FF-Replan: A Baseline for Probabilistic Planning

Sungwook Yoon
Computer Science & Engineering
Arizona State University
Tempe, AZ 85281
Sungwook.Yoon@asu.edu

Alan Fern
Computer Science Department
Oregon State University
Corvallis, OR 97331
afern@cs.orst.edu

Robert Givan
Electrical & Computer Engineering
Purdue University
West Lafayette, IN 47907
givan@purdue.edu

Abstract

FF-Replan was the winner of the 2004 International Probabilistic Planning Competition (IPPC-04) (Younes & Littman 2004a) and was also the top performer on IPPC-06 domains, though it was not an official entry. This success was quite surprising, due to the simplicity of the approach. In particular, FF-Replan calls FF on a carefully constructed deterministic variant of the planning problem and selects actions according to the plan until observing an unexpected effect, upon which it replans. Despite the obvious shortcomings of the approach and its strawman nature, it is the state-of-the-art in probabilistic planning as measured on recent competition benchmarks. This paper gives the first technical description of FF-Replan and provides an analysis of its results on all of the recent IPPC-04 and IPPC-06 domains. We hope that this will inspire extensions and insight into the approach and planning domains themselves that will soon lead to the dethroning of FF-Replan.

Introduction

Probabilistic planning is naturally formulated using Markov Decision Processes (MDP) (Puterman 1994) and many probabilistic planning techniques have been developed based on MDP formulations. When the first international probabilistic planning competition (Younes & Littman 2004a) was held in 2004, as expected, most of the entries were based on MDP solvers. Surprisingly enough, the winner of the IPPC-04 was a planner based on deterministic planning techniques, FF-Replan. In IPPC-06 (Bonet & Givan 2006), FF-Replan was not entered, but tests reported here shows it outperforms all entrants in the probabilistic track. FF-replan is thus the currently reigning state-of-the-art technique for probabilistic planning problems as measured on planning competition benchmarks.

FF-Replan has obvious shortcomings. FF-Replan first determinizes the input domain, removing all probabilistic information from the problem, and then synthesizes a plan. During the execution of this plan, should an unexpected state occur, the planner replans in the same determinization of the problem. Execution and replanning continue until a goal is reached. FF-replan is similar in style to contingent planning (Geffner 1998), although it does not explicitly generate a plan tree. FF-Replan does neither contingent planning nor conformant planning. FF-Replan does not consider the multiple potential effects of an action and this is the biggest weakness of the approach. On the other hand, we believe that the weaknesses of FF-Replan can motivate research that compensates for them while maintaining the merits of the FF-Replan system.

Probabilistic planning and deterministic planning have historically taken different approaches although they have similar goals. Leveraging MDP formulations, probabilistic planning has typically focused on value functions or policies for the entire state space, while deterministic planning developed intelligent heuristic functions enabling exploration of a restricted portion of the state space that is considered to be enroute to the goal state. By relying on search, state of the art deterministic planners function with enormous speed and efficiency—this is the key to the moderate success of FF-Replan.

Cross-fertilization between the deterministic and probabilistic planning communities has been challenging but steady (Mausam, Bertoli, & Weld 2007; Meuleau & Smith 2003). For example, in the probabilistic planning community, (Boutilier, Brafman, & Geib 1998) used reachability analysis from deterministic planning to prune out unnecessary Bellman backup operations. (Karabaev & Skvortsova 2005) considered efficient state space enumeration based on a plangraph reachability analysis. Another very direct application of deterministic planning techniques to probabilistic problems was conducted by Pgraphplan (Blum & Langford 1999). In the deterministic planning community, probabilistic planning techniques have been applied to find planning domain “solvers” that find control information for planning domains, for example using approximate policy iteration (Fern, Yoon, & Givan 2006). We hope that FF-Replan’s success will promote further cross-fertilization, especially the use of deterministic planning techniques in probabilistic planning.

In what follows, we first overview the probabilistic planning competitions. Next we describe the architecture and variations of FF-Replan and present its performance on the domains from the two planning competitions. We discuss the shortcomings of FF-Replan and potential improvements to FF-Replan that can overcome these shortcomings.

The Probabilistic Planning Competitions

There have been two International Probabilistic Planning Competitions: the first in 2004 run by Michael Littman and Hakan Younes (`www.cs.rutgers.edu/~mlittman/topics/ipc04-pt`) and the second in 2006 run by Blai Bonet and Bob Givan (`www.ldc.usb.ve/~bonet/ipc5`). Each has been associated with the International Conference on Automated Planning and Scheduling in the same year. Problems for these competitions have been specified in the Probabilistic Planning Domain Description Language (PPDDL) (Younes & Littman 2004b), which defines a probabilistic planning problem as a set of probabilistic action schemas together with an initial state and a goal description. Each component is specified exactly as in PDDL (McDermott 1998), the widely used deterministic planning language, except that there is one newly available probabilistic effect for action definitions

$$(\textit{probabilistic} \quad p_1 \;\; e_1 \quad \ldots p_n \;\; e_n)$$

where each e_i is a PPDDL effect (either probabilistic or deterministic) that occurs with probability p_i whenever the overall probabilistic effect occurs. A probabilistic effect represents a distribution over deterministic effects. The definition is recursive with the base case being when an e_i is a deterministic effect, in which case the probability of e_i is just p_i. The recursive case is when e_i is also a probabilistic effect, in which case we consider the probability distribution over deterministic effects defined by e_i and multiply those probabilities by p_i. Note that the probabilistic construct can also be used to specify an initial state distribution.

The planners in each competition were evaluated in an online planning setting over the internet by connecting to a competition host server (Younes & Littman 2004a; Bonet & Givan 2006) that acted as an environment simulator. The planners were evaluated on a fixed set of planning domains using a fixed set of problems from each domain. For each problem, the server first sent the planner the initial state and goal. The planner-server interaction then alternated between the planner sending an action and the server sending an updated state to the planner. For each problem, the planners were given a fixed real-time limit, in which they would attempt to solve the problem for up to thirty trials. The planners were evaluated on a per problem basis according to the fraction of trials that reached the goal (out of thirty) and the average number of actions selected on successful trials.

FF-Replan - The Architecture

FF-Replan is an action selection algorithm for online planning in probabilistic domains. FF-Replan has a very simple architecture. Given a new probabilistic planning problem, consisting of a domain description, a goal, and initial state, FF-Replan first generates a deterministic planning problem as described below. FF-Replan then uses the deterministic planner FF (Hoffmann & Nebel 2001) to compute a totally-ordered plan for the generated deterministic problem. During execution of the resulting plan, if confronted with an unexpected state, the process is repeated with the unexpected state as the initial state, until a goal state is reached. Note that the determinization process is conducted once before execution begins and there is a potential improvement of the system by considering adaptive determinization.

Determinization

FF-Replan determinizes the input domain at the start of planning. It converts the probabilistic domain definition into a deterministic domain, different from the input domain. This simple idea turns out to be quite useful for many competition domains.

We have tried two straightforward methods of determinization. The first is *single-outcome determinization*, which selects one outcome for each probabilistic construct. Among $e_1 \quad \ldots \quad e_n$, the determinization process selects one e_i, otherwise ignoring the probabilities p_i and other effects e_i. Probabilistic effects can be nested and the determinization process recursively selects one effect among the nested probabilistic constructs. The single-outcome determinization process produces one deterministic action for each probabilistic action. One can design several heuristic methods for selecting the outcome to keep—e.g. choosing the most likely effect or choosing the probabilistic effect with the most add effects are central candidates. A key problem with the single-outcome approach is that the performance of the replanner depends critically on the method for selecting the effect used and that any choice may neglect important possible effects. For any such outcome-selection heuristic, one can easily design domains where the resulting planner will perform poorly. The first variant of FF-Replan, that participated in and won IPPC-04, used the single-outcome approach, using a heuristic that selected the outcome with highest probability. In our experimental results, we compare this variant with the later variant described below, across the competition domains from both competitions.

The second approach, which the current FF-Replan takes as its determinization process, is *all-outcomes determinization*, which considers every probabilistic outcome as a distinct deterministic action. For each possible outcome e_i, FF-Replan generates one separate action for each effect. If e_i is deterministic then the action corresponding to e_i is also deterministic. Otherwise for probabilistic e_i (i.e. nested probabilities) we recursively apply the all-outcomes procedure. Thus, the number of deterministic actions is exponential in the depth of probabilistic nesting. In practice, this nesting is shallow and has not caused a problem with FF-Replan in the first two competitions.

Since this approach considers every possible probabilistic outcome, the performance does not depend on the choice of the outcome of each action as is the case for the previous idea. While the theoretical properties of this approach are weak, one can at least say that with all-outcomes determinization, FF-Replan will select an action a in state s only if a is the first action of some sequential plan that has non-zero probability of reaching the goal from s. This also implies that FF-Replan will cease selecting actions whenever it enters a dead-end state. However, a key potential weakness of this approach is that the planner is given no information about probabilistic effects and rather treats them all as equal. This means that FF-Replan will not explicitly attempt to avoid actions that have a non-trivial chance of leading to dead-ends or moving away from the goal. One can easily develop domains that exploit this weakness.

Both determinization approaches are implemented by storing a mapping from the generated deterministic actions back to the underlying (probabilistic) actions from which they were created. Then FF-Replan maps the deterministic actions back to the available probabilistic actions during plan execution as it communicates with the server. Considering probabilistic outcomes as separate deterministic outcomes has been discussed before in the MDP community (Boutilier, Dearden, & Goldszmidt 2000). In that work, determinization is used in combination with probabilistic choice (nature's selection) to facilitate a dynamic programming formulation.

FF-Replan's determinization has some similarity to hindsight optimization (Wu, Chong, & Givan 2002). In hindsight optimization, the future sequence of nature's choices is drawn from a simulator and used to provide a non-stationary determinization of the actions. Planning under this non-stationary determinization provides an estimate of the distance to the goal. The mean value of this estimate over multiple simulations underestimates the true distance and can be used as a heuristic to guide action selection. This strategy performed quite well in practice as reported in (Mercier & Van HenTenRyck 2007). FF-Replan's determinization strategy is similar to hindsight optimization but FF-Replan's strategy is more aggressive. Not only does FF-Replan assume the knowledge of the future (since the domain is deterministic), but also FF-Replan is allowed to select the most helpful outcome, especially if the deterministic planner is optimal. When the most helpful outcomes correspond well to reality, which is often the case in real problems (i.e. the desired effect is quite likely), the all-outcomes determinization approach will often perform well. We will discuss relaxing FF-Replan's strategy toward hindsight optimization in the discussion section.

Planning

FF-Replan maintains a partial state-action mapping using a hash-table which is initially empty. When FF-Replan encounters a state that is not in the table, then it determinizes the problem and synthesizes a plan using FF. FF-Replan then simulates the plan according to the deterministic action definitions resulting in a state-action sequence whose pairs are put in the hash table. The first action of the plan is then executed in the environment, which returns a new current state. FF-Replan thus produces a partial policy in an online fashion. Of course due to the deterministic approximation, the partial policy has no quality guarantees in general. One direction for future improvement might be to adaptively repair the partial policy—for example, by attempting to break out of loops.

Standard planning approaches propositionalize existential goals as disjunctions over instantiations with the available objects. For example, if the goal is stacking all the blocks in the Blocksworld, although the quantified goal is a single sentence, $\exists x \exists y \ldots \exists z \ \ (\text{on} \ \ x \ \ y) \wedge \ldots \wedge (\text{on} \ \ . \ \ z)$, the propositionalized goal is a disjunction of any stacked blocks, which is factorial in the number of blocks involved. The resulting disjunction of the instantiated goals is sound in theory but in practice, state of the art planners have difficulty dealing with such goals, due to the popular focus on exploiting conjunctive goal structure. Because the IPPC competitions have used such goals, FF-Replan uses a very simple initial approach to this problem. Rather than deal with the quantified goal, FF-Replan picks an arbitrary grounded form of any existential goal. The obvious pitfall of this approach is that some groundings of the goal are not reachable or are much more expensive to reach from the initial state. Simple sampling of the grounded goals and testing with relaxed reachability analysis would be an immediate but untried improvement on this strategy. More extensive improvements handling this difficult problem should also be considered in the future.

Competition Results

Here we consider the competition results from IPPC-04 and IPPC-06 (Younes & Littman 2004a; Bonet & Givan 2006) extended with new experiments to include both variants of FF-Replan. Figure 1 contains a brief description of each of the domains used in these competitions.

IPPC-04

Figure 2 gives the results for IPPC-04. Each column corresponds to a planner and each row corresponds to a problem. In the competition, 30 trials per problem were conducted with a total time limit of 15 minutes for the 30 trials. Note that this time includes the communication time and the planning time. Each cell of the figure gives two numbers. The first is the number of successful trials for each planner and problem pair and the second number in parens is the average time required to achieve the goal in seconds on successful trials. The columns labeled FFR$_s$ and FFR$_a$ correspond to the single-outcome and all-outcomes variants of FF-Replan respectively. Note that FFR$_a$ was not an official entry in IPPC-04 and was evaluated on these problems for the purposes of this paper. Also note that the planning domain definitions, but not specific problems, had been released prior to the competition and participants in the learning and control-knowledge tracks were allowed to learn from the domain description or provide control knowledge respectively. The planners shown in boldface in the table used human provided control knowledge. Classy was the only learning system.

Blocksworld and Boxworld Results. The problems starting with "bw" are Blocksworld variants and problems starting with "bx" are Boxworld variants. Both domains have probabilistic actions that fail to achieve the intended effect but the failures do not lead to dead-end states or significant departures from the expected state. These domain characteristics are ideal for our replanning approach. In particular, the highest-probability heuristic used by FFR$_s$ is the correct heuristic in this domain since the intended effects are always more probable. Accordingly in these domains the FF-Replan variants dominate all of the regular entries. Compared to planners that used control knowledge (NMRDPP+Control Knowledge (Gretton & Thiebaux 2004) and J1 (Yoon, Fern, & Givan 2004)), FFR$_s$ was slightly worse on two problems.

FFR$_s$ did fail to solve some of the larger sized Blocksworld variants. The primary reason appears to be that FF took too much time solving the deterministic versions generated by FFR$_s$ compared to the problems generated by FFR$_a$. This behavior of FF is somewhat unexpected since the actions used by FFR$_s$ are a subset of those used by FFR$_a$. Our best explanation of this is that some of the failure outcomes considered by FFR$_s$ act as a sort of macro action that allows FF to find plans more quickly. For example, the failure mode of the pickup action is for the block to end up on the table, which would normally require two actions. Thus the deterministic action corresponding to the failure outcome of pickup can be used to quickly get a block on the table. Of course while this behavior might lead to computational speedups, the resulting heuristic is likely to be less accurate.

Both variants of FF-Replan performed very well on Boxworld variants, as its base planner FF is strong on Logistics style domains.

Remaining Problems. The exploding-block, g-tire-problem, and toh-prob-pre have actions that can lead to the dead-end states and need careful planning that minimizes the involvement of such actions. These types of domains are among the worst case scenarios for both of our approaches. For FFR$_s$ the "dead end outcomes" are not selected by the determinization heuristic, so the planner is completely unaware of them. For FFR$_a$ the deterministic planner can optimistically choose only the desired outcomes required to reach the goal. Both variants of FF-Replan failed on most of the trails. Note, however, that for the exploding-block domain all other planners fail and FFR$_a$ is the only planner that had any successful trials. Thus, even the planners that explicitly reason about probabilities are not yet up to the challenge of this domain. For ztravel we again see a significant difference between FFR$_a$, which solves all trials, and FFR$_s$ which always fails. Here the outcome selection heuristic results in deterministic problems that are not solvable, explaining the failure of FFR$_s$. Overall we see that FFR$_a$ outperforms FFR$_s$ showing that the outcome selection heuristic is not as effective in these domains as planning according to the most optimistic potential outcomes, showing the utility of this novel determinization approach.

Finally, the average time results show that FF-Replan is a practical approach in these domains. Note that the current implementation is not optimized and there are many directions for optimization. For example, rather than create a deterministic plan from scratch each time an unexpected outcome occurs, it would be useful to investigate incremental planning approaches that leverage the most recent deterministic plan.

IPPC-06

Figures 3, 4, 5 show the competition results from the IPPC-06 probabilistic track. There were 4 official participants. FFR$_a$ was an unofficial strawman entry and performed better than the winner FPG (Buffet & Aberdeen 2006). There were 9 domains in IPPC-06 and there were 15 problems for each domain, thus there were 135 problems total. For each problem, the participants were asked to solve 30 trials as in IPPC-04. Unlike IPPC-04, no participant utilized control knowledge or learning.

Figure 3 shows the percent of successful trials for each domain. FFR$_a$ and FPG were the only performers who solved at least one or more problems for each domain and both of these planners exhibit more robust performance across domains than the other planners. Compared to the winner FPG, FFR$_a$ has a higher success rate in all domains with the exception of schedule. Overall we see that FFR$_a$ was the top performer, often by significant margins, in 5 of the 9 domains. Furthermore, across all domains it was never worse than second best.

There are four dead-end free domains, which are the ones that we might expect FFR$_a$ to perform well on. Indeed, for two of them, Zeno World and Random, FFR$_a$ performed the best, as these domains are well suited to the replanning idea. However, for the other two domains, Blocksworld and Elevators domain, FFR$_a$ was second best. The primary reason for the worse performance in Blocksworld and Elevator, appears to be that the deterministic planning time for the larger problem instances was significant. This is particularly a problem since currently FF-Replan does not reuse previous planning effort.

Among the 5 non-dead-end-free domains, on 3 domains, Exploding Blocksworld, Drive and Pitchcatch, FFR$_a$ performed the best. We think that the primary reason for the replanners' success here is its scalability compared to the MDP based techniques. One noticeable feature of these results is that the performance of FF-Replan on the exploding blocksworld domain in IPPC-04 was much worse than in IPPC-06. After some investigation we found that the exploding blocksworld problems in IPPC-06 are significantly easier than the ones in IPPC-04 in the sense that it is not nearly as important to consider the "exploding outcome" that results in a dead end. For the rest of the non-dead-end-free domains, FFR$_a$ was close second. We were at first surprised at the high success rate of more than 80% for FFR$_a$ in Tireworld. A closer investigation revealed that many of the instances were not too hard in the sense that there was no need for careful route selection. For other problems, where careful route selection was necessary, FFR$_a$ appeared to be fortunate and not get flat tires.

Both the results in exploding blocksworld and Tireworld suggest that it can be difficult to define problem distributions that properly exercise the complexity of probabilistic domains. One idea for future competitions would be to explicitly design some of the problem distributions with the aid of a replanner to help ensure that the replanner does not do well.

Figure 4 shows the average number of actions used on successful trials in each domain. In some domains we see that FF-Replan performs significantly worse than planners that solve a similar number of problems. For example, in the blocksworld variant, FOALP (Sanner & Boutilier 2006) solves all of the problems and only requires approximately

		Domain Description
IPPC 2004	**Blocksworld**	Similar to the traditional Blocksworld except that there is a probability of dropping a block on the table and the blocks are colored. The goal is to build a tower with a specified color pattern, rather than to achieve a tower containing particular blocks as is more usual.
	Boxworld	Similar to traditional logistics world except that there is a small chance of driving to an unintended city. The domain is dead-end free.
	exploding-block	Similar to the Blocksworld but with some probability putting down a “not-detonated” block destroys the target (either table or a block) and detonates the block. When the table or a goal block is destroyed, we reach a dead-end. Putting down a detonated block does not cause any problem. The best strategy is to putdown a block onto a non-goal block until it detonates. and then build the target tower.
	file-prob-pre	Every file needs to be placed into a folder of its type. The type of a file is assigned uniformly among available types when get-type action is conducted.
	g-tire-problem	There are two routes. One route has spare tires at some locations. The other does not. There is a small chance of having a flat tire when driving. When the tire is flat and there is no spare, the state is a dead-end.
	r-tire-problem	Similar to g-tire-problem. But can call AAA for spares, with penalty. Dead-end free domain.
	toh-prob-pre	A variant of Tower Of Hanoi. Can move two disks at a time with bigger risk of ending up with a deadend state. Should not use two disks actions.
	ztravel-1-2	A variant of Zenotravel IPC3. Each original action has a “complete”-action that must be taken to realize the effects. Each “complete”-action has a very small probability of success. A planner needs to repeat the complete action until the effect is realized.
IPPC 2006	**Blocksworld**	Similar to Blocksworld of IPPC 2004. Additional actions of moving a tower of blocks is available, with smaller success chance than normal actions. Blocks do not have colors in this version.
	Tireworld	Similar to tire-problems of IPPC 2004. Need to consider more routes.
	Drive	Toyish representation of real-life driving. Need to reach a goal point. Roads are placed similar to chess board. Before the move-action, need to check the light and wait when the light is RED. The chance of having RED is different depending on the road type. There is a slim chance of dying on a wait action. Move-action can cause dying, depending on the length of the road.
	Pitchcatch	Need to catch balls of goal types. Each type can be caught when the pre-defined set of bits are set. Some bits can be set through “setbit” action with a big chance of dying. When a ball is thrown, one can catch or pass it. When caught, all the bits are reset. When hard to set bit is set and the ball is not the right type, one should pass and wait for another ball.
	Random	Domain itself is randomly generated with random precondition and effects from randomly generated predicate symbols. Based on the domain, a problem is generated by random walk from randomly generated initial state. Then the goal state is the resulting state of the random walk. Special reset-action can lead any state to the initial state, making it dead-end free.
	Exploding Blocksworld	Similar to IPPC 2004 Exploding Blocksworld domain. Need to reduce the number of putdown actions to raise the success probability. Unlike 2004 version, destroyed table does not immediately lead to a dead-end state.
	ZenoWorld	Similar to IPPC 2004 Zeno travel domain.
	Elevators	Need to collect coins. Coins are at some positions of some levels. Elevators can move along levels. At a level one needs to move to the correct positions to get the coins. The move action is probabilistic and can result in unwanted position with 50% chance. Thus, a planner needs to repeat the move actions to get to the positions with coins.
	Schedule	Toy simulation of packet scheduling problem. Goal is processing goal types of packets. Some types are delivered very rarely. Packets are delivered to the queue more frequently than one can process, and when the queue is overflowed, there is a chance of dying. Need to process those hard-to-come-by types of packets as they are delivered

Figure 1: Domain Description for IPPC 2004 and 2006: Bold faced domains are dead-end free, which is favorable for a replanner.

	Results from IPPC-04										
Domains	**NMR** ^{C}	mGPT	NMR	C	R	**FCP**	**J1**	Cla$'$	FFR$_s$	Pro	FFR$_a$
bw-c-pc-8	30 (1)	0 (-)	0 (-)	0 (-)	0 (-)	30 (26)	30 (1)	30 (1)	30 (1)	0 (-)	30 (0)
bw-c-pc-nr-8	30 (1)	0 (-)	0 (-)	0 (-)	0 (-)	30 (27)	30 (1)	30 (1)	30 (1)	0 (-)	30 (0)
bw-nc-pc-11	30 (4)	30 (0)	0 (-)	0 (-)	0 (-)	0 (-)	30 (1)	30 (1)	30 (1)	0 (-)	30 (0)
bw-nc-pc-15	30 (18)	30 (7)	0 (-)	0 (-)	0 (-)	0 (-)	30 (2)	30 (2)	0 (-)	0 (-)	30 (1)
bw-nc-pc-18	23 (38)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	30 (2)	15 (3)	0 (-)	0 (-)	30 (28)
bw-nc-pc-21	19 (45)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	30 (3)	30 (3)	30 (19)	0 (-)	30 (3)
bw-nc-pc-5	30 (0)	30 (0)	30 (0)	30 (1)	30 (2)	30 (3)	30 (0)	30 (0)	30 (0)	11 (67)	30 (0)
bw-nc-pc-8	30 (1)	30 (0)	0 (-)	0 (-)	0 (-)	30 (24)	30 (0)	30 (0)	30 (0)	0 (-)	30 (0)
bw-nc-pc-nr-8	30 (1)	0 (-)	0 (-)	0 (-)	1 (789)	30 (28)	30 (0)	30 (0)	30 (0)	0 (-)	30 (0)
bx-c10-b10-pc-n	30 (28)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	30 (3)	20 (4)	30 (3)	0 (-)	30 (0)
bx-c10-b10-pc	30 (29)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	30 (3)	20 (4)	30 (2)	0 (-)	30 (0)
bx-c15-b10-pc	14 (60)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	30 (4)	0 (-)	30 (3)	0 (-)	30 (0)
bx-c5-b10-pc	30 (7)	30 (6)	0 (-)	0 (-)	0 (-)	0 (-)	30 (1)	30 (1)	30 (1)	0 (-)	30 (0)
bx-c5-b10-pc-nr	30 (8)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	30 (1)	30 (1)	30 (1)	0 (-)	30 (0)
exploding-block	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	3 (0)	0 (-)	5 (0)
file-prob-pre	0 (-)	30 (2)	3 (290)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	14 (60)	0 (-)	29 (29)
g-tire-problem-	0 (-)	16 (0)	9 (0)	30 (1)	9 (1)	0 (-)	0 (-)	0 (-)	7 (0)	7 (1)	7 (0)
r-tire-problem-	0 (-)	30 (0)	30 (1)	30 (1)	0 (-)	0 (-)	0 (-)	0 (-)	30 (0)	6 (122)	30 (0)
toh-prob-pre	0 (-)	0 (-)	15 (0)	0 (-)	17 (4)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	11 (0)
ztravel-1-2	0 (-)	30 (0)	30 (9)	30 (18)	27 (32)	0 (-)	0 (-)	0 (-)	0 (-)	1 (0)	30 (0)

Figure 2: IPPC-04 competition results. Each column corresponds to a planner and each row corresponds to a problem. The planners were run for up to 30 trials on each problem and given a total of 15 minutes for the 30 trials. Each cell first gives the number of trials for which the goal was achieved and then gives in parens the average time in seconds required for the successful trials. The planners are as follows: NMRDPP^C (NMR^C), NMRDPP (NMR), FCPlanner (FCP), J1 (Planner with Human Control Knowledge), Classy (Cla), FF-Replan with single-outcome determinization (FFR$_s$), Probapop (Pro), FF-Replan with all-outcomes determinization (FFR$_a$). Note that FFR$_a$ was not part of the original competition and these results were obtained for the purposes of this paper. Also note that planners shown in bold utilize human-provided control knowledge.

half as many actions as FFR$_a$. Pitchcatch and zeno are other domains where the average plan length of FFR$_a$ is apparently far from optimal. For other domains, FFR$_a$ produces significantly shorter solutions. For example, in schedule and random, the solutions are much shorter than those of FPG though both planners solve approximately the same number of problems.

Figure 5 shows the average time spent on successful trials in each domain. In most of the domains, FF-Replan spent less time than other performers, again showing the benefit of the determinization process. Note that the average time is only on solved problems and thus the numbers should be interpreted in conjunction with Figure 3.

Summary and Future Directions

The deterministic planning community has developed very efficient and informative planning heuristics. FF-Replan's performance on probabilistic planning is primarily due to these developments, allowing for fast deterministic plan synthesis. We have demonstrated that this approach is state-of-the-art when evaluated on problems from the first two probabilistic planning competitions. It is important to note that the impressive performance is a result of most competition problems not exercising the fully difficulty that is potential in the probabilistic setting. This suggests a more careful design process in future competitions, where there is an explicit attempt to span the spectrum of complexity ranging from deterministic problems to highly stochastic problems.

While FF-Replan has obvious shortcomings, our results have suggested many natural directions for improvement, some of which we have highlighted in this paper. Two such directions that we are currently pursuing are discussed in more detail below.

Hindsight Optimization. FF-Replan's approach bears much similarity to hindsight optimization (Wu, Chong, & Givan 2002). In the all-outcomes variant of FF-Replan, the deterministic planner effectively has the ability to select exactly the probabilistic outcomes that are most useful for achieving the goal, regardless of how likely those outcomes are. In comparison, the hindsight optimization approach is not able to select the future outcomes, but rather is told what the future outcomes will be effectively allowing it to plan in "hindsight". Given knowledge of the future outcomes, hindsight optimization formulates a plan contingent on those outcomes. In this sense, hindsight optimization is not as optimistic as FF-Replan.

An interesting future direction is to evaluate a hindsight optimization variant of FF-Replan. One approach to doing this is as follows. First, at the current state draw a random number sequence $r_1 \dots r_h$ until a predefined horizon h. The predefined horizon h can be increased as the planner fails to find any plan in the horizon. During the planning, the outcome for each action at future time i would be dictated by the random number r_i. Thus, the corresponding deterministic problem is non-stationary in the sense that the outcome selected for a particular action can change with the time step.

	Percent Successful Trials in IPPC-06)					
Domains	FFR$_a$	FOALP	sfDP	FPG	Paragraph	FFR$_s$
blocksworld	86.22	100.00	29.11	62.89	0.00	76.8
ex-blocksworld	51.56	24.22	31.33	42.67	30.74	51.56
drive	71.33	0.00	0.00	56.00	8.60	0
elevators	93.33	100.00	0.00	76.22	0.00	93
pitchcatch	53.56	0.00	0.00	23.11	0.00	0
random	100.00	0.00	0.00	64.67	5.11	73
schedule	51.33	0.00	0.00	54.22	0.92	0
tire	82.22	81.56	0.00	74.89	91.14	69
zeno	100.00	0.00	6.67	26.89	6.67	7

Figure 3: IPPC-06 successful trials results. Each row corresponds to a planning domain and each column corresponds to a planner. Each cell gives the percentage of successful trials in each domain.

	Average Number of Actions for Successful Trials in IPPC-06					
Domains	FFR$_a$	FOALP	sfDP	FPG	Paragraph	FFR$_s$
blocksworld	73.17	37.27	6.00	39.27	-	44.41
ex-blocksworld	19.17	7.47	1.87	3.73	1.87	21
drive	36.64	-	-	48.73	3.53	-
elevators	47.45	57.00	-	33.13	-	50.3
pitchcatch	588.41	-	-	20.33	-	-
random	20.15	-	-	103.40	0.07	12.7
schedule	105.49	-	-	198.20	0.33	-
tire	2.70	5.07	-	2.67	3.79	2.95
zeno	245.84	-	0.00	45.00	0.00	1

Figure 4: IPPC-06 average number of actions results. Each row corresponds to a planning domain and each column corresponds to a planner. Each cell gives the average number of actions used for successful trials in each domain.

	Average Time for Successful Trials in IPPC-06					
Domains	FFR$_a$	FOALP	sfDP	FPG	Paragraph	FFR$_s$
blocksworld	75.21	2.14	9.37	1.73	-	3.6
ex-blocksworld	1.17	0.71	0.69	0.18	0.03	0.36
drive	0.31	-	-	2.06	0.57	-
elevators	0.24	2.57	-	1.43	-	0.17
pitchcatch	1.19	-	-	0.87	-	-
random	2.12	-	-	10.04	0.00	0.32
schedule	0.72	-	-	9.92	0.01	-
tire	0.05	43.11	-	0.17	0.17	0.04
zeno	5.90	-	0.00	1.89	0.00	0

Figure 5: IPPC-06 average solution time results. Each row corresponds to a planning domain and each column corresponds to a planner. Each cell gives the average solution time on successful trials in each domain.

After solving the non-stationary deterministic problem, the first action in the plan is recorded and the process is repeated. After many repetitions the hindsight optimization approach suggests selecting the action that was recorded the most times.

There are a number of straightforward ways to find plans for the resulting non-stationary deterministic problems. In particular, one could encode the non-stationary problem as a standard stationary problem by having h copies of the operators, modified to include preconditions and effects that restrict the selection of an action to its corresponding time. It is also relatively straightforward to modify certain heuristic search planners and many SAT-based planning encodings to naturally facilitate for non-stationary actions.

It is likely that the hindsight optimization approach will be more robust in domains where probabilistic effects are critical, e.g., the exploding blocksworld domain in (Younes & Littman 2004a). The cost of hindsight optimization is computation time as it typically calls for drawing a number of random sequences and solving the corresponding planning problems for each state encountered. However, it is likely that incremental variants of this approach could be developed to significantly decrease the cost.

Policy Rollout. A simpler version of this idea is a technique called policy rollout (Bertsekas & Tsitsiklis 1996). Policy rollout computes a value estimate for each action in a state as follows. Each action is simulated from the current state and then FF-Replan is used to select actions until the goal is reached or until some horizon. This is done multiple times and the resulting plan lengths are averaged for each action, giving a value estimate. This effectively adds one step of look ahead on top of the FF-Replan policy, which can help avoid dead-ends among other types of sub-optimal behavior. We implemented a prototype of this idea and tested it on the exploding blocksworld of IPPC-06. The results showed that policy rollout does indeed improve over FF-Replan by a significant margin.

Acknowledgement

We thank the IPPC-04 and IPPC-06 hosts for their efforts on MDPSIM, domains, and the competition results.

References

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Blum, A., and Langford, J. 1999. Probabilistic planning in the graphplan framework. In *ECP*, 319–332.

Bonet, B., and Givan, R. 2006. International probablistic planning competition, url = `http://www.ldc.usb.ve/~bonet/ipc5/`.

Boutilier, C.; Brafman, R. I.; and Geib, C. 1998. Structured reachability analysis for Markov decision processes. In *Uncertainty in Artificial Intelligence 1998*, 24–32.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1-2):49–107.

Buffet, O., and Aberdeen, D. 2006. The factored policy gradient planner. In *International Probabilistic Planning Competition Booklet of ICAPS*.

Fern, A.; Yoon, S.; and Givan, R. 2006. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research* 25:85–118.

Geffner, H. 1998. Classical, probabilistic and contingent planning: Three models, one algorithm. In *AIPS'98 Workshop on Planning as Combinatorial Search*.

Gretton, C., and Thiebaux, S. 2004. Non markovian reward decision process planner, url=`http://users.rsise.anu.edu.au/~charlesg/nmrdpp/`.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:263–302.

Karabaev, E., and Skvortsova, O. 2005. A Heuristic Search Algorithm for Solving First-Order MDPs. In Bacchus, F., and Jaakkola, T., eds., *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI'2005)*, 292–299. Edinburgh, Scotland: AUAI Press. ISBN-0-9749039-1-4.

Mausam; Bertoli, P.; and Weld, D. S. 2007. A hybridized planner for stochastic domains. In *IJCAI*.

McDermott. 1998. Pddl-the planning domain definition language. In *The 1st International Planning Competition*.

Mercier, L., and Van HenTenRyck, P. 2007. Performance analysis of online anticipatory algorithms for large multistage stochastic programs. In *International Joint Conference on Artificial Intelligence*.

Meuleau, N., and Smith, D. 2003. Optimal limited contingency planning.

Puterman, M. 1994. *Markov Decision Processes*. Wiley, New York.

Sanner, S., and Boutilier, C. 2006. First order approximate linear programming. In *International Probabilistic Planning Competition Booklet of ICAPS*.

Wu, G.; Chong, E.; and Givan, R. 2002. Burst-level congestion control using hindsight optimization. *IEEE Transactions on Automatic Control*.

Yoon, S.; Fern, A.; and Givan, R. 2004. Learning reactive policies for probabilistic planning domains. In *International Probabilistic Planning Competition Booklet of ICAPS*.

Younes, H., and Littman, M. 2004a. International probablistic planning competition, url = `http://www.cs.rutgers.edu/~mlittman/topics/ipc04-pt`.

Younes, H. L. S., and Littman, M. L. 2004b. Ppddl1.0: An extension to pddl for expressing planning domains with probabilistic effects. In *Technical Report CMU-CS-04-162*.