From FOND to Probabilistic Planning: Guiding search for quality policies

Alberto Camacho[†], Christian Muise[*], Akshay Ganeshen[†], Sheila A. McIlraith[†]
[†]Department of Computer Science, University of Toronto
[*]Department of Computing and Information Systems, University of Melbourne
[†]{acamacho,akshay,sheila}@cs.toronto.edu, [*]{christian.muise}@unimelb.edu.au

Abstract

We address the class of probabilistic planning problems where the objective is to maximize the probability of reaching a prescribed goal (MAXPROB). State-of-the-art probabilistic planners, and in particular MAXPROB planners, offer few guarantees with respect to the quality or optimality of the solutions that they find. The complexity of MAXPROB problems makes it difficult to compute high quality solutions for big problems, and existing algorithms either do not scale well, or provide poor quality solutions. We exploit core similarities between probabilistic and fully observable non-deterministic (FOND) planning models to extend the state-of-the-art FOND planner, PRP, to be a sound and sometimes complete MAXPROB solver that is guaranteed to sidestep avoidable dead ends. We evaluate our planner, Prob-PRP, on a selection of benchmarks used in past probabilistic planning competitions. The results show that Prob-PRP outperforms previous state-of-the-art algorithms for solving MAXPROB, and computes substantially more robust policies, at times doing so orders of magnitude faster.

1 Introduction

Planning involves finding action strategies, also called *plans*, that lead an agent to a desired *goal condition*. In scenarios with exogenous events, or where the effects of an agent's actions cannot be accurately modeled, the execution of a plan may not be fully predictable. In the planning community, uncertainty in the outcome of actions is modeled as a variant of the classical planning formalism that incorporates non-deterministic actions. Two similar models can be distinguished: fully observable non-deterministic (FOND) planning, and probabilistic planning. The former assumes fair non-determinism on the potential effects of the actions, while the latter breaks this assumption and assigns a probability distribution over the action outcomes. We address the subclass of probabilistic problems called MAXPROB, whose objective is to find policies that maximize the probability of reaching a prescribed goal (Kolobov et al. 2011).

We reflect on *what makes for a good quality policy*. Whereas optimal solutions to MAXPROB problems maximize the probability of reaching a goal, solutions often have other properties that are also desirable. Further, in contrast to *online* solutions, computing *offline* solutions makes it possible to offer guarantees with respect to the quality of solutions.

The state of the art in MAXPROB planning is the online planner *Robust-FF* (RFF) (Teichteil-Königsbuch, Kuter, and Infantes 2010). RFF is sound and complete in the all-outcomes determinization of problems with no dead ends, but it offers no guarantees with respect to the quality and optimality of the solutions in the presence of dead ends.

We exploit the core similarities between probabilistic and FOND planning models to extend the state-of-the-art FOND planner, PRP, to be a sound *offline* MAXPROB solver, which we call Prob-PRP. Prob-PRP prunes the search space by identifying state-action pairs that lead to undesired states, making it possible to handle large and probabilistically complex MAXPROB problems. The dead end detection mechanism in Prob-PRP guarantees finding optimal solutions in domains where all of the dead ends are avoidable.

We compare Prob-PRP with RFF across a variety of benchmarks from the International Probabilistic Planning Competition (IPPC). The results show that the quality of the policies is better for Prob-PRP across the majority of problems; particularly in domains with avoidable dead ends. Further, the computation time required is at times orders of magnitude faster, despite being offline.

Illustrative Example: The River Problem Consider the *River* problem introduced in (Little and Thiébaux 2007). In this problem, the agent has two options to cross a river: (i) traverse a path of slippery rocks with a 25% chance of success, a 25% chance of slipping and falling into the river, and a 50% chance of reaching a small island. In the latter case, she can swim towards the other side of the river with an 80% chance of success and a 20% chance of drowning; (ii) swim from one side of the river to the other, with a 50% chance of success, and a 50% chance of falling in.

The optimal solution is for the agent to attempt to traverse the rocks, and if she falls and survives, to swim from the island. This policy causes the agent to reach the other side of the river with a 65% probability of success, whereas the greedy action of swimming has only a 50% chance of success. Online replanners (e.g., (Yoon, Fern, and Givan 2007)) may be attracted by the shortest path to the goal (i.e. swimming directly across): once a wrong choice is made, online replanning approaches have no recourse to undo the bad action. This motivates the need for effective offline planning for MAXPROB problems with dead ends.

2 Preliminaries

We adopt the notation of Mattmüller et al. (2010) and Muise, McIlraith, and Beck (2012) for non-deterministic SAS$^+$ planning problems, extending it to probabilistic planning with conditional effects.

A SAS$^+$ *probabilistic planning problem* is a tuple $\langle \mathcal{V}, s_0, s_\star, \mathcal{A} \rangle$, where $\mathcal{V}$ is a finite set of variables v, each with domain $\mathcal{D}_\mathcal{V}$. We denote $\mathcal{D}^+_\mathcal{V}$ to be the extended domain that includes the undefined status, $\perp$, of v. A *partial state* (or simply, a state) is an assignment to variables $s : \mathcal{V} \to \mathcal{D}^+_\mathcal{V}$. If $s(v) \neq \perp$, we say that v is *defined* for s. When every variable is defined for s we say that s is a *complete* state. The *initial state* s_0 is a complete state, and the *goal state* $s_\star$ is a partial state. A state s *entails* a state s', denoted $s \models s'$, when $s(v) = s'(v)$ for all v defined for s'. The state obtained from *applying* s' to s is the *updated* state $s \oplus s'$ that assigns $s'(v)$ when v is defined for s', and $s(v)$ otherwise.

The actions $a \in \mathcal{A}$ have the form $a = \langle Pre_a, \mathit{Eff}_a \rangle$, where Pre_a is a state describing the condition that a state s needs to entail in order for a to be applicable in s. $\mathit{Eff}_a = \langle (p_1; \mathit{Eff}^1_a), \ldots, (p_n; \mathit{Eff}^n_a) \rangle$ is a finite set of tuples $(p_i; \mathit{Eff}^i_a)$ where $\Sigma_i p_i = 1$. Each *effect* Eff^i_a is a set of the form $\{\langle \mathit{cond}_1, v_1, d_1 \rangle, \ldots, \langle \mathit{cond}_k, v_k, d_k \rangle\}$, where for each j the *condition* cond_j is a partial state, and $v_j \in \mathcal{V}$, $d_j \in \mathcal{D}_{\mathcal{V}_i}$. The result of applying the action a in the partial state $s \models Pre_a$ with effect Eff^i_a is the partial state $\mathit{Result}(s, \mathit{Eff}^i_a) = \{v = d \mid \langle \mathit{cond}, v, d \rangle \in \mathit{Eff}^i_a \text{ and } s \models \mathit{cond}\}$. Finally, the *progression* of s w.r.t. action a and effect Eff^i_a is the updated state $Prog(s, a, \mathit{Eff}^i_a) = s \oplus \mathit{Result}(s, \mathit{Eff}^i_a)$.

An action a is applicable in a partial state s when $s \models Pre_a$. With a probability p_i, the outcome of a is one of the effects Eff^i_a that leads to the updated state $s' = Prog(s, a, \mathit{Eff}^i_a)$. The term (s, a, s') is called a *transition*, and has associated *transition probability* $T(s, a, s') = p_i$.

A solution to a probabilistic planning problem is a policy that maps (partial) states into actions with the objective of reaching a state that entails $s_\star$. Given a policy π, we denote S_π the set of states in which π is defined. We say that π is *well defined* when $\pi(s)$ is applicable in s for all $s \in S_\pi$. A policy is *closed* when it is defined in all states, and otherwise it is said to be a *partial* policy. A well-defined policy defines sequences of state-action trajectories $s_0, a_0, s_1, a_1 \ldots$ where $\pi(s_k) = a_k$, and $s_{k+1} = Prog(s_k, a_k, \mathit{Eff})$ for some $\mathit{Eff} \in \mathit{Eff}_{a_k}$. A sequence $P = a_0, a_1, \ldots$ of such actions is called a *plan*. We associate $P(s_i)$ with the action a_i.

We address the class of probabilistic planning problems where the objective is to maximize the probability of reaching a state that entails $s_\star$ (MAXPROB). We refer to this as the *probability of success*. The probability of success ignores action costs, and focuses on goal-achievement. A policy is optimal when its probability of success is maximal.

2.1 Related Planning Problems

A SAS$^+$ *Fully Observable Non-Deterministic* (FOND) planning problem is likewise described as a tuple $\langle \mathcal{V}, s_0, s_\star, \mathcal{A} \rangle$. Nevertheless, in contrast to a probabilistic planning problem, the FOND model assumes fair non-determinism with respect to the effects of the actions. As such, action effects are assumed to be equally likely, so there are no probabilities, p_i associated with the effects of actions (Muise, McIlraith, and Beck 2012). A solution to a FOND problem is a policy, usually distinguished according to the criteria introduced in (Cimatti et al. 2003). *Weak solutions* are plans that reach the goal under some sequence of non-deterministic action outcomes, and *strong solutions* are policies that are guaranteed to reach the goal in a bounded number of transitions. Finally, *strong cyclic solutions* are policies that are guaranteed to eventually reach the goal, under an assumption of fairness – effectively that in the limit, each action will yield each of its non-deterministic outcomes infinitely often.

Markov Decision Processes (MDPs) are another probabilistic planning model. The traditional MDP model introduced in Puterman (1994) associates rewards to state transitions. MAXREWARD MDPs have either a finite horizon or apply a discount factor $\gamma < 1$, and the solutions attempt to maximize the expected reward. Stochastic Shortest Path (SSP) MDPs (Bertsekas and Tsitsiklis 1991) is a class of goal-oriented probabilistic planning problems. Their solutions attempt to minimize the expected plan length under the assumption that the goal state is reachable from any state. A MAXPROB problem can be translated into a goal-oriented MDP with infinite horizon and discount factor $\gamma = 1$ by fixing the rewards to 1 in the goal states, and zero elsewhere (cf. (Kolobov, Mausam, and Weld 2012))

2.2 State of the Art in MAXPROB Planning

The International Probabilistic Planning Competition (IPPC) tests the performance of probabilistic planners periodically. It is notable that none of the winners of past IPPCs was an offline planner. As Little and Thiébaux (2007) point out, one of the reasons why online planners have outperformed offline planners is that the latter simply cannot handle large instances of complex problems.

The first IPPCs (IPPC-04 and IPPC-06) were dedicated to solving MAXPROB problems. The benchmark domains used in these competitions were probabilistically simple (Little and Thiébaux 2007), and a planner without probabilistic reasoning entitled *FF-Replan* (Yoon, Fern, and Givan 2007) won the IPPC-04 edition, and outperformed the participants of the IPPC-06 with only minor changes.

The IPPC-08 adopted more complex benchmark domains and focused on solving MAXREWARD probabilistic problems. The winner of the IPPC-08 was, with minor modifications, the MAXPROB planner *Robust-FF* (RFF) (Teichteil-Königsbuch, Kuter, and Infantes 2010). The IPPC-2011 and IPPC-2014 focused on solving MAXREWARD MDPs rather than goal-oriented probabilistic planning problems. The winner of these competitions, PROST (Keller and Eyerich 2012), works well for reward-based problems, but its performance suffers somewhat in goal-oriented settings such as MAXPROB. To the best of our knowledge, RFF is the state of the art for solving MAXPROB problems.

RFF computes plans in the determinization of the problem that are used to extend an *envelope* gradually until the esti-

mated probability of failure during execution is lower than a threshold parameter ρ. A failure during execution is understood as either falling into a deadend state, or falling into a state s that is unhandled by the envelope (in which case, RFF replans from s). The first envelope computed by RFF is a deterministic plan returned by a call to FF (Hoffmann and Nebel 2001); for each reachable state s not considered in the current envelope, RFF computes a deterministic plan for s and keeps iterating until the estimated probability of failure for the policy in the envelope is lower than ρ.

RFF offers optional modes of operation that have the potential to improve its performance. The *best-goals* strategy computes the set of states handled by the envelope that are the most likely to be reached during execution. Then FF is used to search for plans that lead to the goal, or to states in the best-goals set. Another mode performs *policy optimization* during the search process. Using dynamic programming, the policy is updated in each state to minimize the expected cost, assuming the following: for each transition (s, a, s'), the cost of a is 0 when s' is a goal state; a fixed penalty $1/(1-\gamma)$ when s' is a terminal state; and 1 otherwise. Alternatively, RFF has a mode for solving MAXREWARD problems that performs policy optimization and uses the actual transition costs given in the specification of the problem[1].

The policy optimization mechanism is an effort to avoid falling into terminal states, but it prioritizes the search for plans that reach a goal state in the lowest number of state transitions possible (i.e. stochastic shortest paths (Bertsekas and Tsitsiklis 1991)) that are not necessarily the best quality MAXPROB solutions. As an example, suppose that RFF explores two different plans from state s: (i) the first plan π_1 has a short path to the goal, but may also fall into a dead end; (ii) the second plan π_2 has no dead ends, but all paths to the goal are very long. Then, for sufficiently long paths in π_2, the policy optimization process in s selects π_1 over π_2 even though the latter policy is of higher quality.

RFF is sound and complete in the all-outcomes determinization of problems with no dead ends, but it offers no guarantees with respect to the optimality of the solutions found in the most probable determinization – even in problems without dead ends.

3 Quality of Solutions

The quality of the solutions to MAXPROB planning problems has been traditionally measured according to the probability of success, i.e., the probability of reaching the goal eventually. In this section we discuss the importance of other properties that account for *good* solutions – namely, the size of the policies and the expected length of the plans. To this point, there is no uniform theory of utility elicitation for solutions to MAXPROB problems. This paper does not intend to build that theory, but rather present an algorithm that finds MAXPROB solutions that maintain a good balance between the policy size and the expected plan length.

[1]Personal correspondence with Florent Teichteil-Königsbuch.

3.1 Policy Size

The FOND community has pursued policies that are *small* in the number of state-action pairs, and *compact* in so far as the policies can exploit partial state representations to capture a family of states. An obvious advantage of small, compact policies is that they are easily integrated into simple systems. While the FOND model often assumes the existence of either strong or strong cyclic solutions, core methods for obtaining small compact FOND policies can be applied to find solutions to probabilistic planning problems.

3.2 Expected Plan Length

Policy search should take into account the state transition probabilities, so that the expected length of the resulting plans do not become unnecessarily large.

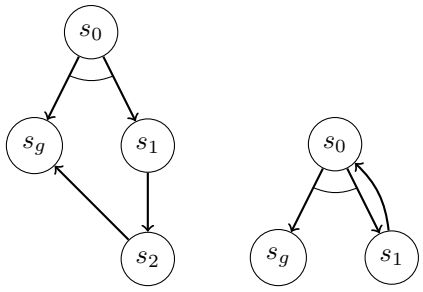
Relevant work has been done to find stochastic shortest path solutions to SSP MDPs (e.g. (Bonet and Geffner 2003; Trevizan and Veloso 2012)). Teichteil-Königsbuch (2012) proposed the class of *Stochastic Safest and Shortest Path* (S^3P) problems, that generalizes SSP MDPs and allows for unavoidable dead ends. Solutions to S^3P MDPs attempt to minimize the expected length of the plans and maximize the probability of success. Unfortunately, this model is not completely solved yet. Kolobov, Mausam, and Weld (2012) distinguishes the class of SSP MDPs with unavoidable dead ends, and presents a new family of heuristic search algorithms, FRET (Find, Revise, Eliminate Traps). Work in (Teichteil-Königsbuch, Vidal, and Infantes 2011) extends the classical planning heuristics h_{add} and h_{max} (Bonet and Geffner 2001) into heuristics for SSP MDPs with dead ends that, when plugged into graph-based MDP search algorithms, achieve competitive results compared to RFF. Unfortunately, none of these algorithms offer guarantees of optimality in problems with unavoidable dead ends.

3.3 Example

As an illustrative example, consider the controllers C_1 and C_2 in Figure 1, representing the state transitions of two policies, π_1 and π_2 respectively, that solve a MAXPROB problem $\mathcal{P}$. The nodes represent states and the arrows describe the state transitions, that may be non-deterministic (e.g., the action $\pi_1(s_0) = \pi_2(s_0)$ maps s_0 into s_1 or s_g with a certain probability distribution). Both solutions are strong cyclic and map s_0 into the goal s_g eventually, but only π_1 is strong and reaches the goal in a bounded number of state transitions. The expected length of the plans increases with the transition probability $T(s_0, \pi_2(s_0), s_1)$. For a sufficiently high value, π_1 may be preferred to π_2 despite the fact that it has a greater size. Furthermore, π_1 may always be preferred because the length of its plans are bounded.

4 Approach

Our contribution is to bring state-of-the-art FOND planning technology to probabilistic planning. We extend the FOND planner, PRP (Muise, McIlraith, and Beck 2012), to solve goal-oriented probabilistic planning problems, and we aim to maximize the probability of success (i.e. computing a solution to the MAXPROB problem).



Controller C_1 Controller C_2

Figure 1: Both solutions map the initial state s_0 into the goal state s_g eventually, but the size of the policies and the expected length of the plans is different.

4.1 Background

To the best of our knowledge, PRP is the state of the art in FOND planning. PRP searches for strong cyclic plans in a FOND problem, and produces a policy that maps partial states to actions. Key components of PRP include a mechanism to evade avoidable dead ends, and a compact representation of the policy in the form of state-action pairs (p, a) that tell the agent to perform the action a in a state s when s entails the condition p.

PRP runs a series of calls to Algorithm 1 until a strong cyclic solution is found, or the algorithm converges. In all cases, PRP returns the best quality policy found. The *Seen* and *Open* lists manage the states that belong to the incumbent policy. More precisely, the *Seen* list contains the states that have been processed already, whereas the *Open* list, initialized to the initial state s_0, contains the states that need to be processed. In each iteration, a state s from the *Open* list is processed. The procedure GENPLANPAIRS processes a non-goal state s for which a policy is undefined. This involves (i) computing a plan P for the all-outcomes determinization of $\langle \mathcal{V}, s, s_\star, \mathcal{A} \rangle$, such that P reaches the goal or a state handled by the policy , and (ii) augmenting the policy with the state-action pairs from the regression of P. The action $P(s)$ is added as a rule to the corresponding state-action pair in s. Processing a non-goal state s that is not in the *Seen* list involves adding all potential successors of s by $P(s)$ into the *Open* list, so that every reachable state of the policy is eventually processed.

Algorithm 1 is guaranteed to find a strong cyclic plan when the problem has no dead ends. When the problem has dead ends, GENPLANPAIRS may not find a plan for a certain state s. In that case, PROCESSDEADENDS computes a minimal partial state p such that $s \models p$ and every $s' \models p$ is a dead end. A set of forbidden state-action pairs is computed by regressing the dead ends through the computed plans. In the next calls to Algorithm 1, PRP resets the policy and the forbidden state-action pairs are excluded from search. The forbidden state-action pairs mechanism ensures completeness of the planner when the dead ends are avoidable.

Theorem 1 ((Muise, McIlraith, and Beck 2012)). *PRP returns a strong cyclic policy in problems with avoidable or no dead ends.*

Input: FOND planning task $\mathcal{P} = \langle \mathcal{V}, s_0, s_\star, \mathcal{A} \rangle$
Output: Partial policy π

```
1  InitPolicy();
2  while π changes do
3      Open ← {s_0} ;
4      Seen ← {} ;
5      while Open ≠ ∅ do
6          s = Open.pop();
7          if s ⊭ s_⋆ ∧ s ∉ Seen then
8              Seen.add(s);
9              if π(s) is undefined then
10                 GenPlanPairs(⟨V, s, s_⋆, A⟩, π);
11             if P(s) is defined then
12                 ⟨p, a⟩ = π(s);
13                 for e ∈ Eff_a do
14                     Open.add(Prog(s, a, e));
15     ProcessDeadends();
16 return π;
```

Algorithm 1: Generate Strong Cyclic Plan

The state-action avoidance, and the succinct states representations in PRP demonstrate an improved performance over existing state-of-the-art FOND planners (Muise, McIlraith, and Beck 2012). More recently, PRP was extended for use in FOND problems with conditional action effects (Muise, McIlraith, and Belle 2014) thus extending the class of domains that PRP can solve.

Similarly to RFF, Prob-PRP constructs a robust policy gradually by exploring weak plans. However, PRP does not store a complete representation of the states, but the portion of the states relevant to the policy computed via regression. The partial state representation enhances the possibilities of GENPLANPAIRS to generate a plan for a (partial) state already handled by the policy, providing substantial savings over approaches that use the complete state. Forbidden-state pairs are an effective mechanism to avoid dead ends. Such a mechanism is non-existent in RFF, leading to few alternatives to avoid dead ends already added to the envelope.

4.2 From FOND to MAXPROB

Given a MAXPROB problem $\mathcal{P}$, consider the FOND problem FOND($\mathcal{P}$) that results from ignoring the transition probabilities in $\mathcal{P}$. The strong cyclic solutions π in FOND($\mathcal{P}$) are certainly well-defined policies in $\mathcal{P}$ because the conditions $s \models Pre_{\pi(s)}$ hold equally in $\mathcal{P}$ and in FOND($\mathcal{P}$). Furthermore, all (fair) executions of π in FOND($\mathcal{P}$) eventually reach the goal. Therefore, all executions of π in $\mathcal{P}$ reach the goal with probability 1, independently of the probability distribution on the action effects, and π is also optimal in $\mathcal{P}$. Conversely, if π is an optimal policy in $\mathcal{P}$ that reaches the goal with probability 1, then every reachable state s has a weak plan that leads to the goal also in FOND($\mathcal{P}$), because π is well defined in the FOND problem. Therefore, π is a strong cyclic solution in FOND($\mathcal{P}$).

Lemma 1. *Let $\mathcal{P}$ be a probabilistic planning problem. If π is a well-defined policy in FOND($\mathcal{P}$), then π is a well-defined policy in $\mathcal{P}$.*

Lemma 2. *Let $\mathcal{P}$ be a MAXPROB planning problem. If π is a strong cyclic solution for FOND($\mathcal{P}$), then π is an optimal solution for $\mathcal{P}$ that reaches the goal with probability* 1.

Lemma 3. *A MAXPROB planning problem $\mathcal{P}$ has an optimal solution that reaches the goal with probability* 1 *iff FOND($\mathcal{P}$) has a strong cyclic solution.*

All strong cyclic solutions in FOND($\mathcal{P}$) are equally optimal to the MAXPROB problem $\mathcal{P}$. The similarities between the FOND and MAXPROB formalisms, as well as the equivalence of their solutions under certain conditions, suggest that existing approaches in FOND planning can be adapted to solve MAXPROB problems.

4.3 From PRP to Prob-PRP

The compact representation of states in PRP makes it possible to find small policies for FOND problems. We take advantage of the core similarities between FOND and MAXPROB formalisms, and define Prob-PRP as an extension of PRP that finds MAXPROB solutions to probabilistic planning problems and provides limited optimality guarantees.

Definition 1 (Plain Prob-PRP)**.** *The plain* Prob-PRP *algorithm on a MAXPROB problem $\mathcal{P}$ is a call to PRP on the mapped problem FOND($\mathcal{P}$).*

The soundness of PRP (Muise, McIlraith, and Beck 2012), and the result of Lemma 1 guarantee that the solutions found by PRP in FOND($\mathcal{P}$) are in fact well defined solutions for $\mathcal{P}$ (Lemma 4). PRP is guaranteed to find a strong cyclic solution to FOND($\mathcal{P}$) whenever one exists (Theorem 1), and FOND($\mathcal{P}$) has a strong cyclic solution whenever a proper plan exists for $\mathcal{P}$ (Theorem 3). Therefore, PRP is guaranteed to find a a strong cyclic solution to FOND($\mathcal{P}$) iff a plan with probability of success 1 exists for $\mathcal{P}$. The last condition occurs in domains with avoidable or no dead ends.

Lemma 4. *The plain Prob-PRP algorithm is sound.*

Theorem 2. *The plain Prob-PRP algorithm is guaranteed to find an optimal solution to MAXPROB problems with avoidable or no dead ends.*

Corollary 1. *Given a MAXPROB problem $\mathcal{P}$, if the plain Prob-PRP algorithm returns a solution where the probability of reaching a goal state is less than 1, then $\mathcal{P}$ has unavoidable dead ends.*

The MAXPROB solutions found by Prob-PRP are computed offline, with the advantage that no further computation is needed during execution. Computing offline solutions makes it possible to estimate the quality prior to execution – e.g. using Monte Carlo simulations as done in Prob-PRP– or even to compute it analytically. A consequence of Theorem 2 is that, if Prob-PRP returns a solution to $\mathcal{P}$ whose probability of success is lower than 1, then the problem $\mathcal{P}$ necessarily has unavoidable dead ends (Corollary 1).

4.4 Towards Better Quality Solutions

We propose two mechanisms that extend the plain version of Prob-PRP and potentially improve the quality of the solutions while maintaining the soundness of the solutions and the validity of Theorem 2.

Full Exploration in Last Iteration The forbidden state-action pairs mechanism in PRP may reduce the size of the search space dramatically, and improve the efficiency of the algorithm in searching for strong cyclic plans (Muise, McIlraith, and Beck 2012). This mechanism is also useful in solving MAXPROB problems with avoidable dead ends, but the search remains incomplete in domains with unavoidable dead ends. More precisely, when a state-action pair leads recognizably to a dead end, it is forbidden from successive searches. That direction in the search, however, may still lead to a goal state with non-zero probability.

Based on the previous observation, Prob-PRP performs a final iteration, where the best incumbent policy is used to initialize the policy on line 1 of Algorithm 1, and the problem is solved *with forbidden state-action pairs and dead end detection disabled*. The returned policy handles a superset of the states that it was able to previously, thus improving the quality of the solution. This final pass optimistically closes every *Open* state in a best effort manner.

Exploring Most Likely Plans Non-deterministic planning algorithms, like RFF or PRP, extend the policy under construction with plans that map unhandled states to any state that has been previously handled by the policy. This mechanism benefits creation of smaller policies, but the subsequent plans to the goal may be unnecessarily large, as the previous example illustrates. In order to reduce this effect, Prob-PRP skews the search towards short plans that have high likelihood. Previously, this method has been used to search plans in the determinization relaxation that minimise the risk of failing (c.f. (Jimenez, Coles, and Smith 2006)). Formally, the *likelihood* of a state-action plan $P = s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n$ is defined as the product:

$$L_P = \Pi_{i=0}^{n-1} T(s_i, a_i, s_{i+1})$$

The likelihood L_P measures how probable it is that the sequence of states $s_0, s_1, \ldots, s_n$ are reached when the stochastic plan $a_0, a_2, \ldots, a_{n-1}$ is executed. In certain domains, like *triangle-tireworld* or *climber*, the most probable outcome of the actions correspond to the desired effects that lead to the goal situation (Little and Thiébaux 2007). A priori, in these kinds of domains it seems reasonable to give preference to exploring the deterministic plans that maximize the likelihood. Loopy or unnecessarily large plans that belong to the policy have necessarily lower likelihood than alternative shorter plans. Subsequently, the expected length of the plans is potentially low.

Prob-PRP modifies the search process in GENPLAN-PAIRS performed by PRP, so that the plans that maximize the likelihood function are given preference to be explored. Since maximizing L_P is equivalent to maximizing the *log-likelihood* $l_P := log(L_P) = \Sigma_{i=0}^{n-1} log(T(s_i, a_i, s_{i+1}))$, and the latter expression offers a clear computational advantage, Prob-PRP maximizes the log-likelihood of the plans instead.

5 Evaluation

We compared the performance of Prob-PRP with RFF in a selection of benchmark problems from past IPPC competitions. The problems are described using standard PPDDL files with probabilistic outcomes and occasionally with conditional action effects (Fox and Long 2003). When required, the goal was modified to request MAXPROB solutions. We used the client-server architecture MDPSim – used in past editions of the IPPC – to simulate the execution of the solutions produced by each planner. All experiments were conducted on a Linux PC with an Intel Xeon W3550 CPU @3.07GHz, limiting the memory usage of each process to 2GB and the run time to 30 minutes.

The probabilistic planner Prob-PRP is implemented as an extension of the FOND planner PRP, and inherits the capability to handle problems with universally quantified formulas and conditional effects. We used the configuration of RFF that reported best results in (Teichteil-Königsbuch, Kuter, and Infantes 2010), namely, the most probable outcome determinization (so that the planner can scale to handle larger instances), and the *best goals* goal-selection strategy with policy optimization enabled. We fixed the probability of failure threshold ρ to 0.2, a number that results in good policy success rates without compromising the run times.

The results of the tests are shown in Table 1. The solution to each problem was run 100 times. For each planner, we report the percentage of successful runs, the average length of the successful plans, the average size of the policies, and the run times – that include only the computation time spent on generating the policy, and exclude the domain pre-processing time (usually negligible). For RFF, the average number of replans during successful runs needed by RFF is also reported, where the computation of the initial policy is counted as a replan.

5.1 General Analysis

Both Prob-PRP and RFF algorithms perform Monte Carlo simulations to estimate the probability of success of the policies found. We set the number of Monte Carlo samples to 1000, a number that does not compromise the execution time significantly and should be, in principle, high enough to provide sufficiently accurate estimations. We found that the cutoff mechanism in RFF is insufficient to compute reliable policies with sufficient guarantees. Indeed, the actual failure rate of the initial policy obtained by RFF in *boxworld-p02* and *boxworld-p12* is nearly 50% – clearly higher than ρ – suggesting that the probability of failure estimated by RFF is not accurate even with 1000 Monte Carlo samples.

The average number of RFF replans with $\rho = 0.2$ is lower than 2 in most of the problems. In practice, decreasing ρ results in a lower number of replans, but the run times increase significantly while the success rate of the solutions don't. The run times of Prob-PRP are comparable or lower than those of RFF with $\rho = 0.2$. More precisely, in many problems the run time needed by Prob-PRP to compute an offline solution is lower than the time needed by RFF to compute even the initial policy.

The size of the solutions found by Prob-PRP are comparable to those found by RFF, but it seems that Prob-PRP scales better in some domains. The compact state representation, and the forbidden-state action mechanisms in Prob-PRP prove efficient in the *triangle-tireworld* domain, leading to solutions that are orders of magnitude smaller, and computed orders of magnitude faster compared to RFF.

	RFF					Prob-PRP			
Problem	%	L	S	T	R	%	L	S	T
blocksworld-p01	**100**	23,2	18,0	0,02	1,00	**100**	**20,9**	**17**	**0,00**
blocksworld-p02	**100**	22,1	18,0	**0,02**	1,00	**100**	**20,8**	**17**	**0,02**
blocksworld-p03	**100**	22,6	18,0	0,02	1,00	**100**	**20,8**	**17**	**0,00**
blocksworld-p04	**100**	22,4	18,0	**0,02**	1,00	**100**	**20,9**	**17**	**0,02**
blocksworld-p05	**100**	64,9	60,6	0,72	1,01	**100**	**50,0**	**43**	**0,16**
blocksworld-p06	**100**	64,3	59,9	0,69	1,00	**100**	**50,6**	**43**	**0,16**
blocksworld-p07	**100**	64,3	60,6	0,69	1,01	**100**	**49,5**	**43**	**0,16**
blocksworld-p08	**100**	64,5	60,0	0,69	1,00	**100**	**50,0**	**43**	**0,16**
blocksworld-p09	**100**	**40,5**	**38,0**	0,67	1,00	**100**	68,4	61	**0,46**
blocksworld-p10	**100**	**41,2**	**39,1**	0,68	1,03	**100**	68,8	61	**0,46**
blocksworld-p11	**100**	**42,4**	**38,7**	0,66	1,02	**100**	68,6	61	**0,46**
blocksworld-p12	**100**	**41,7**	**38,4**	0,68	1,01	**100**	68,4	61	**0,46**
blocksworld-p13	0	∞	117	16,5	1,00	**100**	125	**107**	**1,38**
blocksworld-p14	0	∞	117	16,6	1,00	**100**	**125**	**107**	**1,40**
blocksworld-p15	0	∞	117	16,6	1,00	**100**	**125**	**107**	**1,38**
boxworld-p01	**100**	**29,4**	**49,3**	0,43	1,27	**100**	31,3	57	**0,06**
boxworld-p02	**100**	**29,3**	**49,2**	0,43	1,26	**100**	31,4	57	**0,06**
boxworld-p03	**100**	**29,0**	**47,9**	0,38	1,24	**100**	31,4	57	**0,06**
boxworld-p04	**100**	39,4	**75,7**	1,73	1,31	**100**	**38,6**	105	**0,24**
boxworld-p05	**100**	39,3	**80,9**	1,77	1,38	**100**	**38,5**	105	**0,24**
boxworld-p06	**100**	**64,9**	**166**	13,0	1,35	**100**	68,1	266	**2,34**
boxworld-p07	**100**	**64,5**	**160**	13,0	1,29	**100**	68,1	266	**2,32**
boxworld-p08	**100**	64,6	**125**	7,5	1,30	**100**	**62,4**	207	**1,82**
boxworld-p09	**100**	64,7	**132**	7,56	1,38	**100**	**62,5**	207	**1,84**
boxworld-p10	**100**	**74,3**	**199**	22,9	1,37	**100**	102	415	**17,2**
boxworld-p11	**100**	**73,3**	**183**	22,3	1,27	**100**	102	415	**17,9**
boxworld-p12	**100**	**74,1**	**199**	23,3	1,36	**100**	102	415	**18,0**
boxworld-p13	0	∞	344	35,6	1,94	**100**	178	906	130
boxworld-p14	0	∞	325	34,9	1,83	**100**	177	906	157
boxworld-p15	0	∞	347	35,2	1,96	**100**	177	906	160
ex-blocksworld-p01	60	8,0	20,8	0,05	1,06	**100**	**8,0**	**9**	**0,00**
ex-blocksworld-p02	28	12,0	36,8	0,11	1,16	**54**	14,0	**15**	**0,02**
ex-blocksworld-p03	38	10,0	31,8	0,10	1,14	**60**	**10,0**	**12**	0,12
ex-blocksworld-p04	52	14,0	49,5	0,09	1,13	**59**	32,9	**18**	**0,06**
ex-blocksworld-p05	**100**	**6,0**	11,6	**0,01**	1,09	**100**	**6,0**	**11**	0,02
ex-blocksworld-p06	90	12,6	62,2	0,10	1,35	**96**	20,7	**28**	0,34
ex-blocksworld-p07	60	12,0	36,2	0,20	1,12	**100**	**12,0**	**21**	**0,04**
ex-blocksworld-p08	7	24,0	68,9	0,64	1,20	**36**	30,0	**32**	**0,38**
ex-blocksworld-p09	13	25,2	95,7	1,07	1,23	–	–	–	t
ex-blocksworld-p10	2	36,0	76,8	0,97	1,24	**3**	116	105	14,3
ex-blocksworld-p11	**13**	**32,0**	92,7	**1,59**	1,31	**13**	93,4	**82**	7,42
ex-blocksworld-p12	1	38,0	96,6	2,15	1,21	**2**	91,5	**78**	6,28
ex-blocksworld-p13	**10**	**59,2**	**451**	**5,76**	1,45	–	–	–	t
ex-blocksworld-p14	**0**	**0**	**130**	**116**	1,24	–	–	–	t
ex-blocksworld-p15	**9**	**43,6**	**172**	**8,91**	1,28	–	–	–	t
schedule-p02	**100**	59,2	**5,00**	**0,01**	1,00	**100**	**51,0**	7	0,04
schedule-p03	**100**	100	**5,00**	**0,01**	1,00	**100**	**95,0**	7	0,12
schedule-p04	96	57,8	14,3	0,02	1,12	**100**	**46,9**	21	0,14
schedule-p05	89	116	14,5	0,03	1,15	**100**	**92,0**	16	0,18
schedule-p06	**45**	**364**	**141**	**1,42**	3,01	–	–	m	–
schedule-p07	**36**	**390**	**146**	**1,34**	3,11	–	–	m	–
schedule-p08	**34**	**354**	**146**	**3,94**	3,17	–	–	m	–
schedule-p09	**4**	**402**	**317**	**3,17**	4,31	–	–	m	–
triangle-tireworld-p01	**100**	**5,5**	22,7	0,02	1,07	**100**	**5,5**	**10**	**0,00**
triangle-tireworld-p02	**100**	13,2	80,7	0,17	1,32	**100**	**12,0**	**23**	**0,00**
triangle-tireworld-p03	**100**	21,8	135	0,66	1,13	**100**	**18,5**	**38**	**0,02**
triangle-tireworld-p04	**100**	29,6	248	1,76	1,20	**100**	**25,1**	**55**	**0,06**
triangle-tireworld-p05	**100**	37,6	348	3,76	1,12	**100**	**31,5**	**74**	**0,10**
triangle-tireworld-p06	**100**	45,6	490	7,98	1,14	**100**	**37,9**	**95**	**0,22**
triangle-tireworld-p07	**100**	53,4	714	17,4	1,21	**100**	**44,5**	**118**	**0,42**
triangle-tireworld-p08	**100**	61,5	958	36,5	1,19	**100**	**50,9**	**143**	**0,72**
triangle-tireworld-p09	**100**	69,5	1222	64,1	1,20	**100**	**57,5**	**170**	**1,40**
triangle-tireworld-p10	**100**	77,6	1595	111	1,21	**100**	**64,0**	**199**	**2,38**

Table 1: Performance of RFF and Prob-PRP. % indicates the percentage of successful executions; S indicates the size of the policy; T indicates run-time, in seconds and R indicates the number of replans in RFF. Dash (–) indicates that the experiments ran out of time (t) or memory (m).

5.2 Analysis of the Probability of Success

We split the analysis of the probability of success between problems without dead ends, problems with avoidable dead ends, and problems with non-avoidable dead ends.

Problems without Dead Ends Prob-PRP solves all the problems without dead ends in lower run times than RFF. The probabilistic reasoning overhead performed by Prob-PRP in the *blocksworld*, a simple domain without dead ends, does not translate into bigger run times than those in RFF.

We identified a looping behaviour in the solutions found by RFF to *blocksworld-p13*, *blocksworld-p14*, and *blocksworld-p15*, most likely originating in the policy optimization mechanism. The *problem-goals* strategy does not make use of the policy optimization mechanism, but fails to scale up to handle big problem instances (Teichteil-Königsbuch, Kuter, and Infantes 2010). We also found that the cutoff mechanism in RFF is insufficient to compute reliable policies with sufficient guarantees. Indeed, the actual success rate obtained in *boxworld-p04* and *boxworld-p12* is significantly lower than the threshold $1 - \rho$, suggesting that the probability of failure estimated by RFF is not accurate.

Problems with Avoidable Dead Ends The *triangle-tireworld*, introduced in (Little and Thiébaux 2007), requires a car to drive to a goal location via a number of intermediate locations. During each move, there is a possibility of getting a flat tire, and only some locations have spare tires. It is a challenging domain because both the size of the state space and the number of deadend states increase exponentially with the number of variables in the problem. We used the variant in which the car cannot carry a tire. The forbidden state-action mechanism makes it possible to identify the causes that lead to dead end states, and reduce the size of the state space significantly. Prob-PRP is able to optimally solve big instances of the *triangle-tireworld* domain orders of magnitude faster than RFF.

Problems with Unavoidable Dead Ends Neither RFF nor Prob-PRP offers guarantees on the optimality of the solutions in problems with unavoidable dead ends. As discussed earlier, RFF has a poor mechanism for *backtracking* once a bad plan that leads to a dead end has been explored. The forbidden state-action pairs mechanism and the full exploration in the last iteration performed by Prob-PRP makes it possible to improve the quality of the solutions relative to RFF in the *exploding-blocksworld* and *schedule* domains.

We again detected a looping behavior in some of the solutions given by RFF to the *exploding-blocksworld* problems. Prob-PRP exceeds the time limit (t) of 30 minutes in four instances. In *ex-blocksworld-p09*, this is not to the run-time itself, but to the time needed to decode and process the policy in a manageable format to be used by MDPSim, thus maintaining dead end avoidance. In the other instances, Prob-PRP does not converge within the time limits. Similarly, Prob-PRP exceeded the memory limits (m) in large instances of the *schedule* domain before convergence of the algorithm. These issues will be fixed by enabling an anytime mode in Prob-PRP, making it possible to output the best incumbent policy before the time or memory limits are reached.

5.3 Analysis of the Expected Length of the Plans

The policy optimization mechanism used by RFF prioritizes the search for plans that reach a goal state in the lowest number of state transitions possible. The results reported in Table 1 reflect that, on average, the plan length of the successful runs of the solutions computed by Prob-PRP are in the same order of magnitude than those of the solutions computed by RFF. In this section we evaluate the impact of the log-likelihood plan maximization strategy used by Prob-PRP towards the construction of policies with smaller expected plan length.

We compared the solutions obtained by Prob-PRP with a version that omits probabilities in the search of deterministic plans, and considers uniform action costs instead. We refer to this variation as Prob-PRP$_{uc}$. Note that this is not the same as assuming uniformly distributed outcome probabilities: a plan π has cost equal to its length regardless of the branching factor of the non-deterministic actions in π, whereas the branching factor influences the derived log-likelihood cost.

Table 2 shows the quality, run time, size, and expected plan length of the MAXPROB solutions to the different probabilistic planning problems obtained by Prob-PRP and Prob-PRP$_{uc}$. The overhead in Prob-PRP due to the probabilistic reasoning does not penalize the overall run time significantly, that remains in the same order of magnitude. Both algorithms find essentially the same solutions to the *triangle-tireworld* problems – where the optimal solutions w.r.t. success rate extend to stochastic shortest plan solutions quite straightforwardly. In the *blocksworld* domain, Prob-PRP obtains smaller policies with, in general, smaller expected plan length. In the *boxworld* domain, both algorithms find policies that are similar in size, but the expected plan length of the solutions found by Prob-PRP is considerably smaller in big problem instances. The *exploding-blocksworld* is a domain with many dead end states. In general, the size of the policies and the expected length of the plans found by each algorithm differ. In this domain, the most likely paths are not necessarily the more robust and the quality of the solutions do not seem to rely directly on a likelihood maximization criterion, nor on the election of a good heuristic to find deterministic plans. Rather, the quality of the solution appears to depend on a serendipitous election of the right sequence of actions during the search process.

The inconsistent quality of the plans obtained for the *exploding-blocksworld* domain suggests that the heuristic used by Prob-PRP in the search of deterministic weak plans is not informative in this domain. We obtained the best global results using a best-first search with the h_{FF} heuristic, although the combination of an $A^\star$ or breadth-first search with other heuristics is also competitive in certain problems. In particular, a best-first search with the additive heuristic h_{add}, which is usually informative, leads to similar results in most of the problems. Remarkably, in the *blocksworld* domain, the h_{add} heuristic performed better than the h_{FF} heuristic, generating smaller policies with smaller expected length. In the last three instances, this configuration finds solutions with 58 states and an expected plan length of 64 transitions, thus reducing the size and expected plan length to one half of the results reported in Table 2.

	Prob-PRP$_{uc}$				Prob-PRP			
problem	%	T	S	L	%	T	S	L
blocksworld-p01	**100**	0,02	21	24	**100**	**0,00**	**17**	**19**
blocksworld-p02	**100**	**0,00**	21	24	**100**	0,02	**17**	**19**
blocksworld-p03	**100**	**0,00**	21	24	**100**	0,02	**17**	**19**
blocksworld-p04	**100**	**0,00**	21	24	**100**	0,02	**17**	**19**
blocksworld-p05	**100**	**0,14**	**35**	**39**	**100**	0,18	43	47
blocksworld-p06	**100**	**0,14**	**35**	**39**	**100**	0,16	43	47
blocksworld-p07	**100**	**0,14**	**35**	**39**	**100**	0,16	43	47
blocksworld-p08	**100**	**0,14**	**35**	**39**	**100**	0,16	43	47
blocksworld-p09	**100**	**0,46**	71	75	**100**	0,48	**61**	**65**
blocksworld-p10	**100**	**0,44**	71	75	**100**	0,46	**61**	**65**
blocksworld-p11	**100**	**0,44**	71	75	**100**	0,46	**61**	**65**
blocksworld-p12	**100**	**0,46**	71	75	**100**	0,46	**61**	**65**
blocksworld-p13	**100**	**1,38**	110	119	**100**	1,40	**107**	**115**
blocksworld-p14	**100**	**1,38**	110	119	**100**	1,44	**107**	**115**
blocksworld-p15	**100**	**1,38**	110	119	**100**	1,40	**107**	**115**
boxworld-p01	**100**	0,14	**57**	156	**100**	**0,06**	**57**	**32**
boxworld-p02	**100**	0,16	**57**	156	**100**	**0,06**	**57**	**32**
boxworld-p03	**100**	0,14	**57**	156	**100**	**0,06**	**57**	**32**
boxworld-p04	**100**	0,36	**101**	86	**100**	**0,26**	105	**39**
boxworld-p05	**100**	0,34	**101**	86	**100**	0,28	105	**39**
boxworld-p06	**100**	6,56	269	363	**100**	**2,44**	**266**	**69**
boxworld-p07	**100**	6,60	269	363	**100**	**2,44**	**266**	**69**
boxworld-p08	**100**	**1,44**	**166**	170	**100**	1,92	207	**63**
boxworld-p09	**100**	**1,46**	**166**	170	**100**	1,92	207	**63**
boxworld-p10	**100**	**9,74**	**301**	328	**100**	18,2	415	**102**
boxworld-p11	**100**	**9,84**	**301**	328	**100**	18,2	415	**102**
boxworld-p12	**100**	**9,86**	**301**	328	**100**	18,0	415	**102**
boxworld-p13	**100**	576	949	1000+	**100**	**161**	**906**	**178**
boxworld-p14	**100**	507	949	1000+	**100**	**160**	**906**	**178**
boxworld-p15	**100**	586	949	1000+	**100**	**159**	**906**	**178**
ex-blocksworld-p01	**100**	**0,00**	**9**	**9**	**100**	**0,00**	**9**	**9**
ex-blocksworld-p02	**53,9**	0,04	**15**	**10**	**53,9**	**0,02**	**15**	**10**
ex-blocksworld-p03	59,3	0,16	**11**	9	**59,7**	**0,14**	12	**8**
ex-blocksworld-p04	60,1	**0,06**	**16**	**21**	**61**	**0,06**	18	**21**
ex-blocksworld-p05	**100**	**0,02**	**8**	23	**100**	0,02	11	**7**
ex-blocksworld-p06	96,3	0,68	**25**	**22**	**96,8**	**0,32**	28	**22**
ex-blocksworld-p07	**100**	1,76	**14**	31	**100**	**0,04**	21	**13**
ex-blocksworld-p08	**39,2**	1,00	**32**	23	36,6	**0,38**	**32**	**18**
ex-blocksworld-p09	**22,8**	**10,3**	**45**	31	10,2	58,7	77	**28**
ex-blocksworld-p10	**10,2**	**4,08**	**52**	28	4,6	14,0	105	**26**
ex-blocksworld-p11	9,6	33,8	89	29	**19,2**	**7,20**	**82**	**27**
ex-blocksworld-p12	–	–	m	–	**2,4**	**5,90**	**78**	**17**
schedule-p02	**100**	**0,04**	**7**	**48**	**100**	**0,04**	**7**	**48**
schedule-p03	**100**	**0,12**	**7**	**87**	**100**	**0,12**	**7**	**87**
schedule-p04	**100**	**0,08**	**16**	**43**	**100**	**0,14**	21	46
schedule-p05	**100**	**0,18**	**16**	96	**100**	0,20	16	**95**
triangle-tireworld-p01	**100**	**0,00**	**10**	**6**	**100**	**0,00**	**10**	**6**
triangle-tireworld-p02	**100**	**0,00**	**23**	**12**	**100**	**0,00**	**23**	**12**
triangle-tireworld-p03	**100**	**0,02**	**38**	**19**	**100**	**0,02**	**38**	**19**
triangle-tireworld-p04	**100**	**0,06**	**55**	**25**	**100**	**0,06**	**55**	**25**
triangle-tireworld-p05	**100**	**0,12**	**74**	**32**	**100**	**0,12**	**74**	**32**
triangle-tireworld-p06	**100**	**0,20**	**95**	**39**	**100**	**0,20**	**95**	**39**
triangle-tireworld-p07	**100**	**0,32**	**118**	**45**	**100**	0,36	**118**	**45**
triangle-tireworld-p08	**100**	**0,54**	**143**	**52**	**100**	0,62	**143**	**52**
triangle-tireworld-p09	**100**	**0,94**	**170**	**58**	**100**	1,14	**170**	**58**
triangle-tireworld-p10	**100**	**1,56**	**199**	**65**	**100**	1,84	**199**	**65**

Table 2: Solutions obtained by Prob-PRP using uniform action costs and log-prob action costs. % indicates the percentage of successful executions; T indicates run time, in seconds; S indicates the size of the policy; and L indicates the average length of the plans.

6 Extended Evaluation

In this section we introduce the benefits of a planner to be robust to small probability perturbations and different orderings used in the declaration of the actions.

6.1 Robustness to Probability Perturbations

The probabilistic planning model specifies the probability distribution of the outcomes of the actions, and is assumed to be known by the planner. In many real problems, however, these probabilities are unknown, or not known with complete accuracy. Ideally, the probability of success of a good solution is robust to small fluctuations in these probabilities. In practice, the search space explored by probabilistic algorithms exhibit *phase transitions* that change the structure of the solutions found. These phase transitions are most evident when the most probable outcome of the action changes.

In this section we evaluate the robustness of the solutions in the face of small deviations in the transition probabilities declared in the *triangle-tireworld* model. In this domain, the probability of a flat tire after moving the car is 0.5. We informed the planners with a slightly inaccurate probability, 0.45, breaking the uniform non-determinism of the action *move-car*. Strong solutions to the problem need to consider the faulty effect, that is no longer the most probable outcome. For that reason, we configured RFF to use the all-outcomes determinization.

	RFF		Prob-PRP	
problem	% sol	% sim	% sol	% sim
triangle-tireworld-p01	56.7	53.4	**100**	**100**
triangle-tireworld-p02	16.8	12.9	**100**	**100**
triangle-tireworld-p03	4.9	3.2	**100**	**100**
triangle-tireworld-p04	1.8	0.9	**100**	**100**
triangle-tireworld-p05	0.5	0.2	**100**	**100**
triangle-tireworld-p06	0.0	0.0	**100**	**100**
triangle-tireworld-p07	0.0	0.0	**100**	**100**
triangle-tireworld-p08	0.1	0.0	**100**	**100**
triangle-tireworld-p09	0.0	0.1	**100**	**100**
triangle-tireworld-p10	0.0	0.0	**100**	**100**

Table 3: Quality of the solutions in the *triangle-tireworld* domain when the probability of having a flat tire is perturbed from 0.5 to 0.45.

Table 3 shows the probability of success for solutions computed by RFF and Prob-PRP. The columns % *sol* and % *sim* in Table 3 indicate the probability of success when the environment model corresponds, respectively, to the model given to the planner, or to the original model.

The solutions found by RFF are not optimal anymore. Even when using the all-outcomes determinization, FF skews the search towards the shortest, but also optimistic plans that go through unsafe locations not equipped with a spare. Therefore, the envelope constructed by RFF does not converge to an optimal policy.

The success rate of the solutions found by RFF drops dramatically from the 100% achieved in the original domain description, and the performance in the simulation has an even lower percentage of success. On the other hand, since the *triangle-tireworld* domain has avoidable dead ends, Prob-PRP is guaranteed to find a strong cyclic independent of the transition probabilities of the model.

6.2 Robustness to Action Orderings

For evaluation purposes, we swapped the order in the declaration of the (equally probable) effects of the action *move-car*. In the new model, the *first* effect is optimistic and considers that the car's tire will not become flat, whereas the second effect is pessimistic and considers that the car's tire will become flat. With this ordering, the deterministic plans computed by FF in the most probable determinization of the problem are optimistic and not robust. As a consequence, RFF consistently fails to find robust policies, and the quality

of the solutions drops dramatically reaching a failure rate of 100% in the fourth instance. On the other hand, Prob-PRP is guaranteed to find optimal solutions to problems with avoidable dead ends regardless of the order used in the declaration of the actions and probability fluctuations.

7 Summary and Discussion

We introduced Prob-PRP, an algorithm that extends the state-of-the-art FOND planner PRP to compute solutions to MAXPROB problems. Prob-PRP is sound and complete for problems with avoidable dead ends. Probabilistic planning in the presence of avoidable and/or unavoidable dead ends is a challenging and important task (Kolobov, Mausam, and Weld 2012). We detailed a number of related approaches. Perhaps most similar to Prob-PRP's use of partial state policies and forbidden state-action pairs are the "basis functions" and "nogoods" computed by the GOTH planner and SixthSense algorithm (Kolobov and Weld 2010). Key distinctions include how Prob-PRP uses and updates the policy during the search for a plan, and how our dead ends are computed and used as forbidden state-action pairs.

MAXPROB solutions found by Prob-PRP are often optimal, and outperform the solutions found by RFF. We examined different properties that make for a good quality policy. Prob-PRP's solutions nicely balance their compactness and the expected length of the plans. Moreover, Prob-PRP demonstrates better scalability than RFF, and produces offline solutions. Computing offline solutions makes it possible to estimate the probability of success prior to execution, thus offering a better guarantee of the policy's quality than the solutions computed by online planners. Moreover, the guarantees on the optimality of the solutions in Prob-PRP makes it robust to small variations in the transition probabilities such as those found from an imprecise planning model.

We found that different search techniques for the deterministic subsolver – namely, a combination of breadth-first search, best-first search, $A^\star$, different heuristics, and uniform or probabilistic costs – offer similar results that are sometimes not of high quality. Whether the selection of an effective heuristic or search algorithm will significantly improve the results in these types of domains remains an open question. In future work we will explore these and other related issues associated with finding high-quality policies for such non-deterministic domains.

Establishing the correspondence of the computational core that is shared by FOND and probabilistic planning enables advances in either discipline to be exploited by the other. In this paper we demonstrated this by exploiting compact policy representations, relevance reasoning, and dead end avoidance developed within the FOND community and used these to advance the state of the art in probabilistic planning. Moving forward, we aim to inspire new methods for solving FOND problems using some of the insights from probabilistic planning, such as sample-based search.

Acknowledgements: The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and from Australian Research Council (ARC) discovery grant DP130102825.

References

Bertsekas, D. P., and Tsitsiklis, J. N. 1991. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research* 16(3):580–595.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33.

Bonet, B., and Geffner, H. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *ICAPS*, 12–21.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147:35–84.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:263–312.

Jimenez, S.; Coles, A.; and Smith, A. 2006. Planning in probabilistic domains using a deterministic numeric planner. *PlanSIG*.

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. *ICAPS*.

Kolobov, A., and Weld, D. S. 2010. SixthSense: Fast and reliable recognition of dead ends in MDPs. *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic search for generalized stochastic shortest path MDPs. *ICAPS* 130–137.

Kolobov, A.; Mausam; and Weld, D. S. 2012. A theory of goal-oriented MDPs with dead ends. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, 438–447.

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

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, 105–112.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Nondeterministic Planning by Exploiting State Relevance. In *ICAPS*, 172–180.

Muise, C.; McIlraith, S. A.; and Belle, V. 2014. Non-deterministic planning with conditional effects. In *ICAPS*, 370–374.

Puterman, M. 1994. *Markov Decision Processes: Discrete Dynamic Programming*. New York: Wiley.

Teichteil-Königsbuch, F.; Kuter, U.; and Infantes, G. 2010. Incremental plan aggregation for generating policies in MDPs. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1*, number 1, 1231–1238.

Teichteil-Königsbuch, F.; Vidal, V.; and Infantes, G. 2011. Extending Classical Planning Heuristics to Probabilistic Planning with Dead-Ends. *AAAI* 1017–1022.

Teichteil-Königsbuch, F. 2012. Stochastic Safest and Shortest Path Problems. *AAAI* 1825–1831.

Trevizan, F., and Veloso, M. 2012. Short-Sighted Stochastic Shortest Path Problems. *ICAPS*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *ICAPS*, 352–359.