with a conformant (as opposed to classical) planning heuristic. We employ a number of technical approaches within the planner, namely we i) translate expected reward to a probability of goal satisfaction criterion, ii) monitor belief states with a Rao-Blackwellized particle filter, and iii) employ 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 are used as the value of each action.

Introduction

Some of the most successful entries in the MDP track of the IPPC are based on the idea of using classical planners to evaluate deterministic futures, and either execute the plans (Yoon, Fern, and Givan 2007) found in this manner, or evaluate the average quality of several futures prior to selecting an action (Yoon et al. 2008). In adapting the idea to a POMDP competition setting, there are several unique challenges/opportunities: 1) POMDPs allow observation uncertainty in addition to action outcome uncertainty; 2) the POMDP instances are formulated with rewards instead of goals (as in past MDP tracks); and 3) the POMDP instances are finite horizon and not discounted.

Our approach in POND-Hindsight is 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, we are still planning in belief state space, and make use of a reachability heuristic for belief state space based on the McLUG (Bryce, Kambhampati, and Smith 2008). 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.

We translate rewards 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 where 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 where the goal proposition is true. To more accurately track the rewards, we use Rao-Blackwellized particles where 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 we also use Rao-Blackwellized particles 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. We limit the depth of the futures explored for each action to reduce search cost, and use the McLUG heuristic to bias the search.

In the following, we detail the translation of a POMDP to a goal-based belief state MDP, the POND-Hindsight algorithm, empirical results for the competition domains, related work, and a conclusion and future work.

Framework

We base our work within two models of acting in partially observable, stochastic worlds: expected discounted reward POMDPs and goal-based POMDPs.

Expected Discounted Reward POMDPs

We consider the POMDP model $(S, A, O, b_0, T, R, \Omega, \gamma)$, where

- S is a finite set of states
- A is a finite set of actions
- O is a finite set of observations

- $b_0(s) = Pr(s_0 = s)$, an initial belief distribution
- $T(s, a, s') = Pr(s_{t+1} = s' | a_t = a, s_t = s)$ is a state transition relation
- $R(s, a)$ is the reward for applying a in s
- $\Omega(o, s, a) = Pr(o_{t+1} = o | a_t = a, s_{t+1} = s)$ is the probability of observing o in s after applying a
- γ is a discount factor, where $0 \leq \gamma < 1$

The belief state after a given history of actions and observations is defined:

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

and is computed incrementally from a prior belief state, given an action and observation, so that:

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

where α is a normalization constant. 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

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

A policy π is a function $\pi(b_t) = a_t$ defining which action to take, given the belief state, so that the expected discounted reward attainable from a belief state (its value) is defined:

$V_\pi(b) = R(b, \pi(b)) + \gamma \sum_{o \in O} T(b, \pi(b), b_{\pi(b)}^o) V_\pi(b_{\pi(b)}^o)$

The value of a policy is defined by $V_\pi(b_0)$.

Translation to Goal-Based POMDPs

As defined by Majercik and Littman (2003), 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

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

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 states on the lookahead horizon as goal states, during 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 state is revisited. We even preserve the policy 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)
```

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
```

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.

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. As part of the particle filter, we sample one observation based on its probability of being generated after applying a to b. We represent the actions as a dynamic Bayesian network (DBN), and compute the successor belief state by sampling the original state propositions but analytically computing the distribution over the goal proposition. 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.

Another improvement, that we report 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. As was discovered by Hoffmann and Brafman (2004), limiting the number of antecedents of conditional effects significantly reduces the cost of heuristic construction in conformant planning heuristics.

Results

A portion of the benchmarks presented in this section are actual results from IPPC-2011. Parameters 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. So 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.

There were 8 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 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 an all-"noop" policy and a random action policy, also averaged over 30 trials, and shown in Figure 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 the best on this domain, as can be seen in Figure 1a.

Navigation requires a robot to cross a sort of mine field to get to its goal. As with 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 in the size and shape of the field. Figure 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 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 1d, 1e, and 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 where 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 1g and 1h).

It is likely that failure to perform better than random and noop comes from the greedy nature of the lookahead algorithm.

Times are shown in Figure 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.

Related Work

Work by Yoon, Fern, and Givan (2007) and Yoon et al. (2008) has shown that deterministic translations of probabilistic problems can be used to quickly find approximate solutions to the original problem.

FF-Replan

FF-Replan generates partial policies for MDPs in an online manner (Yoon, Fern, and Givan 2007). 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 (Hoffmann 2001) - to generate a policy trajectory. FF-Replan follows the generated policy trajectories during online search, where possible,

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
Traffic	4	1	1	2	2
Game of Life	2	1	1	0	1
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

Table 1: Parameters used to solve each domain (particular instances if specified in parenthesis, all instances otherwise)

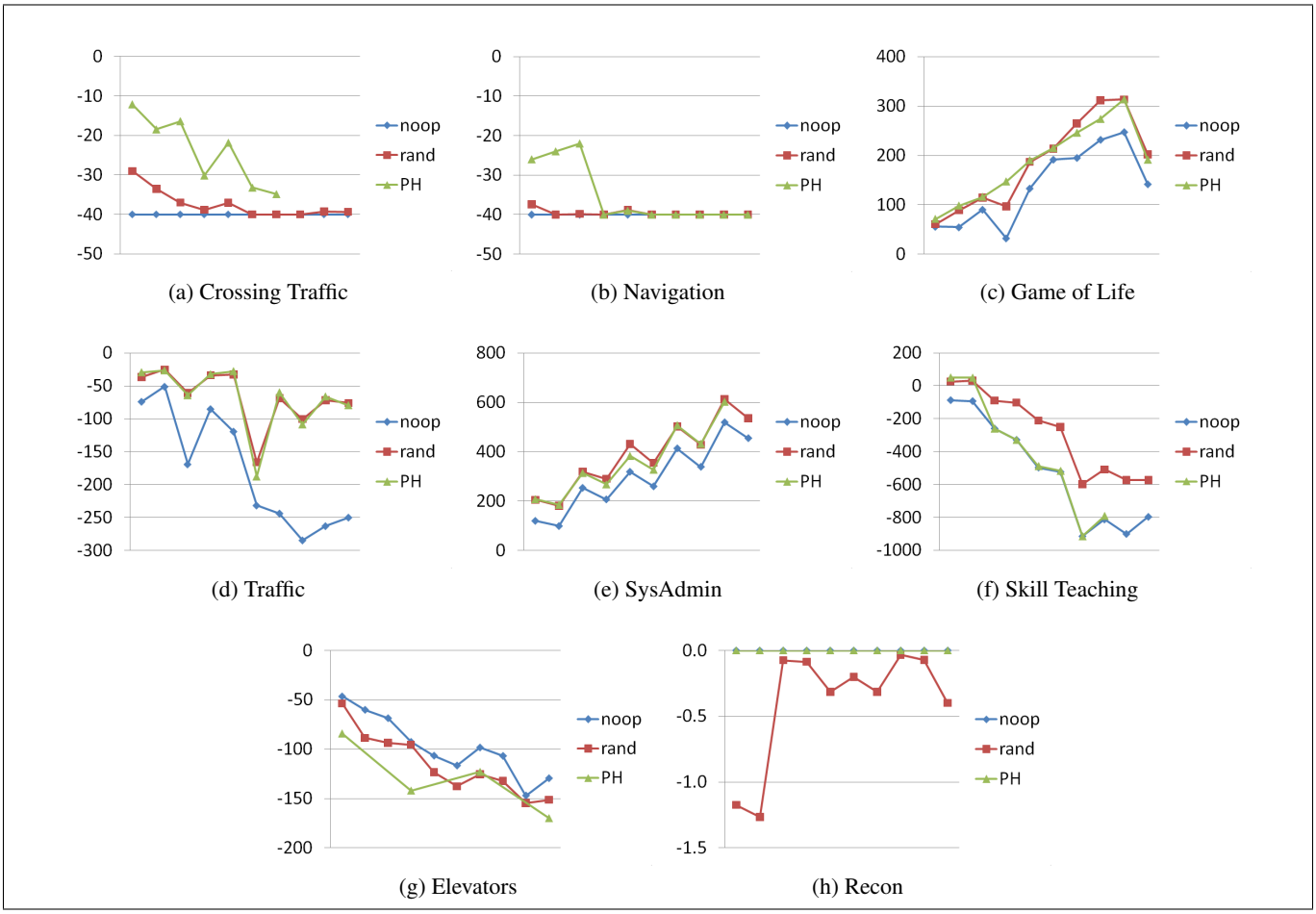
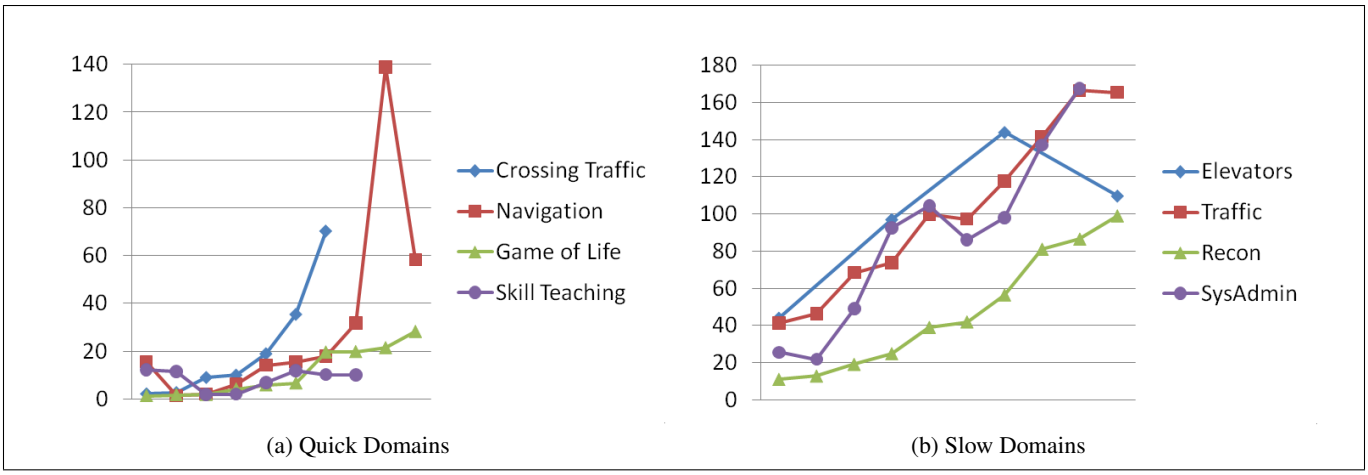


Figure 1: Rewards for 10 instances, along the horizontal axis, of each domain (averaging over 30 trials for each instance)



(a) Quick Domains (b) Slow Domains

Figure 2: Total time (in minutes) to solve 30 trials for each of the 10 instances of each domain

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 (since the author was also head of the competition). The paper (Yoon, Fern, and Givan 2007) 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 (Little and Thiébaux 2007), 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.

FF-Hindsight

Hindsight Optimization (HOP) (Yoon et al. 2008), later called FF-Hindsight (Yoon et al. 2010), improves upon FF-Replan by utilizing the probabilistic information of the problem. This time the generation of a policy trajectory obeys a "future" which 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+ (Yoon et al. 2010). 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.

Future Work

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 at finding a good plan.

Common random numbers should be shared among futures in the lookahead algorithm, as was done for FF-Hindsight.

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's 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.

Conclusion

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 HOP in POMDPs.

Acknowledgments

This work was supported by DARPA contract HR0011-07-C-0060.

References

Bryce, D.; Kambhampati, S.; and Smith, D. 2008. Sequential monte carlo in probabilistic planning reachability heuristics. *AIJ* 172(6-7):685–715.

Hoffmann, J., and Brafman, R. 2004. Conformant planning via heuristic forward search: A new approach. In *Proceedings of ICAPS'04*, 355–364.

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

Little, I., and Thiébaux, S. 2007. Probabilistic planning vs replanning. In *ICAPS Workshop on IPC: Past, Present and Future*. Citeseer.

Majercik, S., and Littman, M. 2003. Contingent planning under uncertainty via stochastic satisfiability. *AIJ* 147(1-2):119–162.

Yoon, S.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic planning via determinization in hindsight. In *Proceedings of Conference on Artificial Intelligence (AAAI)*.

Yoon, S.; Ruml, W.; Benton, J.; and Do, M. 2010. Improving Determinization in Hindsight for Online Probabilistic Planning.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *17th International Conference on Automated Planning and Scheduling (ICAPS-07)*, 352–359.