# Incremental Plan Aggregation for Generating Policies in MDPs*

Florent Teichteil-Königsbuch
ONERA-DCSD
Toulouse Cedex 4, France
**Florent.Teichteil@onera.fr**

Ugur Kuter
University of Maryland
College Park, MD 20742, USA
**ukuter@cs.umd.edu**

Guillaume Infantes
ONERA-DCSD
Toulouse Cedex 4, France
**Guillaume.Infantes@onera.fr**

## ABSTRACT

Despite the recent advances in planning with MDPs, the problem of generating good policies is still hard. This paper describes a way to generate policies in MDPs by (1) determinizing the given MDP model into a classical planning problem; (2) building partial policies off-line by producing solution plans to the classical planning problem and incrementally aggregating them into a policy, and (3) using sequential Monte-Carlo (MC) simulations of the partial policies before execution, in order to assess the probability of replanning for a policy during execution. The objective of this approach is to quickly generate policies whose probability of replanning is low and below a given threshold.

We describe our planner RFF, which incorporates the above ideas. We present theorems showing the termination, soundness and completeness properties of RFF. RFF was the winner of the fully-observable probabilistic track in the 2008 International Planning Competition (IPC-08). In addition to our analyses of the IPC-08 results, we analyzed RFF's performance with different plan aggregation and determinization strategies, with varying amount of MC sampling, and with varying threshold values for probability of replanning. The results of these experiments revealed how they impact the time performance of RFF to generate solution policies and the quality of those solution policies (i.e., the average accumulated reward gathered from the execution of the policies).

## Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation

## General Terms

Algorithms

## Keywords

Planning (Single- and Multi-Agent)

*This research was supported in part by the French *Délégation Générale pour l'Armement* grant 07.60.031.00.470.75.01, US AFOSR grant FA95500610405, and US ARO grant W911NF0920072. The opinions in this paper are those of the authors and do not necessarily reflect those of the funders.

**Cite as:** Incremental Plan Aggregation for Generating Policies in MDPs, F. Teichteil-Königsbuch, U. Kuter, and G. Infantes, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1231-1238


## 1. MOTIVATIONS

In planning under uncertainty, the best known formalism is Markov Decision Processes (MDPs). Despite the general applicability and mathematical soundness of MDPs, the problem of planning with MDPs is still often computationally challenging, even in simple toy planning domains. This is due to the fact that most existing MDP planning algorithms try to optimize exactly large parts of the policy at once. This typically induces the requirement of enumerating the entire state and/or action spaces during planning.

There has been great strides in MDP planning over the years. Examples include techniques on state abstractions and aggregations [6, 2, 8], value function approximations [5, 16], heuristic and greedy search [1, 9, 12], policy-gradient computations [4], and generalizations of planning formalisms and algorithms developed originally for deterministic planning domains [13].

The most recent success in MDP planning is due to determinization and replanning techniques as evidenced by the works [17, 18]. The basis of these approaches is to take an MDP planning problem and "determinize" it into a classical planning problem and to generate a sequence of action (i.e., a plan) that represents a possible execution path in a potential solution policy. The MDP planner then executes this plan: if the goals are achieved then the algorithm reports success; otherwise, the algorithm re-generates (i.e., replan) a new plan from the failed state of the world.

Although this approach has been demonstrated to be practical in various MDP planning benchmarks, existing planners terminate just after having computed the first deterministic path to a goal state, which is typically likely to fail. Thus, the planner needs to generate and execute many plans to reach to the goal. This may result in the generation of the same plan over and over again to fail – although the current planners incorporate heuristic techniques that may avoid such situations. Furthermore, as a result of the above approach, the planner does not generate a policy; only an execution path from an initial state of an MDP to a goal state.

Most realistic scenarios, such as search and rescue [15], require that an MDP planner quickly generate offline policies that have some guarantees for not failing. If a failure still occurs during execution, then the planner could take a replanning approach as above in order to update its previous policy to circumvent the failure.

## 2. OVERVIEW

We identified four aspects of policy generation for MDP planning problems using a classical planning algorithm: *(1) how to determinize the MDP planning problem; (2) which plan (initial state and goal) to ask for (3) how to incrementally aggregate the plans generated by the classical planner into a solution policy for the MDP; (4) how to combine all of the above coherently.*

This paper presents our contributions on the four aspects of MDP

planning via plan aggregation. In particular, we describe:

- the Robust FF (RFF) planning algorithm that generates a policy for an MDP planning problem via well-informed calls to a classical planner on determinizations of the MDP. For generating classical plans, RFF uses the well-known FF planner [11].
- two MDP determinization mechanisms for RFF. These mechanisms are similar to the determinization techniques proposed by the works mentioned above [17, 18].
- a suite of policy-generation strategies for RFF selecting the starting states and the goals for generating plans via FF for an MDP planning problem. Both involves reasoning about the current policy and the likelihood about its successful execution towards achieving the original goals of the MDP.
- a suite of strategies for aggregating deterministic plans into a policy. These strategies specify different ways to combine the plans back into a policy for the original MDP, and to use Monte-Carlo simulations in order to compute, during planning time, the probability of replanning in case the execution of the policy fails. The planning agent then uses this probability value to decide whether (and how) to expand the policy further, or to execute it.
- theorems showing that RFF terminates in finite time, it is sound, and conditionally complete given the particular determinization strategy it uses. Finally, we show that if there are unsolvable states (i.e., have no path to a goal state), RFF's probability of replanning is complementary to its probability of success, i.e. reaching to a goal state, so that the probability of success of the aggregated policy is implicitly controlled by RFF.
- an extensive experimental evaluation of RFF. RFF was the winner of the 2008 International Planning Competition (IPC-08), the fully-observable probabilistic track. We report the IPC-08 results comparing RFF's performance with several other state-of-the-art planners. In additional experiments, we evaluated (1) different ways of selecting goal state(s) when the planning agent invokes FF; (2) different strategies for generating deterministic relaxations of an input MDP; and (3) the effects of varying thresholds on the probability of replanning.

## 3. DEFINITIONS AND NOTATION

Our definitions and notations are adapted from the usual MDP formulation of planning problems as in [7].

We consider MDPs of the form $M = (\mathcal{S}, \mathcal{A}, app, Pr, R)$. $\mathcal{S}$ and $\mathcal{A}$ are the finite sets of states and actions. $app(s)$ is the set of all actions applicable in $s$. For every $a \in app(s)$, $Pr(s, a, s')$ is the probability of the *state transition* $(s, a, s')$. For every $s \in \mathcal{S}$, $s' \in \mathcal{S}$ and $a \in app(s)$, $R(s, a, s')$ is the *reward* of applying $a$ to $s$ and generating $s'$.

An *MDP planning problem* is a triple $\mathcal{P} = (M, s_0, G, \rho)$, where $M = (S, A, app, Pr, R)$ is an MDP, $s_0 \in \mathcal{S}$ is the initial state[1], $G \subseteq \mathcal{S}$ is the set of goal states, and $0 < \rho \leq 1$ is a probability threshold as described below.

We say that a state $s$ in an MDP problem $\mathcal{P} = (M, s_0, G, \rho)$ is *unsolvable* in $M$ if there is no path from $s$ to a goal in $G$. We define $succ(s, a) = \{s' \mid Pr(s, a, s') > 0\}$ if $a \in app(s)$; $succ(s, a) = \emptyset$ otherwise. We let $succ(s, A)$ denote the set of successors of $s$ given the actions in $A$: $\bigcup_{a \in A} succ(s, a)$.

A *policy* is a partial function $\pi : S_\pi \rightarrow A$ for some set $S_\pi \subseteq S$. The *size* of $\pi$ is $|S_\pi|$. $V_\pi(s)$, the value of $s$ induced by the policy $\pi$, is defined as: $V_\pi(s) = E\left[\sum_{t=0}^{+\infty} \gamma^t r_t \mid s_{t_0} = s, \pi\right]$, where $\gamma \in [0, 1]$ is the discount factor and $r_t$ is the reward gathered at time step $t$ if we start from $s$ at time $t_0$. The optimal policy for an MDP problem is the one that maximizes $V_\pi(s)$ for each state $s$ in $\pi$.

$\pi$'s *execution structure* is a directed graph $\Sigma_\pi$ representing all possible executions of $\pi$. Formally, $\Sigma_\pi = (N_\pi, E_\pi)$, where $N_\pi = S_\pi \cup \{succ(s, app(s)) \mid s \in S_\pi\}$ and $E_\pi = \{(s, s') \mid s \in S_\pi, s' \in succ(s, app(s))\}$. A state $s$ is *expanded* in $\Sigma_\pi$, if $app(s) \neq \emptyset$ and $\exists (s, s') \in E_\pi : \forall s' \in succ(s, app(s))$. The state $s$ is *terminal* in $\Sigma_\pi$, if there is no $s'$ such that $(s, s') \in E_\pi$ and $s$ is not a goal state.

Let $\Pi_k$ be the set of all finite paths of length $k$ in $\Sigma_\pi$ such that each path $p \in \Pi_k$ starts at the initial state and ends in a terminal state $s$. Then, when $\pi$ is executed, the *probability of replanning* in $s$ is given by the following formula:

$$\omega(s_0, \pi, s) = \sum_{k=0}^{+\infty} \sum_{\langle (s_0,a_0)\ldots(s_k,a_k)\rangle \in \Pi_k} \prod_{i=0\ldots k} \Pr(s_i, \pi(s_i), s_{i+1}),$$

where $Pr(s, a, s')$ is the probability of the transition $(s, a, s')$.

Let $T = \{\text{all terminal states in } \Sigma_\pi\}$. Then, the *probability of replanning* when $\pi$ is executed in $s_0$ is: $\Omega(s_0, \pi) = \sum_{s \in T} \omega(s_0, \pi, s)$. The above probability value can be efficiently assessed by computing the successive probabilities $P_t(T \mid s, \pi)$ of reaching some terminal states in $T$ from $s$ in $t$ steps[2]:

$$P_0(T \mid s, \pi) = \begin{cases} 1, & \text{if } s \in T, \\ 0, & \text{otherwise} \end{cases}$$

$$P_t(T \mid s, \pi) = \sum_{s' \in succ(s, \pi(s))} P(s, \pi(s), s')\ P_{t-1}(T \mid s', \pi)$$

The replanning probability is: $\Omega(s_0, \pi) = \lim_{t \rightarrow +\infty} P_t(T \mid s_0, \pi)$

The rationale behind computing the probability of replanning of $\pi$ in terms of the probabilities of reaching to a terminal state of $\pi$, when $\pi$ is executed in its initial state is as follows. Suppose that $\pi$ has no terminal states – i.e. all states in which $\pi$ does not specify an action are goal states. Then the execution of $\pi$ is guaranteed to reach to a goal despite the uncertainty in the action outcomes. Now suppose $\pi$ has only one terminal state $s$. In this case, if the execution of $\pi$ reaches $s$, then this is a failure since $\pi$ does not specify what action to execute in $s$. Thus, the planning agent must replan for this state and hence the probability of replanning is equal to the probability of reaching to $s$ when $\pi$ is executed.

Note that the probability of replanning is generally *not* complementary of the probability of reaching a goal state, because the execution can lead as well to terminal states and absorbing expanded states from which it is impossible to reach the goals. In Section 5, we show the conditions under which the probability of replanning would be complementary to the probability of reaching a goal state.

Finally, a *solution* for an MDP planning problem $\mathcal{P} = (M, s_0, G, \rho)$ is a policy such that $\Omega(s_0, \pi) < \rho$.

## 4. ROBUST FF (RFF)

Figure 1 gives a high-level illustration of how RFF would proceed given an MDP planning problem. Given an MDP planning problem $(M, s_0, G, \rho)$, RFF first generates an initial plan $p$ from the initial state to some goal state by using FF on a *determinization* of the MDP. Then, RFF aggregates the original (i.e., MDP) versions of actions in $p$, and the states induced by the $succ$ function over those actions into the current partial policy $\pi$.

The execution of $p$ may fail with a high probability in the sense that it may lead to a terminal state of $\Sigma_\pi$. In each terminal state $s$

[1]By assuming a single initial state, we are not loosing any generality; one can define a special action that is only applicable in $s_0$ and generates only the states in a desired set of initial states with uniform probability.

[2]This is a special case of probabilistic dynamic programming with $\gamma = 1$ and where all rewards are 0 but 1 when arriving in a terminal state.

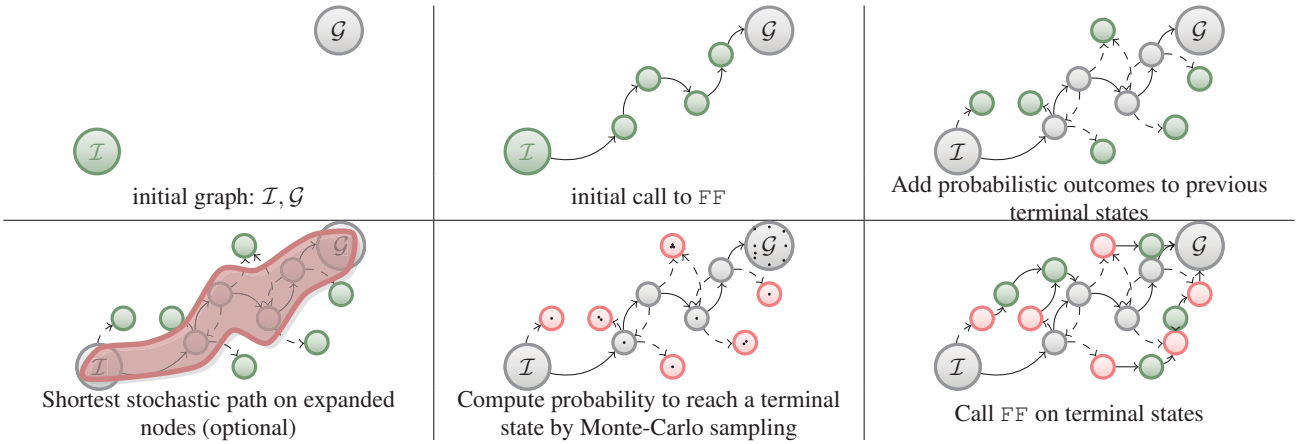


**Figure 1: An overview of planning in RFF.**

**Algorithm 1**: RFF$(M, s_0, G, \rho, N)$

1 $\mathcal{D} \leftarrow$ a deterministic relaxation of $M$
2 $T \leftarrow \{s_0\}$; $\pi \leftarrow \emptyset$; $\omega(s_0, \pi, s_0) \leftarrow 1$
3 **repeat**
4  $T' \leftarrow \emptyset$ // new terminal states
5  $X \leftarrow \emptyset$ // new expanded states
6  **for** $s \in T$ *such that* $\omega(s_0, \pi, s) > \rho$ **do**
7  pick $G_{\text{FF}} \subseteq G \cup S_\pi$
8  $p \leftarrow \text{FF}(\mathcal{D}, s, G_{\text{FF}})$
9  **if** $p \neq failure$ **then**
10  $s' \leftarrow s$; let $p = \langle \hat{a_1}, \ldots, \hat{a_k} \rangle$
11  **for** $1 \leqslant i \leqslant k$ **do**
12  $X \leftarrow X \cup \{s'\}$
13  $\pi(s') \leftarrow a_i$
14  $T' \leftarrow T' \cup succ(s', a_i) \setminus (S_\pi \cup G)$
15  $s' \leftarrow succ_{\mathcal{D}}(s', \hat{a_i})$
16  **else** $X \leftarrow X \cup \{s\}$
17  $T \leftarrow (T \setminus X) \cup T'$
18  $\{\omega(s_0, \pi, s) \mid s \in T)\} \leftarrow$ Fail_Prob$(s_0, \pi, T, N)$
19  $\Omega(s_0, \pi) = \sum_{s \in T} \omega(s_0, \pi, s)$
  // Next line is optional
20  Optimize the shortest stochastic path in $S_\pi$ by considering all states in $T$ as if they were unsolvable
21 **until** $\Omega(s_0, \pi) \leqslant \rho$ **or** $T = \emptyset$
22 **if** $\pi \neq \emptyset$ **then return** $\pi$
23 **else return** $failure$

reachable with a probability higher than $\rho$, compute a new path to a goal state from $s$ by using again the classical planner. We discuss how to choose the next goal state (or states) for FF below.

Optionally, we can compute the shortest stochastic path from the initial state to the goal states with a very high cost on terminal states (this optional step is costly but it incrementally expands the policy around the execution paths that reach from the initial state to a goal.

Finally, RFF selects a terminal state in the execution structure induced by the aggregated policy and repeat the above steps. This iterative process terminates as soon as the probability to reach a terminal state from the initial state with the current policy is less than the threshold $\rho$.

Algorithm 1 shows the pseudo-code for RFF. We are now ready to describe the details of the operations in the algorithm.

### *Generating Deterministic Relaxations of MDPs.*

In Lines 1 and 2, RFF generates a deterministic relaxation of the input MDP. The initial policy $\pi$ is the empty policy, and the only terminal state is the initial state (since it has not been expanded yet).

There are several known strategies for generating a deterministic relaxation of an MDP: see [17] for a review of some of them. In RFF, we primarily used a technique that generates a deterministic relaxation $\mathcal{D}$ of the input MDP as follows: *for each action $a$ in the MDP, $\mathcal{D}$ includes one and only action $\hat{a}$ such that $\hat{a}$ is applicable in exactly the same states as $a$ and $\hat{a}$ has the most probable outcome of $a$ as its only effect.* Note that the above determinization strategy is appropriate to our approach because RFF seeks to minimize path execution failure during execution due to least probable outcomes. We call this strategy the MOSTPROBABLEOUTCOME (MPO) relaxation of the MDP.

Another strategy we used in RFF for generating a deterministic relaxation of the MDP is the following: *for each action $a$ in the MDP, $\mathcal{D}$ includes one action $\hat{a}$ for each possible effect of $a$ such that $\hat{a}$ is applicable in exactly the same states as $a$.* We call this strategy as the ALLOUTCOMES (AO) relaxation of the MDP. Note that the AO relaxation of an MDP is useful in planning domains where the goal is not reachable using only the most probable outcomes.

In Line 15 of Algorithm 1, the function $succ_{\mathcal{D}}(s, \hat{a})$ specifies the successor states when the deterministic action $\hat{a}$ is applied in $s$.

### *Expanding the Execution Structure $\Sigma_\pi$.*

At each iteration, RFF attempts to expand the execution structure $\Sigma_\pi$ by planning one or more actions for each terminal state in $\Sigma_\pi$. For each terminal state $s$ in $\Sigma_\pi$ such that the probability to reach $s$ while executing the policy is higher than a given threshold (see Line 6), RFF first selects one or more goal states, $G_{\text{FF}}$ in Line 7, and then calls FF in order to generate a plan from the terminal state $s$ to any goal in $G_{\text{FF}}$. If FF returns a failure then this means that $s$ is unsolvable in the determinized domain. In that case, Lines 11–15 are not executed. Instead, Line 16 is executed in order to avoid to consider the current terminal state as unexpanded at the next iteration. This line is coherent with the probability of replanning and prevents the planner to replan from an unsolvable state.

**Choosing a terminal state in $\pi$ as an initial state for FF.** As mentioned above, RFF calls FF in the state $s$ that is a terminal state in the current policy $\pi$'s execution structure $\Sigma_\pi$ such that the probability of reaching to $s$ from the initial state $s_0$ of the input MDP planning problem, i.e., $\omega(s_0, \pi, s)$ in Line 6, is greater to the input threshold probability $\rho$.

In Lines 18 and 19 of Algorithm 1, RFF computes the probability that a policy $\pi$ will fail when executed from a state $s$. As we defined earlier, the probability to reach any terminal state from the initial

**Algorithm 2**: Fail_Prob$(s_0, \pi, T, N)$

```
1 for s ∈ T do ω(s_0, π, s) ← 0
2 for 1 ⩽ i ⩽ N do
3     s ← s_0
4     while s ∉ G or s ∉ T do
5         sample the next state s' given s and Σ_π
6         s ← s'
7     if s ∈ T then ω(s_0, π, s) ← ω(s_0, π, s) + 1/N
```

state $s_0$ is: $\lim_{t\to+\infty} P_t(T \mid s_0, \pi)$, where $P_t(T \mid s_0, \pi)$ is the probability to reach in $t$ steps a terminal state in $T$ starting from $s_0$.

Nevertheless, this computation may be quite costly. Thus, in RFF, we assess this probability computation by means of sequential Monte-Carlo (MC) simulations: We perform $N$ stochastic simulations in the MDP $M$ starting from the state $s$ given the policy $\pi$ and generate $N$ trajectories which all start from the initial state, and which end each as soon as a terminal state is reached or after a given depth. The probability to reach some terminal state is approximately the ratio between the number of trajectories which reach a terminal state and $N$.

The sequential MC simulation procedure Fail_Prob that we used in RFF is shown in Algorithm 2. Fail_Prob computes the probability to reach each terminal state. In Line 5, an outcome of a state $s$ by applying the current policy in $s$, is generated by sampling the outcomes of $s$ in the execution structure $\Sigma_\pi$. Each time a terminal state is reached, the routine exits the loop in Lines 4–6, and the weight of the reached terminal state increases by $1/N$ (Line 7). This procedure returns an array of size $|T|$ containing the probability failures $\omega(s_0, \pi, s)$ for all $s$ in $T$.

The probability of replanning is then given by the sum of the probabilities $\omega(s_0, \pi, s)$ for all $s$ in the set $T$ of terminal states of the execution structure (Line 19).

**Choosing the goal states for FF.** There are several possibilities for defining the goals $G_{\text{FF}}$ that RFF passes to FF at each iteration. A straightforward possibility is always to call FF with the goals of the input MDP planning problem. We name this simple strategy as PG (short for PROBLEMGOALS) for "goals of the input problem."

Although this strategy's implementation is simple, it often results in computing several times the same subpaths. To avoid this drawback, we also developed two other strategies. In both of them, when RFF generates a deterministic problem for FF, the goals of this problem is a subset of states for which the policy is already computed. In this case, one expects FF to reach neighbor states for which the policy is available, instead of trying to reach perhaps far goal states. The two strategies differ in how the set $G_{\text{FF}}$ of goal states for FF is generated, as described below.

As our first strategy, we randomly choose $k$ states from the policy graph and give them as goals to FF. The second strategy is a selection of the best $k$ states from the policy graph where the states are ranked according to some criterion. In particular, we execute the optional Line 20 of Algorithm 1 to optimize a local shortest stochastic path problem restricted to the policy graph. In this optimization, we consider all terminal states as if they were unsolvable, in order to focus the locally optimized policy to the already expanded states. We refer to the first strategy as RG (short for RANDOMGOALS), and to the latter as BG (short for BESTGOALS).

### *Aggregating FF's solution with $\Sigma_\pi$.*

With the input $(s, G_{\text{FF}})$, suppose FF returns a plan of the form $\langle \hat{a_1}, \ldots, \hat{a_k} \rangle$, where each $\hat{a}_i, i = 1, \ldots, k$ is a deterministic action generated by the determinization strategy given the set $A$ of actions of the MDP (see Line 8). We developed two methods for aggregating the plan returned by FF into the current execution structure $\Sigma_\pi$.

In the first aggregation method, RFF successively applies the deterministic actions of the plan $\langle \hat{a_1}, \ldots, \hat{a_k} \rangle$ in the current state $s$, and generates the state trajectory $\langle s, s_1, s_2, \ldots, s_k \rangle$. For each state in the trajectory, RFF first expands the current state $s'$ (Line 12) with the current action $\hat{a}_i$ and sets the current policy $\pi(s') = a_i$ (Line 13). Note that RFF sets $\pi(s')$ with the original action $a_i$ such that $\hat{a}_i$ is a determinization of $a_i$. Then, it adds the probabilistic successors of the current state in the set of new terminal states if they are not in the policy (Line 14).

Our second way to aggregate plans with a policy is shown on line 20: RFF optionally optimizes the stochastic length of paths to the goal states from the initial state. Here, RFF performs dynamic programming [14] over $S_\pi$ by assuming the following. For each state transition $(s, a, s')$ in $\Sigma_\pi$, if $s'$ is a goal state then the cost of $a$ is 0; if $s'$ is a terminal state then the cost of $a$ is $1/(1-\gamma)$; otherwise, the cost of $a$ is 1.

The cost of an action that produce a terminal state corresponds to the accumulated cost of an infinite trajectory in the MDP that never reaches a goal state. This allows us to consider terminal states as unsolvable states, so that the optimized policy will focus on the already expanded states. By selecting the action costs as above, RFF optimizes $\pi$ in a way that increases the probability of success while decreasing that of replanning during execution.

When RFF uses the stochastic optimization[3] in order to do plan aggregation, it does not update the current partial policy with the deterministic actions in the plan returned by FF. In other words, in this case, RFF uses FF as a heuristic that gives new interesting states to explore, not as a planner that gives a course of action to do. Thus, our implementation of the planner omits the Line 13.

Finally, RFF using the aggregation methods above does not necessarily optimize the *value function* of the input MDP [14]. However, it is possible that RFF still will produce sub-optimal solutions by computing *optimal* deterministic paths in the deterministic relaxation of the MDP if we use Metric-FF [10] instead of FF in Line 8 of the Algorithm 1. In that case, the average accumulated reward would not be optimal in many cases, but its value may be closer to the optimal than the value of policies generated using FF.

### *Termination.*

In Lines 3– 21, RFF computes successive paths from each terminal state reachable with a probability higher than the threshold $\rho$ (Line 6).[4] RFF terminates if either of the following conditions are satisfied: the global probability to reach some terminal state is less than the $\rho$, or there are no new terminal state to be added to the execution structure $\Sigma_\pi$ and all terminal state in $\Sigma_\pi$ are expanded already – i.e., $T$ is the empty set (see Line 21). If RFF cannot expand the initial state, i.e., FF cannot generate a plan given $s_0$ in the first iteration of RFF, then we have $T = \emptyset$ immediately and the planner returns $failure$ at Line 23.

## 5. THEORETICAL ANALYSIS

Recall that RFF is not an optimization algorithm and that it tries to find a solution to the following problem: *« Does it exist a policy whose probability of reaching to a goal state is non-zero, and whose*

[3] As the Bellman optimization can be quite expensive, when we do it, we use it both for selecting the BestGoals (BG) to give to FF and updating the policy. When we do not use the BG selection, then we implicitly use the first option for plan aggregation.

[4] Note that reactive approaches such as [17] terminate just after they generate an execution path; thus, they are a special case of RFF when $\rho = 1$.

*probability of replanning when executed is less than $\rho$? »*

The following lemma establishes that RFF terminates, i.e. it always returns a policy in finite time for any value of $\rho$.

LEMMA 1. *For every MDP planning problem $\mathcal{P} = (M, s_0, G, \rho)$, RFF terminates in finite time.*

PROOF. At each iteration, RFF always expands some terminal states in the execution structure, because the algorithm continues while the probability to reach some terminal states is greater than $\rho$. Because expanded terminal states are no more terminal by definition, RFF expands new states at each iteration. Yet, the number of states, and all the more of terminal states, is finite, so that the number of iterations is finite as well. □

THEOREM 1 (SOUNDNESS OF $\text{RFF}_{\text{MPO}}$). *For every MDP planning problem $\mathcal{P} = (M, s_0, G, \rho)$, every solution that $\text{RFF}_{\text{MPO}}$ finds is correct.*

PROOF. Let $\hat{\pi}$ be a solution found by $\text{RFF}_{\text{MPO}}$. Since FF is sound, every path starting in any state in $\hat{\pi}$ reaches a goal in the determinized domain, and a fortiori with a non-zero probability in the probabilistic original one. Therefore, $\hat{\pi}$ contains as least one path to a goal state from the initial state with a non-zero probability (in fact, as many such paths as successful calls to FF). Moreover, Monte-Carlo sampling is sound if the number of samples is large enough. In this case, the probability that execution of $\hat{\pi}$ leads to a terminal state is less than $\rho$. □

THEOREM 2 (COMPLETENESS OF $\text{RFF}_{\text{AO}}$). *For every MDP planning problem $\mathcal{P} = (M, s_0, G, \rho)$, if a solution exists, it is found by $\text{RFF}_{\text{AO}}$, otherwise $\text{RFF}_{\text{AO}}$ returns $failure$.*

PROOF. First case: a solution exists, i.e. it exists a non-zero probabilistic path from the initial state to goal states. In this case, it exists a path from the initial state to goal states in the deterministic domain determinized with the AO strategy. As FF is complete, it will find such a path in the determinized domain. Therefore, RFF will include this path in its execution structure, so that its solution policy contains at least one path to goal states from the initial state with a non-zero probability. As Monte-Carlo sampling is sound, probability that this policy leads to a terminal state is less than $\rho$.

Second case: no solution exists. In this case, no solution exists in the deterministic domain, so that FF which is complete, will return an empty plan from the initial state. Therefore, RFF immediately terminates by returning an empty policy. □

COROLLARY 1 (SOUNDNESS OF $\text{RFF}_{\text{AO}}$). *For every MDP planning problem $\mathcal{P} = (M, s_0, G, \rho)$, every solution that $\text{RFF}_{\text{AO}}$ finds is correct.*

PROOF. Follows from the proofs of Theorems 1 and 2. □

Note that $\text{RFF}_{\text{MPO}}$ is not complete in general. Indeed, in some planning problems, all paths from the initial state to any goal states could be associated to transitions whose outcome is never the most probable (among all outcomes that have the same starting state for each transition). Therefore, in this case, there is no path from the initial state to goal states in the determinized domain: FF always returns an empty plan so that RFF immediately returns an empty policy, even if there is solution to the problem.

Finally, we establish a condition between the probability of replanning and that of success, i.e. of reaching to a goal state. This proves that RFF implicitly controls the probability of success if there are no unsolvable states in the stochastic planning domain. In this case, the probability of replanning during execution, which is directly controlled at planning time, is the complementary of the probability of success, so that RFF returns a policy whose probability of reaching to a goal state is higher than $1 - \rho$.

THEOREM 3. *Let $\mathcal{P} = (M, s_0, G, \rho)$ be an MDP planning problem. If there are no unsolvable states in $M$, then the probability of success of any solution found by RFF is higher than $1 - \rho$.*

PROOF. The execution of a policy can lead to three different state classes: goal states, terminal states, and absorbing expanded states from which it is impossible to reach to goal states (defined as unsolvable). As a result, the execution of a policy found by RFF can lead to only two different state classes if there are no unsolvable states: goal states or terminal states. Therefore, the probability of reaching to a terminal state during execution is the complementary of the probability of reaching to a goal state. □

# 6. EXPERIMENTS

We have implemented RFF in C++ and performed our experiments on the planning domains from the "fully-observable probabilistic (FOP) planning track" of the IPC-08 [3].

There were 7 planning domains in total at the FOP track of IPC-08: namely, Blocks World, Exploding Blocks World, BoxWorld, 2-Tireworlds (2TW), Schedule, Search-and-Rescue, and SysAdmin. For each of the planning domain, there are 15 problem instances used at the FOP, where the problems differ not only with the size of the problems but also the costs and rewards of the underlying MDP formulation of the above planning domains. All of the experiments was done on a computer with a 2.4 GHz CPU and 2 GB memory running Linux kernel 2.6.27 (Ubuntu 8.10).

Due to space limitations, we give an extensive summary of our results below; the complete set of figures and discussions on the results, as well as the planning domains, planning problems, and our source code of RFF, are available at http://www.cs.umd.edu/users/ukuter/rff/.

**IPC-08 Experiments.** In the IPC-08, a solution policy was simulated 100 times in order to statistically assess the following: (1) the percentage of simulated trajectories that reach a goal state (Coverage), where $-10$ means that the instance was not solved, (2) the average number of time steps required to reach a goal state (Time), and (3) the average accumulated reward gathered among all simulated trajectories (Metric). Figure 2 shows the results.[5]

We used the following setup for RFF. The number $N$ of samples used by RFF's Monte-Carlo sampling procedure was 10. We used MPO relaxations of MDPs and both the PROBLEMGOALS (PG) and the BESTGOALS (BG) strategies for selecting goals in calling FF. When RFF calls FF if there are more than 100 states in the partial policy, then the number $|G_{\text{FF}}|$ of goals given to FF is 100 in the case of BG. We set the probability threshold to $\rho = 0.2$.

Overall, RFF solved many more planning problems across all planning domains from the FOP of IPC-08 than other competitors, with exception the Search-and-Rescue domain. We do not report any results in this domain since the planning problems do not have explicit goal states and RFF is not designed to solve such problems.

RFF did not perform well in the Exploding Blocks World problems. Results show that among the planning problems RFF could solve, it was among the fastest planners. Using PROBLEMGOALS goal-selection strategy, it was only able to solve 6 problem instances out of the 15 problems. The reason is that may become unsolvable during execution, and RFF may not detect this during planning time. When RFF calls FF with a deterministic relaxation of the MDP, FF generates a plan without any dead ends but when that plan is executed, one of the blocks that appear in the goal condition explodes: the planning problem becomes unsolvable.

[5]We are not showing any data with the Metric criterion since all planners performed similarly in terms of the average accumulated rewards.

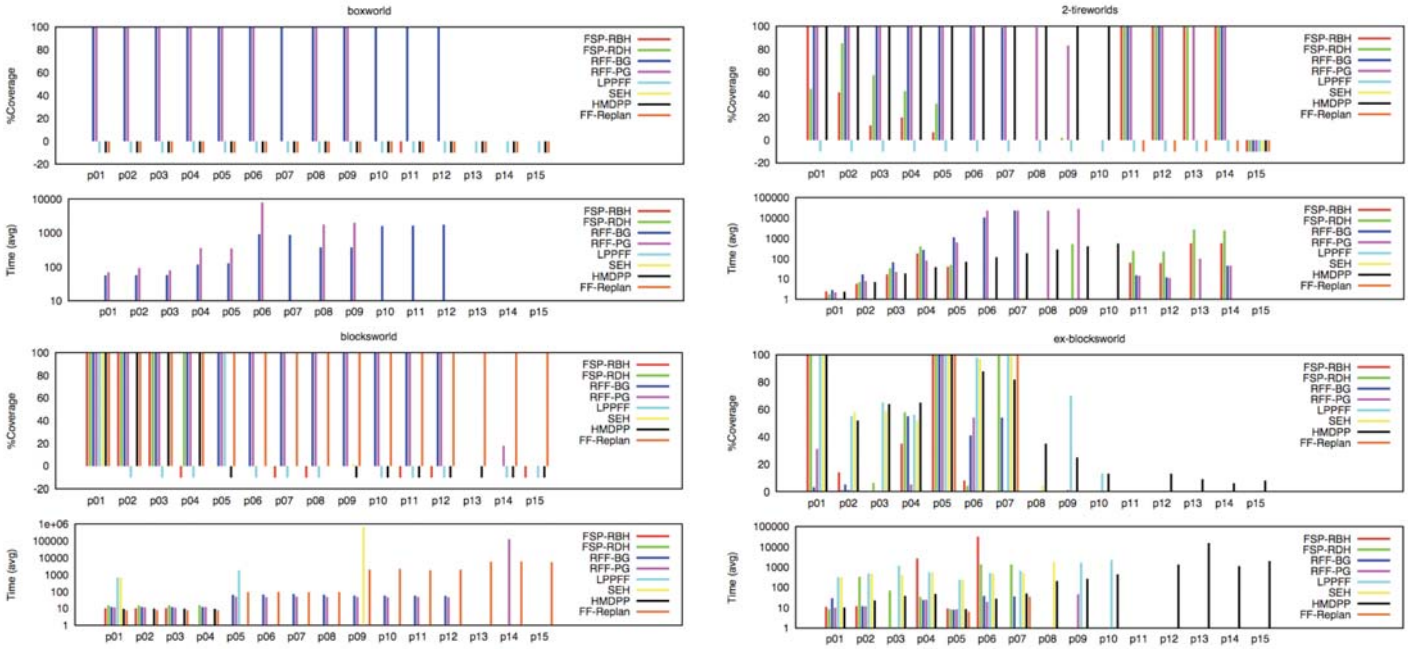


**Figure 2: Coverage and Time (described in text) results for the planning problems from the Uncertainty Track of the IPC-08. The parameter settings in RFF are given in text.**

RFF was the only planner that was able to solve problem instances from the BoxWorld domain. Note also that RFF was able to solve more (and larger) problem instances than FF-Replan, which did not participate in IPC-08 but was the winner of the 2004 probabilistic track. The reason is that RFF is able to reason, offline, about the terminal states of a policy and the probability of a policy will end up in such a state, when executed. This way it generates policies that avoid executions leading into terminal states, whereas FF-Replan do not reason about such states during planning time.

In terms of the average number of steps required to reach a goal state in the benchmark problems, the results show that RFF was among the best but there are no clear conclusions to be drawn from the results shown in Figure 2.

**Varying the Probability Threshold $\rho$.** Table 1 shows the average solution lengths generated by RFF, the average number of calls to FF, and the average CPU times for RFF, as a function of RFF's $\rho$. In these experiments, we fixed the following parameters: The number $N$ of samples used by RFF's Monte-Carlo sampling procedure was 10. We use MPO relaxations of MDPs and the PROBLEMGOALS strategy for selecting goals in calling FF. The time limit for each experiment was 20 minutes.

The results showed that the number of times a policy generated by RFF failed when executing that policy increases with $\rho$. This is as expected since higher values of $\rho$ mean that RFF will generate policies that are more prone to replanning by the definition of $\rho$. Moreover, these results confirm our Theorem 3, because

- in the domains without unsolvable states, the probability of success is higher than $1 - \rho$, and it is even equal to 1 in most cases;
- in the Exploding BlocksWorld domain (only domain with unsolvable states), the probability of success is generally not higher than $1 - \rho$.

In 2TW, Exploding Blocks World, and Blocks World domains, it is interesting to note that all three $\rho$ values we tried RFF with lead to almost the same behavior (except from two instances of the 2TW domain) in terms of number of problems solved and time taken. This is surprising because the number of failures happened in the executions is very different, but RFF achieved the same performance with different $\rho$ values. The reason is that RFF does not return a policy when the probability of replanning for that policy is $\rho$; instead, it may return any policy whose probability of replanning is less than or equal to $\rho$. RFF can generate the same solutions for the different values of $\rho$ in our experiments. Thus, the results show similar behavior for different values of $\rho$ in Table 1.

In BoxWorld and Schedule, smaller values for $\rho$ made RFF fail in trying to solve large instances. The small differences in the performance of RFF with varying values of $\rho$ is due to the fact that the goals given to FF are the same as the goals from the original problem, so RFF does not take advantage having more off-line planning. (Recall that in off-line planning, we can choose to merge policies by choosing RANDOMGOALS or BESTGOALS strategies).

**Different Goal-Selection Strategies.** Table 2 shows the results in all IPC-08 domains mentioned above, comparing our three different goal-selection strategies in RFF: PROBLEMGOALS (PG), RANDOMGOALS (RG), and BESTGOALS (BG). We fixed the number of Monte-Carlo simulations per call for the sampling routine in RFF to 10. We used $\rho = 0.2$. When RFF calls FF, if there are more than 100 states in the partial policy, then the number $|G_{\text{FF}}|$ of goals given to FF is 100 in the cases of the BG and RG. Otherwise, it is the maximum number given the selection mechanisms in these strategies. Note that in the case of PG, $G_{\text{FF}} = G$. As above, we used the MOSTPROBABLEOUTCOME (MPO) relaxations of the MDPs. Finally, we used 60 minutes as our time limit.

In our experiments, the PG strategy was clearly not the best one, it solved less number of problems in almost every domain. The BG strategy generally produced solutions with better quality than the RG strategy, and it took more time using the former compared to the latter, this mainly because of the way plans are aggregated: recall that in the BG strategy, the action to apply in states are com-

**Table 1: RFF with varying probability threshold $\rho$. The parameter settings in RFF are given in text.**

| DOM | PB | % success | | | sol. length (avg.) | | | total time (avg.) | | | # calls to RFF | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\rho$:0.1 | $\rho$:0.4 | $\rho$:0.7 | $\rho$:0.1 | $\rho$:0.4 | $\rho$:0.7 | $\rho$ :0.1 | $\rho$:0.4 | $\rho$:0.7 | $\rho$:0.1 | $\rho$:0.4 | $\rho$:0.7 |
| 2-tireworlds | 01 | **100** | **100** | **100** | 6.14 | **5.97** | 6.23 | 0.034 | 0.028 | **0.025** | 1 | 3 | 6 |
| | 02 | **100** | **100** | **100** | 11.83 | 11.71 | **11.4** | 0.149 | 0.135 | **0.115** | 8 | 14 | 18 |
| | 03 | **100** | **100** | **100** | 19.43 | **19.12** | 19.49 | 0.863 | **0.849** | 1.482 | 17 | 52 | 72 |
| | 04 | **100** | **100** | **100** | 27.1 | 27.24 | **26.34** | **9.714** | 175.098 | 1149.11 | 44 | 106 | 176 |
| | 05 | **100** | — | — | **34.46** | — | — | **549.31** | — | — | 48 | — | — |
| | 05-10 | — | — | — | — | — | — | — | — | — | — | — | — |
| | 11 | **100** | **100** | **100** | **5.08** | 5.2 | 5.2 | 2.514 | **2.425** | 2.48 | 1 | 2 | 2 |
| | 12 | **100** | **100** | **100** | **3** | **3** | **3** | **1.639** | 1.665 | 1.726 | 1 | 1 | 1 |
| | 13 | **100** | **100** | **100** | **5.82** | 5.97 | 6.22 | **20.751** | 21.256 | 21.015 | 1 | 3 | 4 |
| | 14 | **100** | **100** | **100** | **2** | **2** | **2** | 10.981 | 9.395 | **9.049** | 1 | 1 | 1 |
| | 15 | — | — | — | — | — | — | — | — | — | — | — | — |
| blocksworld | 01 | **100** | **100** | **100** | **20.03** | 20.37 | 20.52 | 2.699 | 17.469 | **0.507** | 1 | 1 | 1 |
| | 02 | **100** | **100** | **100** | 21.56 | 21.56 | **20.47** | 0.032 | **0.015** | 0.018 | 1 | 1 | 1 |
| | 03 | **100** | **100** | **100** | **20.47** | **20.47** | **20.47** | 0.019 | **0.017** | **0.017** | 1 | 1 | 1 |
| | 04 | **100** | **100** | **100** | **20.47** | 20.96 | 21.02 | 0.02 | 0.021 | **0.017** | 1 | 1 | 2 |
| | 05 | **100** | **100** | **100** | 47.96 | 48.5 | **47.7** | **1.311** | 1.373 | 2.739 | 2 | 11 | 22 |
| | 06 | **100** | **100** | **100** | **47.62** | 47.91 | 48.11 | 1.331 | **1.056** | 2.417 | 2 | 11 | 22 |
| | 07 | **100** | **100** | **100** | **47.24** | 47.91 | 47.94 | 1.368 | **1.057** | 3.309 | 2 | 12 | 17 |
| | 08 | **100** | **100** | **100** | 47.8 | 47.73 | **47.19** | 1.308 | **1.081** | 3.375 | 1 | 12 | 21 |
| | 09 | **100** | **100** | **100** | 37.18 | **37.05** | 38.32 | **1.028** | 1.056 | 1.141 | 1 | 4 | 9 |
| | 10 | **100** | **100** | **100** | 37.83 | **37.26** | 37.9 | 0.993 | **0.986** | 1.148 | 1 | 4 | 10 |
| | 11 | **100** | **100** | **100** | 37.04 | 38.53 | **36** | 1.096 | **0.993** | 1.038 | 1 | 3 | 9 |
| | 12 | **100** | **100** | **100** | 38.01 | **37.48** | 38.11 | 1.07 | **1.05** | 1.055 | 1 | 3 | 7 |
| | 12-15 | — | — | — | — | — | — | — | — | — | — | — | — |
| boxworld | 01 | **100** | **100** | **100** | 29.07 | 28.99 | **28.94** | 17.386 | **8.066** | 9.738 | 9 | 40 | 49 |
| | 02 | **100** | **100** | **100** | 28.94 | 28.94 | **28.79** | 11.851 | **6.244** | 8.274 | 8 | 28 | 43 |
| | 03 | **100** | **100** | **100** | 29.06 | **28.83** | 29 | 18.107 | **6.132** | 7.969 | 10 | 33 | 44 |
| | 04 | **100** | **100** | **100** | 35.33 | 35.57 | **35.17** | 482.728 | 114.038 | **59.427** | 23 | 80 | 101 |
| | 05 | **100** | **100** | **100** | **35.11** | 35.8 | 35.46 | 318.709 | 57.352 | **48.936** | 17 | 70 | 88 |
| | 06-07 | — | — | — | — | — | — | — | — | — | — | — | — |
| | 08 | — | **100** | **100** | — | **57.49** | 58.63 | — | 550.75 | **334.789** | — | 107 | 222 |
| | 09 | — | **100** | **100** | — | 58.44 | **58** | — | 667.635 | **269.167** | — | 100 | 194 |
| | 10-15 | — | — | — | — | — | — | — | — | — | — | — | — |
| ex-blocksworld | 01 | **58** | 55 | 54 | **424.64** | 454.4 | 464.32 | 0.045 | 0.127 | **0.038** | 2 | 9 | 5 |
| | 02 | **25** | 18 | 18 | **753** | 822.16 | 822.16 | 0.157 | **0.092** | 0.269 | 4 | 7 | 12 |
| | 03 | 39 | **46** | 45 | 613.9 | **544.6** | 554.5 | 0.383 | 0.298 | **0.197** | 8 | 12 | 11 |
| | 04 | 48 | **60** | 59 | 526.72 | **408.4** | 418.26 | **0.122** | 0.15 | 0.255 | 5 | 14 | 22 |
| | 05 | **100** | **100** | **100** | **6** | **6** | **6** | 0.024 | **0.022** | **0.022** | 3 | 4 | 4 |
| | 06 | **98** | 97 | 97 | **32.56** | 42.54 | 42.54 | 0.226 | **0.134** | 0.16 | 6 | 15 | 19 |
| | 07 | 56 | **64** | 58 | 446.72 | **367.68** | 426.96 | 0.169 | **0.121** | 0.31 | 2 | 9 | 13 |
| | 08 | 10 | **14** | 10 | 902.4 | **863.36** | 902.4 | 1.121 | **0.712** | 0.766 | 10 | 21 | 20 |
| | 09 | 9 | **15** | 9 | 912.28 | **853.86** | 912.3 | 2.316 | **1.463** | 1.252 | 20 | 34 | 30 |
| | 10 | 0 | **2** | 1 | 1000 | **980.72** | 990.36 | 2.332 | **1.319** | 1.722 | 9 | 24 | 26 |
| | 11 | 5 | 8 | **9** | 951.6 | 922.56 | **912.88** | 3.745 | **1.199** | 2.294 | 32 | 23 | 37 |
| | 12 | 1 | **4** | 2 | 990.38 | **961.52** | 980.76 | 6.238 | 1.858 | **1.394** | 27 | 28 | 23 |
| | 13 | 9 | **10** | 8 | 914.72 | **905.84** | 924.62 | 15.212 | 5.947 | **3.5** | 23 | 30 | 40 |
| | 14 | 1 | **2** | 1 | 990.48 | **981.1** | 990.58 | 437.651 | 493.033 | **230.651** | 38 | 30 | 37 |
| | 15 | **9** | 8 | 4 | **913.76** | 923.18 | 961.52 | 26.053 | **19.327** | 21.228 | 39 | 51 | 71 |
| schedule | 01 | **100** | **100** | **100** | 34.08 | 33.03 | 33.03 | 0.201 | 0.017 | 0.017 | 1 | 1 | 1 |
| | 02 | **100** | **100** | **100** | 51.57 | 45.54 | **41.7** | 0.018 | **0.017** | **0.017** | 1 | 1 | 1 |
| | 03 | **100** | **100** | **100** | **98.16** | **98.16** | **98.16** | **0.017** | **0.017** | **0.018** | 1 | 1 | 1 |
| | 04 | **96** | 96 | 93 | **92.08** | 92.13 | 127.44 | **0.449** | 1 | 0.582 | 2 | 18 | 11 |
| | 05 | 87 | 85 | **89** | 213.66 | 241.3 | **209.62** | 1.097 | **0.427** | 0.963 | 11 | 8 | 17 |
| | 06 | 70 | 60 | **89** | 464.72 | 594.35 | **209.62** | 207.154 | 305.934 | **0.963** | 96 | 270 | 17 |
| | 07 | 58 | 64 | **89** | 579.8 | 568.7 | **209.62** | 215.833 | 389.826 | **0.963** | 82 | 193 | 17 |
| | 08-15 | — | — | — | — | — | — | — | — | — | — | — | — |
| s-SLP | 01 | **100** | **100** | **100** | **6.49** | 6.86 | 7.16 | 0.327 | **0.286** | 0.377 | 2 | 8 | 11 |
| | 02 | **100** | **100** | **100** | **8.56** | 9.01 | 8.62 | 4.037 | 5.651 | **3.986** | 3 | 13 | 18 |
| | 03 | **100** | **100** | **100** | 12.14 | 11.68 | **10.43** | 53.239 | 41.606 | **41.194** | 5 | 20 | 33 |
| | 04-15 | — | — | — | — | — | — | — | — | — | — | — | — |

puted using Bellman optimization, whereas in others, this is the last action returned by calls to FF. However, BG failed to solve as many domains as the RG strategy, due to the overhead in the calculation, causing more time overflows (except in the 2TW domain).

As a summary, using PG in RFF resulted in solving a whole new problem for every terminal state (or for every policy failure if we consider a purely repairing approach as in FF-replan), which is clearly not a good idea for large domains. On the other hand, the BG strategy provided good results, as it reuses previous policy and optimizes it, but its overhead can be significant for large domains.

**Different Ways of Determinizing MDPs.** Table 3 shows number of successful runs, averaged number of steps taken to reach the goal, and averaged time for solving the problem for MOSTPROBABLEOUTCOME (MPO) and ALLOUTCOMES (AO) relaxations of MDPs. We fixed the number of Monte-Carlo simulations per the call of the sampling routine in RFF to 10. We used $\rho = 0.2$. The goals $G_{\texttt{FF}}$ that FF was given each time it was called are selected the same as above. The time limit was 20 minutes.

RFF using AO calls FF to solve much larger problems; for example, in the boxworld domain, RFF was not able to solve any problem instance using AO, whereas it generated good results with

**Table 2: RFF using the goal-selection strategies** PROBLEMGOALS(PG)**,** RANDOM GOALS (RG)**, and** BEST GOALS (BG)**. The parameter settings in RFF are given in text.**

| DM | PB | % coverage | | | solution length (avg.) | | | total time (avg.) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | PG | RG | BG | PG | RG | BG | PG | RG | BG |
| 2-tireworlds | 01 | 100 | **100** | **100** | **6.03** | 7.61 | 7.61 | **0.025** | 0.099 | 0.098 |
| | 02 | **100** | 40 | **100** | **12.27** | 606.05 | 22 | **0.126** | 1.795 | 1.685 |
| | 03 | **100** | 21 | **100** | **18.93** | 794.66 | 32.32 | **0.704** | 11.993 | 3.992 |
| | 04 | **100** | 6 | **100** | **27.02** | 941.89 | 44.47 | **6.943** | 79.702 | 9.862 |
| | 05 | **100** | 1 | 92 | **34.62** | 990.53 | 132.46 | 149.433 | 220.661 | **23.375** |
| | 06 | 3 | — | **100** | 971.32 | — | **68.17** | 948.315 | — | **45.366** |
| | 07 | 1 | — | **100** | 990.63 | — | **79.48** | 1105.9 | — | **74.832** |
| | 08 | 1 | — | **100** | 990.65 | — | **91.44** | 1136.5 | — | **121.78** |
| | 09 | — | — | **100** | — | — | **101.81** | — | — | **201.885** |
| | 10 | — | — | **100** | — | — | **115.57** | — | — | **320.598** |
| | 11 | **100** | 79 | 79 | **4.96** | 214.74 | 214.74 | **2.309** | 2.67 | 5.589 |
| | 12 | **100** | **100** | **100** | **3** | **3** | **3** | 1.501 | 1.507 | **1.5** |
| | 13 | **100** | 55 | 56 | **6.34** | 454.05 | 444.06 | **19.493** | 23.905 | 34.166 |
| | 14 | **100** | **100** | **100** | **2** | **2** | **2** | 8.548 | 8.529 | **8.526** |
| | 15 | — | — | — | — | — | — | — | — | — |
| blocksworld | 01 | **100** | **100** | **100** | **20.44** | 21.85 | 21.85 | 0.162 | 0.03 | **0.028** |
| | 02 | **100** | **100** | **100** | **20.96** | 21.85 | 21.85 | **0.025** | 0.03 | 0.03 |
| | 03 | **100** | **100** | **100** | **20.96** | 21.85 | 21.85 | **0.017** | 0.03 | 0.03 |
| | 04 | **100** | **100** | **100** | **20.96** | 21.85 | 21.85 | **0.015** | 0.03 | 0.03 |
| | 05 | **100** | **100** | **100** | **46.96** | 64.8 | 64.18 | **1.043** | 1.537 | 1.546 |
| | 06 | **100** | **100** | **100** | **46.82** | 65.27 | 65.24 | **1.03** | 1.537 | 1.556 |
| | 07 | **100** | **100** | **100** | **48.35** | 64.93 | 65.42 | **1.088** | 1.55 | 1.566 |
| | 08 | **100** | **100** | **100** | **47.83** | 64.5 | 64.41 | **1.06** | 1.545 | 1.542 |
| | 09 | **100** | **100** | **100** | **36.96** | 41.39 | 40.97 | **0.947** | 1.551 | 1.546 |
| | 10 | **100** | **100** | **100** | **37.54** | 40.43 | 40.85 | **0.966** | 1.554 | 1.556 |
| | 11 | **100** | **100** | **100** | **36.83** | 40.01 | 40.69 | **0.963** | 1.549 | 1.548 |
| | 12 | **100** | **100** | **100** | **36.87** | 40.6 | 40.26 | **0.964** | 1.551 | 1.549 |
| | 13 | — | **100** | — | — | 170.25 | — | — | 108.975 | — |
| | 14 | — | **100** | — | — | 182.02 | — | — | 109.946 | — |
| | 15 | — | **100** | — | — | 180.83 | — | — | 110.4 | — |
| boxworld | 01 | **100** | **100** | **100** | **29.07** | 30.55 | 30.32 | 11.082 | 3.293 | **3.022** |
| | 02 | **100** | **100** | **100** | **28.72** | 30.55 | 30.53 | 12.187 | **2.766** | 3.184 |
| | 03 | **100** | **100** | **100** | **29.15** | 30.65 | 30.82 | 9.564 | 3.373 | **2.791** |
| | 04 | **100** | **100** | **100** | **35.02** | 36.75 | 36.09 | 99.244 | **14.415** | 15.521 |
| | 05 | **100** | **100** | **100** | **35.34** | 36.47 | 36.1 | 100.654 | **13.587** | 16.693 |
| | 06 | — | **100** | **100** | — | 74.06 | **63.55** | — | **90.983** | 188.426 |
| | 07 | — | **100** | **100** | — | 72.82 | **64.63** | — | **96.223** | 181.199 |
| | 08 | **100** | **100** | **100** | 57.02 | 59.82 | **56.62** | 639.113 | **44.642** | 58.707 |
| | 09 | **100** | **100** | **100** | 58.44 | 60.06 | **57.62** | 673.589 | **43.558** | 45.641 |
| | 10 | — | **100** | **100** | — | 84.16 | **73.75** | — | **150.637** | 318.923 |
| | 11 | — | **100** | **100** | — | 85.89 | **73.83** | — | **168.326** | 321.594 |
| | 12 | — | **100** | **100** | — | 84.3 | **73.19** | — | **157.767** | 293.149 |
| | 13 | — | **100** | — | — | **209.76** | — | — | **2555.03** | — |
| | 14 | — | **100** | — | — | **202.54** | — | — | **2024.53** | — |
| | 15 | — | **100** | — | — | **205.42** | — | — | **2122.47** | — |
| ex-blocksworld | 01 | 50 | **67** | **67** | 504 | **335.36** | **335.36** | **0.078** | 0.081 | 0.081 |
| | 02 | 16 | **26** | **26** | 841.92 | **743.12** | **743.12** | **0.208** | 0.341 | 0.341 |
| | 03 | 31 | 34 | **35** | 693.1 | 663.4 | **653.5** | **0.104** | 0.361 | 0.132 |
| | 04 | 54 | **67** | 66 | 467.56 | **339.38** | 349.24 | **0.108** | 0.69 | 0.384 |
| | 05 | **100** | **100** | **100** | **6** | **6** | **6** | **0.023** | 0.038 | 0.038 |
| | 06 | **97** | 95 | 95 | **42.52** | 62.12 | 62.28 | 0.225 | **1.603** | 1.616 |
| | 07 | **63** | 60 | 56 | **377.56** | 407.2 | 446.72 | 0.436 | **0.371** | 0.716 |
| | 08 | **19** | 10 | 11 | **814.56** | 902.4 | 892.64 | **1.27** | 2.794 | 26.424 |
| | 09 | **17** | 7 | 9 | **834.34** | 931.76 | 912.32 | **1.448** | 8.188 | 20.802 |
| | 10 | **1** | 0 | **1** | **990.36** | 1000 | **990.36** | **0.961** | 14.686 | 38.013 |
| | 11 | **12** | 4 | 4 | **883.84** | 961.28 | 961.28 | **2.192** | 16.872 | 34.311 |
| | 12 | **2** | 0 | 1 | **980.76** | 1000 | 990.38 | **2.762** | 22.495 | 63.424 |
| | 13 | 10 | **13** | **13** | 905.52 | **877.12** | 878.28 | **9.477** | 265.081 | 247.439 |
| | 14 | **3** | **3** | 2 | **971.54** | **971.54** | 981.18 | **266.475** | 316.221 | 349.997 |
| | 15 | **10** | 8 | 6 | **904.16** | 923.06 | 942.38 | **16.355** | 83.639 | 92.11 |
| schedule | 01 | 100 | **100** | **100** | **31.65** | 32.76 | 32.76 | 0.231 | **0.016** | **0.016** |
| | 02 | **100** | **100** | **100** | **46.71** | **46.71** | **46.71** | 0.035 | **0.017** | **0.017** |
| | 03 | **100** | **100** | **100** | **107.25** | 108.78 | 108.78 | **0.015** | 0.017 | 0.018 |
| | 04 | 97 | **98** | **98** | 82.58 | **75.49** | **75.49** | 0.266 | **0.094** | **0.094** |
| | 05 | 89 | **93** | 89 | 208.96 | **179.19** | 207.47 | 0.398 | **0.121** | 0.126 |
| | 06 | **67** | 14 | 13 | **556.71** | 890.47 | 913.16 | 241.181 | 3.594 | **3.073** |
| | 07 | **57** | 12 | 13 | **572.72** | 914.23 | 911.72 | 216.666 | **3.26** | 3.77 |
| | 08 | **68** | 20 | 19 | **525.87** | 857.32 | 857.12 | 669.763 | 6.842 | **6.153** |
| | 09 | — | **1** | **1** | — | 993.34 | **991.23** | — | 51.378 | **21.693** |
| | 10 | — | **15** | 2 | — | **908.1** | 994.61 | — | 297.108 | **21.506** |
| | 11-15 | — | — | — | — | — | — | — | — | — |

MPO. In all other domains, MPO relaxations generally solve more problems in less time, with a good solution quality. This is because some of the problems are probabilistic versions of deterministic problems, where the most probable outcome is the desired one, and other are failures. In such cases, our results suggest that MPO relaxations of MDPs are worked quite well for RFF.

# 7. CONCLUSIONS AND FUTURE WORK

We have described RFF, our planning algorithm that (1) determinizes the given MDP model into a classical planning problem; (2) generates partial policies off-line by producing solution plans to the classical planning problem and incrementally aggregating them into a policy, and (3) uses sequential Monte-Carlo (MC) simulations of the partial policies before execution, in order to assess the

**Table 3: RFF with different method for relaxations of MDPs. MPO: Most Probable Outcome; AO: All Outcomes. The parameter settings in RFF are given in text.**

| DM. | PB. | %coverage | | sol. length (avg.) | | total time (avg.) | |
|---|---|---|---|---|---|---|---|
| | | MPO | AO | MPO | AO | MPO | AO |
| 2-tireworlds | p01 | **71** | 53 | **294.84** | 471.06 | 0.03 | **0.009** |
| | p02 | **60** | 10 | **410.11** | 900.4 | 2.015 | **0.021** |
| | p03 | **33** | 1 | **677.96** | 990.06 | 43.598 | **0.054** |
| | p04 | **9** | 0 | **912.79** | 1000 | **70.067** | 0.101 |
| | p05 | **3** | 1 | **971.02** | 990.1 | 268.498 | **0.373** |
| | p06 | — | — | — | — | — | — |
| | p07 | **1** | 0 | **990.6** | 1000 | 349.217 | **1.046** |
| | p11 | **79** | 16 | **214.74** | 840.32 | 6.254 | **5.677** |
| | p12 | **100** | 23 | **3** | 770.46 | **1.517** | 11.851 |
| | p13 | **42** | 19 | **583.01** | 810.38 | 52.952 | **35.627** |
| | p14 | **100** | 20 | **2** | 800.4 | **8.479** | 12.919 |
| | p15 | — | — | — | — | — | — |
| blocksworld | p01 | **100** | **100** | 22.08 | **15.83** | 0.211 | **0.028** |
| | p02 | **100** | **100** | 21.3 | **15.28** | 0.039 | **0.028** |
| | p03 | **100** | **100** | 21.3 | **15.28** | 0.03 | **0.028** |
| | p04 | **100** | **100** | 21.3 | **15.28** | **0.027** | 0.029 |
| | p05 | **100** | 5 | **63.93** | 981.79 | **1.57** | 1.603 |
| | p06 | **100** | 2 | **63.2** | 995.12 | **1.588** | 1.777 |
| | p07 | **100** | 6 | **64.74** | 977.06 | **1.689** | 1.912 |
| | p08 | **100** | 3 | **63.34** | 983.35 | 1.86 | **1.6** |
| | p09 | **100** | 32 | **41.21** | 859.16 | **1.567** | 6.813 |
| | p10 | **100** | 5 | **42.04** | 984.47 | **1.559** | 7.577 |
| | p11 | **100** | 6 | **40.36** | 977.17 | **1.567** | 7.584 |
| | p12 | **100** | 4 | **40.6** | 984.11 | **1.573** | 7.644 |
| | p13 | **100** | — | **175.28** | — | **110.461** | 58.752 |
| | p14 | **100** | — | **176.52** | — | **108.93** | 65.155 |
| | p15 | **100** | — | **176.89** | — | **110.213** | 59.18 |
| boxworld | p01 | 100 | — | 30.47 | — | 2.914 | — |
| | p02 | 100 | — | 30.75 | — | 3.177 | — |
| | p03 | 100 | — | 30.58 | — | 2.803 | — |
| | p04 | 100 | — | 37.01 | — | 14.59 | — |
| | p05 | 100 | — | 36.79 | — | 14.084 | — |
| | p06 | 100 | — | 73.95 | — | 87.748 | — |
| | p07 | 100 | — | 73.56 | — | 97.182 | — |
| | p08 | 100 | — | 60.08 | — | 42.351 | — |
| | p09 | 100 | — | 60.58 | — | 45.909 | — |
| | p10 | 100 | — | 85.9 | — | 151.257 | — |
| | p11 | 100 | — | 86.74 | — | 169.494 | — |
| | p12 | 100 | — | 85.87 | — | 141.396 | — |
| | p13 | 23 | — | 200.304 | — | 1183.44 | — |
| | p14 | 29 | — | 202.276 | — | 1181.42 | — |
| | p15 | 25 | — | 200.68 | — | 1179.23 | — |
| ex-blocksworld | p01 | **56** | 43 | **444.48** | 573.44 | 0.07 | **0.148** |
| | p02 | 20 | **23** | 802.4 | **772.76** | 0.231 | **0.139** |
| | p03 | **42** | 30 | **584.2** | 706.56 | **0.127** | 0.224 |
| | p04 | **63** | 62 | **378.82** | 397.64 | **0.415** | 0.619 |
| | p05 | **100** | **100** | **6** | 9.62 | 0.037 | **0.028** |
| | p06 | **97** | 91 | **42.24** | 115.02 | **1.567** | 2.336 |
| | p07 | 53 | **95** | 476.36 | **105.74** | 0.642 | **0.604** |
| | p08 | **10** | 9 | **902.4** | 915.78 | **2.547** | 3.554 |
| | p09 | 13 | **15** | 873.32 | **854.98** | **9.139** | 10.028 |
| | p10 | 0 | **1** | 1000 | **990.34** | **8.642** | 9.202 |
| | p11 | **5** | **5** | **951.6** | 952.12 | 26.858 | **24.342** |
| | p12 | **3** | — | **971.14** | — | **24.875** | — |
| | p13 | **11** | — | **896.38** | — | **190.348** | — |
| | p14 | **2** | — | **981** | — | **299.167** | — |
| | p15 | **6** | — | **942.44** | — | **44.532** | — |
| schedule | p01 | **100** | **100** | **28.71** | 28.89 | 0.157 | **0.013** |
| | p02 | **100** | **100** | **35.88** | 49.32 | 0.018 | **0.013** |
| | p03 | **100** | **100** | **103.23** | 103.59 | **0.015** | **0.015** |
| | p04 | **97** | 93 | **91.16** | 126.84 | 0.112 | **0.093** |
| | p05 | **89** | 86 | **214.07** | 236.73 | 0.138 | **0.128** |
| | p06 | 16 | **18** | 904.78 | **872.79** | **3.337** | 14.359 |
| | p07 | **18** | 16 | **857.8** | 892.91 | **3.662** | 14.317 |
| | p08 | **50** | 20 | **627.2** | 864.03 | 148.11 | **34.385** |
| | p09 | **2** | — | **994.73** | — | **56.195** | — |
| | p10 | **2** | — | **988.76** | — | **63.899** | — |
| | p11-p15 | — | — | — | — | — | — |

probability of replanning for a policy. This way, RFF assesses the probability of replanning for a policy and generates policies whose such probability is below a given threshold $\rho$.

We have provided theorems showing the termination, soundness, and completeness properties of RFF. We also showed that policies returned by RFF reach to goal states with a probability higher than $1 - \rho$ if there are no unsolvable states in the MDP.

We have done an extensive experimental study with RFF. RFF outperformed the planning systems that participated to the fully-observable probabilistic track in the 2008 International Planning Competition. In several additional experiments, we analyzed RFF's performance with different strategies for selecting goals for FF, for generating deterministic relaxations of the input MDPs, and the effect of varying threshold value for probability of replanning. RFF was best using deterministic relaxations of MDPs via the most probable outcomes of probabilistic actions. Furthermore, RFF solved the benchmark planning problems very effectively when the algorithm called FF with the best previously-explored states from the policy as goals. This enabled RFF to generate near-optimal policies while favoring small modifications to the current policy instead of exploring bad execution paths.

Our primary objective is to generalize our planning framework to hybrid MDPs in which there are both discrete and continuous valued state variables. The factored MDP representations in RFF will be the basis of such generalizations. We are currently investigating which classical planners are suitable to incorporate in RFF.

# 8. REFERENCES


[1] B. Bonet and H. Geffner. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *ICAPS-03*, 2003.

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

[3] D. Bryce and O. Buffet. The uncertainty part of the 2008 international planning competition, 2008. http://ippc-2008.loria.fr/wiki/index.php/Main_Page.

[4] O. Buffet and D. Aberdeen. FF+FPG: Guiding a policy-gradient planner. In *Proceedings of ICAPS*, 2007.

[5] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

[6] R. Dearden and C. Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1-2):219–283, 1997.

[7] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, May 2004.

[8] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147((1-2)):163–233, 2003.

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

[10] J. Hoffmann. The metric-FF planning system: Translating ignoring delete lists to numerical state variables. *JAIR*, 20, 2003.

[11] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[12] E. Keyder and H. Geffner. The hmdp planner for planning with probabilities, 2008.

[13] I. Little and S. Thiébaux. Concurrent Probabilistic Planning in the Graphplan Framework. In *ICAPS*, 2006.

[14] M. Puterman. *Markov Decision Processes*. Wiley, 1994.

[15] F. Teichteil-Königsbuch and P. Fabiani. A multi-thread decisional architecture for real-time planning under uncertainty. In *3rd ICAPS'07 Workshop on Planning and Plan Execution for Real-World Systems*, 2007.

[16] M. Trick and S. Zin. Spline approximations to value functions: A linear programming approach. *Macroeconomic Dynamics*, 1:255–277, 1997.

[17] S. Yoon, A. Fern, and R. Givan. FF-Replan: A baseline for probabilistic planning. In *ICAPS-07*, 2007.

[18] S. Yoon, A. Fern, R. Givan, and S. Kambhampati. Probabilistic planning via determinization in hindsight. In *AAAI-08*, 2008.