Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

From FOND to Robust Probabilistic Planning: Computing Compact Policies that Bypass Avoidable Deadends

Alberto Camacho
Dept. of Computer Science
University of Toronto, Canada
acamacho@cs.toronto.edu

Christian Muise
MIT CSAIL
Massachusetts, USA
cjmuise@mit.edu

Sheila A. McIlraith
Dept. of Computer Science
University of Toronto, Canada
sheila@cs.toronto.edu

Abstract

We address the class of probabilistic planning problems where the objective is to maximize the probability of reaching a prescribed goal. The complexity of probabilistic planning problems makes it difficult to compute high quality solutions for large instances, and existing algorithms either do not scale, or do so at the expense of the solution quality. We leverage core similarities between probabilistic and fully observable non-deterministic (FOND) planning to construct a sound, offline probabilistic planner, ProbPRP, that exploits algorithmic advances from state-of-the-art FOND planner, PRP, to compute compact policies that are guaranteed to bypass avoidable deadends. We evaluate ProbPRP on a selection of benchmarks used in past probabilistic planning competitions. The results show that ProbPRP, in many cases, outperforms the state of the art, computing substantially more robust policies and at times doing so orders of magnitude faster.

1 Introduction

Many of the actions agents execute in the world exhibit some degree of nondeterminism in their outcomes and as such many real-world automated planning problems necessitate non-deterministic actions in their problem specification. Two common approaches to planning in the face of uncertainty in the outcome of actions are: *fully observable non-deterministic planning* (FOND) (e.g., (Alford et al. 2014; Mattmüller et al. 2010; Fu et al. 2011; Muise, McIlraith, and Beck 2012)), which assumes fair non-determinism in the effects of actions (Cimatti et al. 2003); and *probabilistic planning* which generally assigns a probability distribution over action outcomes (e.g., (Teichteil-Königsbuch 2012; Kolobov et al. 2011; Keller and Eyerich 2012)).

Recent advances in FOND planning have resulted in highly optimized planners that scale well, computing compact policies *offline* that bypass avoidable deadends (e.g., (Muise, McIlraith, and Beck 2012)). Our interest is in exploiting the computational core shared by FOND and probabilistic planning, as noted in (Hertle et al. 2014), to investigate whether state-of-the-art FOND planning techniques can be leveraged to produce an offline probabilistic planner that also bypasses avoidable deadends, scales well, and produces compact policies. We focus on what we will refer to as the task of *HighProb* whose objective is to find policies that attempt to maximize the probability of reaching a prescribed goal (e.g., (Teichteil-Königsbuch, Kuter, and Infantes 2010). We contrast this with MaxProb planners, which guarantee finding the maximum probability plan (e.g., (Kolobov et al. 2011)). We assume goals are *absorbing* or final-state, as in classical planning and as typically assumed for MaxProb.

To this end, we develop ProbPRP, an offline HighProb planner that exploits algorithmic advances from state-of-the-art FOND planner, PRP (Muise, McIlraith, and Beck 2012). We compare ProbPRP with the state of the art in HighProb planning, RFF (Teichteil-Königsbuch, Kuter, and Infantes 2010). The results show that ProbPRP, in many cases, outperforms the state of the art, computing substantially more robust policies at times doing so orders of magnitude faster.

2 Preliminaries

A *probabilistic* planning problem is a tuple $\mathcal{P} = \langle S, s_{\mathcal{I}}, \mathcal{A}, T, S_{\mathcal{G}} \rangle$ where S is a finite set of *states*, $s_{\mathcal{I}} \in S$ is the *initial state*, $S_{\mathcal{G}} \subseteq S$ is a set of *goal states*, and $\mathcal{A}$ is a finite set of actions. For each action $a \in \mathcal{A}$, and pair of states $s, s' \in S$, $T(s, a, s')$ is the probability that *transition* (s, a, s') occurs, i.e., that s' results from applying a in s.

In a SAS$^+$ representation of the problem, states are assignments to a finite set of variables $\mathcal{V}$, each $v \in \mathcal{V}$ with domain $\mathcal{D}_v$. We denote $\mathcal{D}_v^+$ the extended domain that includes the undefined status, $\perp$, of v. A *partial state* (or simply, a state) is an assignment to variables $s : v \in \mathcal{V} \to \mathcal{D}_v^+$. 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'. 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. $\mathit{Eff}_a = \langle \mathit{Eff}_a^1, \ldots, \mathit{Eff}_a^n \rangle$ is a finite set of *effects*. Each effect Eff_a^i is a set of the form $\{\langle cond_1, v_1, d_1 \rangle, \ldots, \langle cond_k, v_k, d_k \rangle\}$, where for each j the *condition* $cond_j$ is a partial state, and $v_j \in \mathcal{V}$, $d_j \in \mathcal{D}_{v_i}$. The result of applying the action a in the partial state $s \models Pre_a$ with effect Eff_a^i is the partial state $\mathit{Result}(s, \mathit{Eff}_a^i) = \{v = d \mid \langle cond, v, d \rangle \in \mathit{Eff}_a^i \text{ and } s \models cond\}$. Finally, the *progression* of s with respect to action a and effect Eff_a^i is the updated state $Prog(s, a, \mathit{Eff}^i_a) = s \oplus \mathit{Result}(s, \mathit{Eff}^i_a)$. Here, $s \oplus s'$ assigns $s'(v)$ when v is defined for s', and $s(v)$ otherwise.

Solutions to probabilistic planning problems are *policies*, or mappings $\pi : S \to \mathcal{A}$ from states into actions. In goal-oriented probabilistic planning models such as MaxProb (e.g. (Kolobov et al. 2011)), solutions are policies that lead the agent to a goal state with maximal probability. We say a policy π is *well-defined* when $\pi(s)$ is applicable in s for all reachable states by π. A well-defined policy defines sequences of state-action trajectories, or *plans*, $P = s_0, a_0, s_1, a_1 \ldots s_n$ where $\pi(s_k) = a_k$, and (s_k, a_k, s_{k+1}) is a transition. The *likelihood* of P is the product $L_P = \Pi_{i=0}^{n-1} T(s_i, a_i, s_{i+1})$

The model for probabilistic planning is related to *Fully Observable Non-Deterministic* (FOND) planning. A FOND problem is a tuple $\mathcal{P} = \langle S, s_\mathcal{I}, \mathcal{A}, T, S_\mathcal{G} \rangle$, where $S, s_\mathcal{I}, \mathcal{A}$, and $S_\mathcal{G}$ are defined as in the probabilistic planning model. However, for each action $a \in \mathcal{A}$ and state $s \in S$, the result of applying a in s is one of the states in the set $F(s, a) \subseteq S$ without specifying transition probabilities. Solutions to FOND planning problems are *policies*, similar to probabilistic problems. The so-called *strong-cyclic* solutions are those that lead the agent to a goal state in the limit (Cimatti et al. 2003).

3 Approach

We address the class of HighProb planning problems where the objective is to maximize the probability of reaching a goal state. We refer to this as the *probability of success*. Solutions with higher probability of success are preferred, and optimal solutions are those that maximize it.

3.1 PRP Preliminaries

To the best of our knowledge, PRP (Muise, McIlraith, and Beck 2012) is the state of the art in FOND planning. PRP searches for weak plans that are gradually extended into robust policies (see Algorithm 1). Key components of PRP include compact representation of the policy in the form of state-action pairs (p, a), and a mechanism to bypass avoidable deadends. The procedure GENPLANPAIRS processes a non-goal state s in the *Open* list for which a policy is undefined. This involves (i) computing a plan P for the all-outcomes determinization of the problem, such that P reaches the goal or a state handled by the policy, and (ii) augmenting the policy with the (partial) state-action pairs obtained from the regression of P; When s is a deadend, PROCESSDEADENDS computes a minimal partial state p such that $s \models p$ and every $s' \models p$ is a deadend. A set of *forbidden state-action pairs* (FSAPs) is computed by analyzing the deadends with respect to all existing action outcomes. More precisely, when one outcome of an action leads recognizably to a deadend in a particular state, the action is forbidden from being used during search. In subsequent calls to Algorithm 1, PRP resets the policy and the FSAPs restrict the search for a weak plan. PRP is guaranteed to converge and find a strong cyclic plan when all deadends in the problem, if any, are avoidable.

```
Input: FOND planning task $\mathcal{P} = \langle \mathcal{V}, s_0, s_\star, \mathcal{A} \rangle$
Output: Partial policy $\pi$
InitPolicy();
while $\pi$ changes do
    Open $\leftarrow \{s_0\}$ ;
    Seen $\leftarrow \{\}$ ;
    while Open $\neq \emptyset$ do
        $s$ = Open.pop();
        if $s \not\models s_\star \wedge s \notin$ Seen then
            Seen.add(s);
            if $\pi(s)$ is undefined then
                GenPlanPairs($\langle \mathcal{V}, s, s_\star, \mathcal{A} \rangle, \pi$);
            if $P(s)$ is defined then
                $\langle p, a \rangle = \pi(s)$;
                for $e \in \mathit{Eff}_a$ do
                    Open.add($Prog(s, a, e)$);
    ProcessDeadends();
return $\pi$;
```
Algorithm 1: Generate Strong Cyclic Plan

Theorem 1 (Muise, McIlraith, and Beck 2012). *PRP computes a strong cyclic plan for FOND problems that do not contain unavoidable deadends.*

PRP's state-action avoidance, and succinct state representations are central to its superior performance. More recently, PRP has additionally been extended to handle conditional effects of actions (Muise, McIlraith, and Belle 2014).

3.2 From FOND to HighProb

The similarities between FOND and HighProb models, and recent advances in FOND technology motivate the use of state-of-the-art FOND technology to solve HighProb. It is well-known that solutions to the FOND problem FOND($\mathcal{P}$) that result from ignoring transition probabilities in a probabilistic planning problem $\mathcal{P}$ are also solutions to $\mathcal{P}$. Moreover, strong-cyclic solutions to FOND($\mathcal{P}$) are optimal HighProb solutions to $\mathcal{P}$ and vice-versa (cf. (Hertle et al. 2014)).

Lemma 1. *Let $\mathcal{P}$ be a HighProb planning problem. If π is a solution for FOND($\mathcal{P}$), then π is a solution for $\mathcal{P}$.*

Lemma 2. *Let $\mathcal{P}$ be a HighProb 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 HighProb planning problem $\mathcal{P}$ has an optimal solution that reaches the goal with probability* 1 *iff FOND($\mathcal{P}$) has a strong cyclic solution.*

3.3 From PRP to ProbPRP

We exploit core similarities between the FOND and probabilistic planning models, to extend the state-of-the-art FOND planner, PRP (Muise, McIlraith, and Beck 2012), into a HighProb probabilistic planner that we call ProbPRP. ProbPRP benefits from the techniques present in PRP, including the partial-state representation and FSAP deadend avoidance that results in faster convergence rate and more compact policies. In addition, ProbPRP implements new features that result in better-quality HighProb solutions. We detail these features below.

Bias towards high-likelihod plans In PRP, the policy under construction is extended with plans that map unhandled states in the *Open* list to either a goal state or to a state that is already handled by the policy (i.e. for which there exist a strong-cyclic plan). The partial-state representation facilitates state entailment, and ultimately benefits creation of smaller policies. Whereas smaller policies are preferred, subsequent plans to the goal may be unnecessarily long – a property of the solutions that is not considered in the FOND model. To mitigate for this, ProbPRP focuses the search on high likelihood plans. This method has been used in the past to search for plans in the determinization relaxation that minimize the risk of failing (Jimenez, Coles, and Smith 2006). The rationale in ProbPRP is slightly different. Intuitively, high-likelihood plans are potentially short plans that contain most-probable outcomes of the actions. ProbPRP biases the search towards short plans of high likelihood. Our tests revealed that searching for plans that maximize L_P leads to policies with lower expected plan length. In terms of implementation, ProbPRP modifies the search process in GENPLANPAIRS, and uses heuristic search to find minimum-cost plans (not necessarily optimal) with associated transition costs set to $-log(T(s_i, a_i, s_{i+1}))$.

Final FSAP-Free Iteration The forbidden state-action pair (FSAP) mechanism prunes the search by state-action pairs that lead to a deadend with non-zero probability. The FSAP mechanism proved to be effective in reducing the size of the search space, but it can be too aggressive in domains with unavoidable deadends where FSAPs still lead to plans that reach the goal state with non-zero probability. Based on this observation, ProbPRP performs a final FSAP-free iteration, where the policy with highest probability of success found is used to initialize the policy on line 1 of Algorithm 1, and the problem is solved *with forbidden state-action pairs and deadend detection disabled*. This final pass optimistically closes every *Open* state in a best effort manner. The returned policy handles a superset of the states that it handled previously, thus improving the solution quality.

Safety Belt Mechanism The idea of a *safety belt* mechanism is to gradually disable a feature when it is detected not to be contributing to the solver's progress. The current version of ProbPRP implements a safety belt for the strong-cyclic detection (SCD) aspect of the planner which prunes the simulated policy based on sufficient conditions. If the SCD mechanism is consistently never used to detect states, then the (potentially costly) SCD computation is gradually disabled and used less over time.

Eliminating Non-Robust Plans When strong cyclic solutions exist, the policy will never include non-robust plans. Deletion of non-robust plans in PRP's algorithm is delayed until a deadend state in the *Open* list is processed. One way to accelerate convergence is to detect during GENPLANPAIRS search those plans that are easily determined to be non-robust. In particular, when a weak plan non-deterministically leads to a deadend, ProbPRP will find it, compute the FSAPs, and start the search again.

Policy Optimization To reduce the number of state-action pairs in the solution found by ProbPRP, a *simulation* checks all the states reachable by the policy and discards the portion of the policy that is no longer used in the solution.

Soundness and completeness of ProbPRP follows from the soundness and completeness of PRP. Optimality follows from Theorem 1 and the correspondence of solutions detailed in Lemmas 1 – 3. It can be shown that the different features implemented in ProbPRP do not affect the soundness and completeness of the algorithm.

Theorem 2. *For HighProb planning problems, the ProbPRP algorithm is sound and complete in general, and optimal for problems with no unavoidable deadends.*

Corollary 1. *If the solution to the HighProb planning problem $\mathcal{P}$ found by ProbPRP has probability of success lower than 1, then $\mathcal{P}$ has unavoidable deadends.*

ProbPRP benefits from the compact state representation in PRP and its potential to find small policies. Moreover, the solutions found by ProbPRP are computed *offline*, with the advantage over *online* planners like RFF that no further computation is needed during execution. In addition, it is possible to estimate the quality of offline solutions prior to execution – e.g. using Monte Carlo simulations as done in ProbPRP– or even to compute it analytically.

4 Evaluation

We evaluate the performance of ProbPRP in a selection of benchmark domains from past IPPC competitions. MaxReward problems were transformed into HighProb by ignoring rewards and asking for solutions that maximize the probability of success. We used the client-server architecture MDPSim (Asmuth, Littman, and Younes 2007) – used in past IPPC editions – to simulate execution of the solutions, and we average results over 100 runs per problem. All experiments were conducted on a Linux server with an Intel Xeon W3550 CPU @3.07GHz, limiting each process to 2GB memory usage and 30 minutes run time. More detailed results are shown and discussed in a technical report (Camacho, Muise, and McIlraith 2016).

4.1 Benefits of ProbPRP's Features

ProbPRP found HighProb solutions of improved quality when compared to the strong-cyclic solutions found by PRP in the naïve FOND relaxation of those problems. More precisely, the bias towards exploration of high-likelihood plans results in solutions with smaller expected plan length: in the three largest instances of the Boxworld domain, it decreases from more than 1000 to nearly 160 actions. Interestingly, the bias towards high-likelihood plans do not compromise the policy size and success rate, which remain similar in most problems. The final FSAP-free iteration is useful when the probability of success of the best incumbent policy found by ProbPRP is lower than 1. This is the case of problems with unavoidable deadends where, in the most beneficial cases, the probability of success increases by 30%.

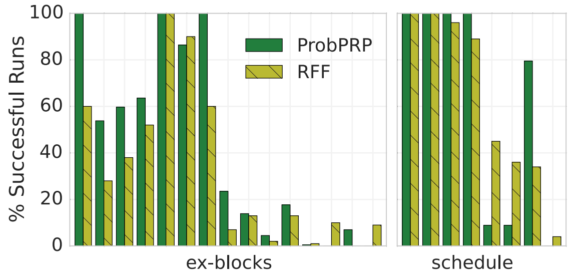


Figure 1: Success rate of the solutions found by ProbPRP and RFF in the *ex-blocksworld* and *schedule* domains.

	p01	p02	p03	p04	p05	p06	p07
Optimal	100%	62%	61%	60%	100%	98%*	100%*
ProbPRP	100%	54%	60%	59%	100%	86%	100%

Table 1: Success rate of solutions to the *ex-blocksworld*.

The final FSAP-free iteration is equally useful when the core search time in ProbPRP is limited. In the Schedule problems, when the search time is limited to 5 seconds the success rate increases from 10% (no FSAP-free round) to 72% (final FSAP-free round) in the most beneficial case. Finally, the Safety Belt mechanism gradually disables strong-cyclic detection when is not being beneficial. In the biggest instances of Boxworld, it reduces the run times from 160 to 3 seconds.

4.2 Comparison with RFF

We compared ProbPRP with the previous state of the art in HighProb, RFF (Teichteil-Königsbuch, Kuter, and Infantes 2010). RFF constructs a policy envelope incrementally until the probability of falling outside the envelope during execution is lower than ρ. In that case, RFF replans by constructing another envelope rooted in the current state. We used the configurations that manifested best global results. Namely, ProbPRP was configured to use h_{add} heuristics, deadend detection, and SCD safety belt. We configured RFF to search in the most-probable outcome determinization with the Best Goals strategy and $\rho = 0.2$. Remarkably, despite the fact that RFF can replan and ProbPRP (an offline planner) cannot, our planner demonstrates better performance and scalability, and produces solutions of better quality as we explain below.

Success Rate ProbPRP and RFF both succeed in finding optimal solutions to the *blocksworld*, *boxworld*, and *triangle-tireworld* problems, which have avoidable or no deadends. A small exception occurs in RFF's solutions to the three largest instances of the *blocksworld* and *boxworld* domains (p13-p15). In these, we detected looping behaviour and RFF's success rate is reduced to zero. In *ex-blocksworld* (a domain with unavoidable deadends) both planners experience difficulty finding optimal policies. However, solutions found by ProbPRP have, near consistently, greater success rates (Figure 1). Although ProbPRP does not offer any guarantees on optimality, its solutions are nonetheless close to the theoretical optimal probability of success. Table 1 shows the percentage of successful runs of the solutions found by *Value Iteration* and ProbPRP. Values marked with an asterisk (*) refer to the best result obtained by any of the IPPC-08 competitors. In the *schedule* domain (a domain with a mix of (un)avoidable deadends) both planners have difficulty scaling to large problems. The execution of ProbPRP in the four largest problems times out before convergence. Figure 1 shows the success rate when the final FSAP-free round in these four largest problems gets 15 minutes; half of the total search time allocated. An interesting phenomenon occurs: ProbPRP finds smaller policies with better success rates when the core search time is reduced to 5 seconds. With the restricted search time, ProbPRP finds solutions with a much smaller size (order of magnitude) and with considerably higher success rates (respectively, 72%, 72%, 85%, and 0.6% for the four largest problems). We conjecture this behaviour exists because the incumbent policy found earlier in the core search has a greater potential for high likelihood plans in the final FSAP-free round. This motivates future research in selecting more suitable policies for the final FSAP-free round.

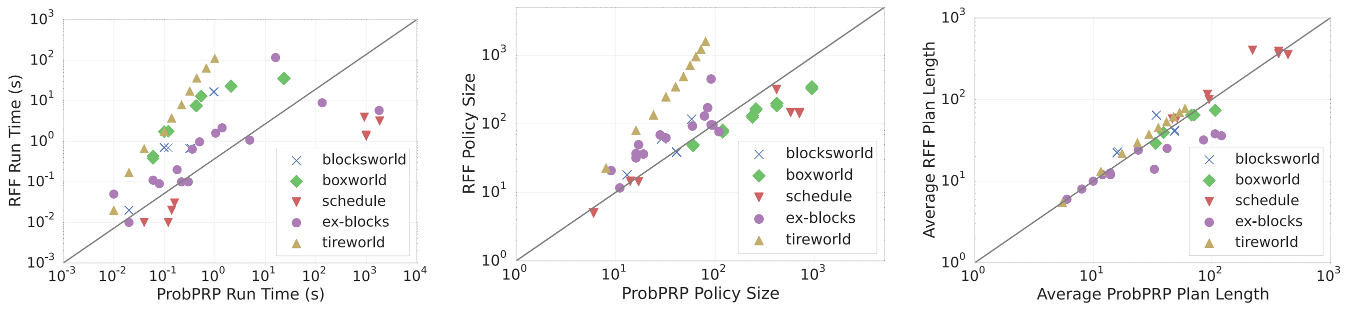
Quality of Solutions As depicted in Figure 2, ProbPRP's run time is comparable to RFF's in some cases, but in many others it is order(s) of magnitude better, while exhibiting equal or better success rates. Anecdotally, by the time that RFF completes the first envelope, ProbPRP can generally find a more robust *offline* solution. The policies computed by ProbPRP are smaller than those computed by RFF (sometimes orders of magnitude smaller), while maintaining a similar expected plan length. This is most evident in domains with avoidable deadends, such as *triangle-tireworld*, where the FSAP mechanism bypasses the deadends efficiently.

Robustness of the Algorithm We conducted two tests in *triangle-tireworld* problems to evaluated the robustness of our algorighm with respect to small perturbations in the transition probabilities. First, we decreased the probability of the *move* action resulting in a flat tire from 50% to 45%. Second, we kept this probability to 50% but varied the order in which action effects are declared. RFF was very sensitive to probability fluctuations, and the success rate of its solutions decreased dramatically down to 1% or below for problems 4-10. ProbPRP was still able to find strong-cyclic solutions with good performance.

5 Summary and Discussion

Probabilistic planning in the presence of avoidable and/or unavoidable deadends is a challenging and important task (Kolobov, Mausam, and Weld 2012). We introduced ProbPRP, a planner that exploits key techniques from state-of-the-art FOND planner, PRP, to compute offline solutions to HighProb problems. ProbPRP is sound and complete in general, and optimal for problems where the deadends are avoidable.

HighProb solutions found by ProbPRP are often optimal or close to optimal, and outperform the solutions found by the incumbent state-of-the-art HighProb planner, RFF.



(a) Run Time Comparison (b) Policy Size Comparison (c) Avg. Plan Length Comparison

Figure 2: Solutions to probabilistic planning problems found by ProbPRP and RFF. Results averaged over 100 runs per problem.

ProbPRP's solutions nicely balance policy size, compactness, and the expected length of plans. Moreover, ProbPRP 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.

We are not alone in recognizing the computational core that is shared by FOND and probabilistic planning (cf. (Hertle et al. 2014)). With this work, we have demonstrated the merit of this correspondence by exploiting compact policy representations, relevance reasoning, and deadend avoidance techniques 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).

References

Alford, R.; Kuter, U.; Nau, D. S.; and Goldman, R. P. 2014. Plan aggregation for strong cyclic planning in nondeterministic domains. *Artificial Intelligence* 216:206–232.

Asmuth, J.; Littman, M.; and Younes, H. 2007. MDPSim 2.2. PPDDL plan evaluator simulator. https://github.com/hlsyounes/mdpsim.

Camacho, A.; Muise, C.; and McIlraith, S. A. 2016. Appendix: From fond to robust probabilistic planning. Technical Report CSRG-630, Department of Computer Science, University of Toronto.

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.

Fu, J.; Ng, V.; Bastani, F. B.; and Yen, I. L. 2011. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. *Proc. of the 22nd Int'l Joint Conference on Artificial Intelligence (IJCAI)* 1949–1954.

Hertle, A.; Dornhege, C.; Keller, T.; Mattmller, R.; Ortlieb, M.; and Nebel, B. 2014. An Experimental Comparison of Classical, FOND and Probabilistic Planning. In *Proc. of the 37th Int'l Conf. on Artificial Intelligence*, 297–308.

Jimenez, S.; Coles, A.; and Smith, A. 2006. Planning in probabilistic domains using a deterministic numeric planner. *Proc. of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG).*

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. *Proc. of the 22th Int'l Conference on Automated Planning and Scheduling (ICAPS).*

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic Search for Generalized Stochastic Shortest path MDPs. *Proc. of the 21th Int'l Conference on Automated Planning and Scheduling (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.

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010. Pattern database heuristics for fully observable nondeterministic planning. In *Proc. of the 20th Int'l Conference on Automated Planning and Scheduling (ICAPS)*, 105–112.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-deterministic Planning by Exploiting State Relevance. In *Proc. of the 22th Int'l Conference on Automated Planning and Scheduling (ICAPS)*, 172–180.

Muise, C.; McIlraith, S. A.; and Belle, V. 2014. Non-deterministic planning with conditional effects. In *Proc. of the 24th Int'l Conference on Automated Planning and Scheduling (ICAPS)*, 370–374.

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. 2012. Stochastic Safest and Shortest Path Problems. *Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI)* 1825–1831.