An Experimental Comparison of Classical, FOND and Probabilistic Planning

Andreas Hertle, Christian Dornhege, Thomas Keller,
Robert Mattmüller, Manuela Ortlieb, and Bernhard Nebel

University of Freiburg, 79110 Freiburg, Germany,
{hertle, dornhege, tkeller, mattmuel,
ortlieb, nebel}@informatik.uni-freiburg.de

Abstract. Domain-independent planning in general is broadly applicable to a wide range of tasks. Many formalisms exist that allow the description of different aspects of realistic problems. Which one to use is often no obvious choice, since a higher degree of expressiveness usually comes with an increased planning time and/or a decreased policy quality. Under the assumption that hard guarantees are not required, users are faced with a decision between multiple approaches. As a generic model we use a probabilistic description in the form of Markov Decision Processes (MDPs). We define abstracting translations into a classical planning formalism and fully observable nondeterministic planning. Our goal is to give insight into how state-of-the-art systems perform on different MDP planning domains.

1 Introduction

Domain-independent planning is used to solve problems from various domains, including tasks from real-world robotics applications. Often, such tasks feature aspects that go beyond classical planning, such as nondeterministic or probabilistic effects, partial observability, etc. However, when modeling a problem, there is a tradeoff between modeling as many of these aspects as possible and finding solutions fast. Different existing planning formalisms capture different (combinations of) aspects of real-world problems. Modeling and abstracting away different aspects, they also induce different solution concepts. Which formalism is the best for a specific problem is not always obvious. As long as the application allows it, a less expressive formalism is chosen often in practice.

In this work, we study the question how to best deal with probabilistic action outcomes when modeling and solving a planning task. *Classical planning* formulations only model deterministic actions. They are often used by embedding the planner in an execution-monitoring-planning loop that replans on unexpected outcomes. *Fully observable nondeterministic (FOND)* planning explicitly considers nondeterministic actions and produces strategies that guarantee to reach a goal. *Probabilistic planning* additionally considers outcome probabilities and aims to maximize the expected accumulated reward.

Since different formulations give different guarantees on the solution, an entirely fair comparison is impossible. We therefore compare classical planning, FOND planning and probabilistic planning experimentally and emphasize differences in solution quality, planning time and the ability to avoid deadend states of the considered algorithms. Since our planning systems [7, 16, 11] require different levels of abstraction regarding the input, we also provide abstracting translations between the planning formalisms.Our goal is to show how state-of-the-art planning systems cope with different problems in a realistic setting, a problem that users are commonly faced with when applying planning to solve actual tasks, which also sheds some light on claimed advantages in speed or solution quality.

We model three domains: one originally used in probabilistic planning, one from FOND planning, and a classical planning domain that is derived from a robotic planning scenario. Probabilistic action outcomes of the latter two are added in a first step, such that all domains are available as a Factored MDP [1]. These are then translated into corresponding FOND and classical planning formulations following the set of rules described in Sec. 4, and evaluated with three state-of-the-art planning systems in Sec. 5. Prior to that, we describe related work and give formal definitions for the used planning formalisms.

2 Related Work

Classical planners have been successfully integrated in robotic systems – a real-world domain known for unexpected outcomes. Kaelbling et al.[8] developed a robot planning system that integrates task and motion planning. This system uses a hierarchical regression planner where a refined prefix is executed directly. They argue that "there are few catastrophic or entirely irreversible outcomes". Following similar arguments, Nebel et al. [18] and Keller et. al [12] demonstrate how a classical planner embedded in a continual planning loop solves complex mobile manipulation tasks on a robot. Replanning is used to deal with execution failures and unexpected situations. In their KVP system, Gaschler et al. [4] combine the power of a symbolic planner with efficient geometric computations to create a framework for knowledge based planning in a real robot environment.

In recent years, nondeterministic planners have been improved in efficiency. Recently, compilation approaches that compute strong cyclic plans with classical planners became popular [14, 3, 17]. Mattmüller et al. [16] show how pattern database heuristics can be used in fully observable nondeterministic domains to guide the search more efficiently towards goal states. Little and Thiébaux [15] investigate the notion of probabilistic interestingness, where they investigate the effectiveness of replanning or compilation approaches to solve probabilistic planning tasks. Keller and Eyerich [11] developed Prost, a domain independent probabilistic planning system based on the UCT* algorithm [13], which is able to improve the quality of the solution policy by identifying unreasonable actions and actively searching for dead ends and goals in the MDP with a reward lock detection procedure.

3 Planning Formalisms

In this section, we describe the planning formalisms we use to specify classical, FOND, and probabilistic planning tasks.

3.1 Classical Planning

A *classical SAS*$^+$ planning task is a tuple $\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{O} \rangle$ consisting of the following components: $\mathcal{V}$ is a finite set of *state variables* v, each with a finite *domain* $\mathcal{D}_v$ and an *extended domain* $\mathcal{D}_v^+ = \mathcal{D}_v \uplus \{\bot\}$, where $\bot$ denotes the *undefined* or *don't-care* value. A *partial state* is a function s with $s(v) \in \mathcal{D}_v^+$ for all $v \in \mathcal{V}$. We say that s is *defined* for $v \in \mathcal{V}$ if $s(v) \neq \bot$. A *state* is a partial state s that is defined for all $v \in \mathcal{V}$. The set of all states s over $\mathcal{V}$ is denoted as $\mathcal{S}$. Depending on the context, a partial state s_p can be interpreted either as a *condition*, which is *satisfied* in a state s iff s agrees with s_p on all variables for which s_p is defined, or as an *update* on a state s, resulting in a new state s' that agrees with s_p on all variables for which s_p is defined, and with s on all other variables. The *initial state* s_0 of a problem is a state, and the *goal description* $s_\star$ is a partial state. A state s is a *goal state* iff $s_\star$ is satisfied in s. $\mathcal{O}$ is a finite set of *actions* of the form $a = \langle \mathit{pre}, \mathit{eff} \rangle$, where the *precondition* pre and the *effect* eff are partial states. The *application* of an outcome eff to a state s is the state $\mathit{app}(\mathit{eff}, s)$ that results from updating s with eff. An action is *applicable* in s iff its precondition is satisfied in s. The application of a to s is $\mathit{app}(a, s) = \mathit{app}(\mathit{eff}, s)$ if a is applicable in s, and undefined otherwise. Solutions to a *classical* planning task $\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{O} \rangle$ are *plans*, i.e. sequences of $a_0, a_1, \dots, a_n$, where $a_i \in \mathcal{O}$, $i = 0, \dots, n$, a_i is applicable in s_i, $\mathit{app}(\mathit{eff}_i, s_i)$ results in s_{i+1} for $i = 0, \dots, n-1$, and $s_\star$ is satisfied in s_{n+1}.

3.2 Nondeterministic Planning

A *fully observable nondeterministic (FOND) SAS*$^+$ planning task is a tuple $\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{O} \rangle$ with the same $\mathcal{V}$, s_0, and $s_\star$ as a classical planning task and *nondeterministic* actions $\mathcal{O}$ of the form $a = \langle \mathit{pre}, \mathit{Eff} \rangle$ with preconditions pre as before, but finite *sets* Eff of possible effects, the *nondeterministic outcomes* of a. Each $\mathit{eff} \in \mathit{Eff}$ is a partial state as before. The application of a set Eff to a state s is the set of states $\mathit{app}(\mathit{Eff}, s) = \{\mathit{app}(\mathit{eff}, s) \mid \mathit{eff} \in \mathit{Eff}\}$ that might be reached by applying a nondeterministic outcome from Eff to s. The application of a to s is $\mathit{app}(a, s) = \mathit{app}(\mathit{Eff}, s)$ if a is applicable in s, and undefined otherwise. Solutions to a planning task Π are now *strategies*, i.e., mappings $\pi \colon \mathcal{S}_\pi \to \mathcal{O} \cup \{\bot\}$ for a set of states $\mathcal{S}_\pi \subseteq \mathcal{S}$ such that $\pi(s) = \bot$ iff s is a goal state and that for all nongoal states s in $\mathcal{S}_\pi$, the action $\pi(s)$ is applicable in s and $\mathcal{S}_\pi$ contains all states in $\mathit{app}(\pi(s), s)$. A strategy π is a *strong cyclic plan* ("trial-and-error strategy") for a planning task Π iff $s_0 \in \mathcal{S}_\pi$ and for each state s reachable from state s_0 following strategy π, a goal state is reachable from s following strategy π.

3.3 Probabilistic Planning

A factored, finite-horizon MDP [1] with initial state and goal is a seven-tuple $\langle \mathcal{V}, \mathcal{O}, s_0, s_\star, P, R, H \rangle$, where the set of states $\mathcal{S}$ is induced by the set of *state variables* $\mathcal{V}$ and the *remaining steps* $h \in \{0, \dots, H\}$ as $\mathcal{S} = 2^V \times \{0, \dots, H\}$. $\mathcal{O}$ is a finite set of *actions*, $s_0 \in \mathcal{S}$ is the *initial state*, $s_\star$ is a *goal description* as above, $P : \mathcal{S} \times \mathcal{O} \times \mathcal{S} \to [0,1]$ is the *transition function* which gives the probability $P(s'|a,s)$ that applying action $a \in \mathcal{O}$ in state $s \in \mathcal{S}$ leads to state $s' \in \mathcal{S}$, $R : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ is the *reward function*, and $H \in \mathbb{N}$ is the *horizon* which specifies the number of decisions before each run terminates. For the purpose of this paper, we assume that the reward function has a special structure, specifically that the reward reflects (unit) action costs and numbers of unsatisfied goals. More formally, we let $\mathit{unsat}(s)$ be the number of variables v such that $s_\star(v)$ is defined and $s(v) \neq s_\star(v)$. Then, $R(s,a) = -(\mathit{unsat}(s) + 1)$, if s is not a goal state, and $R(s,a) = 0$, otherwise.

4 Translation Between Planning Formalisms

Conversion from the probabilistic MDP formalism to the nondeterministic and classical planning formalism follows a set of rules that we present here. The conversion process could be completely automated and does therefore not require any domain-specific knowledge. However, in this paper we convert the domains manually following those rules.

All three planning formalisms work on the same finite set of states $\mathcal{S}$. Given the goal specification for an MDP is derived from a partial state $s_\star$ as described in Sec. 3.3, we use that same partial state $s_\star$ as the goal state for classical and FOND planning. The major differences in the formalisms now lie in how actions and action costs are derived from a given MDP.

4.1 Translating MDP to FOND Planning

An MDP action a is translated to a set of FOND actions. First, we compute all possible predecessors, in which a could have been applied.

$$\mathit{Pre} = \{s | P(s'|a,s) > 0; s, s' \in \mathcal{S}\} \tag{1}$$

Each such state potentially can produce multiple outcomes. We obtain a set of effect states Eff_i for each state $\mathit{pre}_i \in \mathit{Pre}$.

$$\mathit{Eff}_i = \{s' | P(s'|a, \mathit{pre}_i) > 0; s' \in \mathcal{S}\} \tag{2}$$

For each state $\mathit{pre}_i \in \mathit{Pre}$ we produce a FOND action with outcomes Eff_i, so that the MDP action a results in a set of FOND actions $\{\langle \mathit{pre}_i, \mathit{Eff}_i \rangle | \mathit{pre}_i \in \mathit{Pre}\}$. The actions obtained with these rules have full states as preconditions and effects. However, logical simplifications can drastically reduce the number of generated actions by summarizing the full states into partial states. In practice often one MDP action translates to one FOND action.

4.2 Translating MDP to Classical Planning

For converting MDP actions to classical actions we determine predecessors $pre_i \in Pre$ and effect outcomes Eff_i from an MDP action a analogously to the FOND translation. In contrast to FOND, classical actions allow only one deterministic effect eff. There are two commonly used determinizations: all-outcome determinization and most-likely determinization. Both assume that a specific effect can be chosen and plan accordingly.

An all-outcome determinization creates a separate classical action for each possible effect $\mathit{eff} \in \mathit{Eff}_i$. For each action in the MDP, we therefore obtain a set of actions with an entry for each predecessor $pre_i \in Pre$:

$$\{\langle pre_i, \mathit{eff}\rangle | \mathit{eff} \in \mathit{Eff}_i\} \tag{3}$$

Even though there are techniques that only lead to a polynomial blowup of the number of actions in the determinization [10], this can still be prohibitively large in practice. An alternative is the most-likely determinization, which only considers the effect with the highest probability. For a predecessor $pre_i \in Pre$ the most-likely effect $\mathit{eff}_{\max}$ is determined from all possible outcomes Eff_i.

$$\mathit{eff}_{\max} = \underset{\mathit{eff} \in \mathit{Eff}_i}{\operatorname{argmax}} P(\mathit{eff} | a, pre_i) \tag{4}$$

Now for each predecessor $pre_i \in Pre$ only a single most-likely action is created as $\langle pre_i, \mathit{eff}_{\max}\rangle$. Similarly to FOND we apply logical simplifications when possible to reduce the number of distinct actions for both determinizations.

As planners aim for minimal cost plans a meaningful action cost that considers the operator cost and probability $p = P(\mathit{eff} | a, pre_i)$ for an all-outcome operator, or $p = P(\mathit{eff}_{\max} | a, pre_i)$ for a most-likely determinized operator is beneficial to improve the quality of resulting plans. Under the assumption that either that outcome happens with probability p or the state is unchanged with $1-p$, we use the expected cost, when retrying an action with unit cost until the desired outcome is reached.

$$\sum_{i=1}^{\infty} (1-p)^{i-1} \cdot p \cdot i = \frac{1}{p} \tag{5}$$

If action outcomes with unit cost have probability p we therefore use $\frac{1}{p}$ as the operator cost in the classical planning formulation. Other examples for combining operator cost and probability are to use the negative logarithm of p, which produces the most-likely plans (ignoring cost) or a weighted combination of $\mathit{cost}(a)$ and $-\log(p)$ [9].

4.3 Theoretical and Practical Properties of Translations and Planning Algorithms

Classical planning, FOND, and probabilistic planning differ both in their theoretical and their practical computational properties. On the theoretical side, classical planning with all-outcome determinization, FOND planning and probabilistic planning are guaranteed to preserve MDP goal paths in the translated model. For classical planning with most-likely determinization, this is not the case. All goal paths can be lost in the determinization. Classical planning with all-outcome determinization and probabilistic planning have the property that MDP plan existence implies that the translated models still have solutions. Classical planning with most-likely determinization and FOND planning do not have this property. The major advantage of probabilistic planning is the guarantee to find a *reward-optimal* solution in the limit of long deliberation time per step. None of the other approaches has the same guarantee. On the other hand, given that we usually do not compute optimal MDP policies online, offline FOND planning is the only approach that is guaranteed to *avoid dead-ends* at execution time. It does so at the expense of a significant amount of *offline planning time* that the other approaches avoid. Finally, at execution time, the high offline planning time is compensated for by fast state-action table lookups. Online planning speed (average response time) of classical (re-)planning and online probabilistic planning are incomparable, since the latter can use an arbitrary timeout for each step and return the best action so far, whereas classical planning has to expend at least the amount of time necessary to find *some* classical plan.

5 Evaluation

We use three different domains, each one originating from a different planning formalism to give a balanced evaluation. Each domain has been formulated as a factored MDP and was translated to classical and FOND planning.

5.1 Domains

MobileManipulation. In the MOBILEMANIPULATION domain, an autonomous service robot operates in a house. The robot is equipped with two arms and sensors to perceive the environment. There are multiple rooms with a number of objects located on tables. The goal is to tidy up the rooms, i.e. find all objects, pick them up and bring them to a specified destination table. In addition all tables should be wiped clean. Within a room, the robot can move freely between tables. Between rooms doors might need to be opened. The robot traverses through open doors with arms either close to the robot body or not. With retracted arms, success probabilities are higher (0.9 in comparison to 0.4). When close to a table the robot can choose to perceive object locations increasing the success probabilities for manipulation actions from 0.3 to 0.9. Objects can be picked up with either hand and brought to any table. Manipulation can go wrong, as the robot might be unable to grasp an object or might topple another object on the table. Should an object happen to fall to the floor, the robot will not be able to recover it. Only tables cleared of all objects can be wiped with a sponge.

TriangleTireworld. The TriangleTireworld domain was introduced by Little and Thiébaux [15] and is a special case of the IPC Tireworld domain, in which a car has to drive from an initial to a goal location along directed edges. In each step, a tire can go flat probabilistically (we use $p = 0.2$). Before moving further, it has to be fixed, which is only possible if a spare tire is present in the current location. Only a subset of the locations contain a spare, so success is only guaranteed if the car follows a path such that every location along the path (except the goal) has a spare. The TriangleTireworld domain is designed for offline planners to outperform replanners. This is achieved by requiring a particular structure of the roadmap graph: The locations form a triangle with corners A, B and C, the start is A, the goal is B, and the only safe path with spares in all locations is the maximal detour from A via C to B. Replanners trying to find shortest paths from A to B have a high probability of getting stuck, whereas offline planners should find and follow the only safe path. Instances of the TriangleTireworld domain vary in their numbers of locations.

EarthObservation. The EarthObservation domain models a satellite orbiting the earth. It can take pictures of the landscape below with a camera. The landscape is subdivided into square regions of interest forming a grid wrapped around a cylindrical projection of the earth surface. The camera focuses on one region at a time and can be shifted north or south. It can take a picture of the region currently in focus. The focus may not be shifted while taking a picture. Regardless whether the focus is shifted or a picture is taken, the satellite travels eastward around the earth, shifting the focus one grid cell to the east in addition to the other effects in each step. The objective is to take pictures of certain regions in a limited timeframe with as few shifts as possible. Taking a picture of a region does not guarantee good image quality: the worse the weather, the lower the chance of success. Over time the visibility in each region can change probabilistically, and changes between similar levels of visibility are more likely than vast changes. Apart from the weather change probabilities, which vary between 0.01 and 0.5, instances of the EarthObservation domain differ in the numbers of grid cells and imaging objectives.

5.2 Planners

We use state-of-the-art planners from the field of classical planning (Fast Downward), fully observable nondeterministic planning (myND) and probabilistic planning (Prost). We give a short overview of the underlying approaches.

Fast Downward. For classical planning, we use the Fast Downward planning system [7] in the LAMA 2011 configuration. In this setting, the planner first looks for a suboptimal plan with Greedy Best First search. When a plan is found, the search engine is switched to weighted A* to look for plans of higher quality. We let the planner search until an optimal plan is found or the timeout of 90 seconds is reached. Since we only produce classical plans for probabilistic problems, we have to deal with unexpected results when executing a plan on the plan simulator. Therefore, we wrap an execute-monitor-replan loop around FAST DOWNWARD. When monitoring determines that a plan is invalid, a new planning process is initiated for the current situation.

MYND. For offline planning, we use the FOND variant of the MYND planner [16] that uses LAO* search [5] guided by the canonical PDB heuristic [2, 6]. We use the goal variables as singleton patterns for the pattern collection. The planner outputs strong cyclic plans in the form of state-action tables that are then interpreted by an execution simulator. For each single planning task, we set an offline planning time limit of 30 minutes. Lookup times during plan simulation are negligible.

PROST. We use the PROST planning framework [11] for MDP planning, equipped with the UCT* algorithm [13] as used in the configuration that won IPPC 2014. This search procedure combines dynamic programming, heuristic search and Monte-Carlo tree search to an algorithm that is asymptotically optimal in the limit, but which is also able to make decisions under tight time constraints in an online fashion. In our experiments, a time limit of one second per simulation step was used. The heuristic is the base heuristic of PROST [11], which is based on an iterative deepening search in the most-likely determinization.

	Response Time [s]		Reward		
	eff $_{\max}$	MYND	*eff* $_{\max}$	MYND	PROST
1	8 ± 4	1	-151 ± 72	-66 ± 8	-62 ± 17
2	1 ± 1	0	-9 ± 3	-11 ± 6	-9 ± 4
3	16 ± 3	1	-391 ± 103	-109 ± 34	-85 ± 25
4	124 ± 93	3	-1141 ± 24	-399 ± 56	-355 ± 65
5	75 ± 31	3	-894 ± 35	-294 ± 64	-283 ± 46
6	9 ± 4	0	-172 ± 80	-126 ± 29	-67 ± 14
7	6 ± 3	0	-86 ± 39	-46 ± 14	-48 ± 16
8	3 ± 2	0	-16 ± 8	-10 ± 5	-10 ± 6
9	41 ± 6	2	-908 ± 146	-290 ± 94	-172 ± 35
10	39 ± 33	156	-2203 ± 18	-885 ± 187	-677 ± 97
11	1 ± 1	0	-10 ± 5	-15 ± 8	-18 ± 15
12	57 ± 12	4	-1191 ± 93	-301 ± 54	-237 ± 45
13	163 ± 53	15	-1481 ± 146	-341 ± 74	-275 ± 47
14	38 ± 8	2	-614 ± 160	-179 ± 44	-165 ± 68
15	128 ± 100	40	-2717 ± 61	-1015 ± 277	-826 ± 148
16	3 ± 1	0	-24 ± 10	-20 ± 7	-18 ± 9
17	478 ± 192	102	-2316 ± 125	-646 ± 200	-416 ± 76
18	230 ± 78	15	-1807 ± 80	-527 ± 111	-394 ± 81
19	466 ± 154	90	-2038 ± 116	-604 ± 202	-497 ± 90
20	101 ± 74	52	-2871 ± 29	-1144 ± 219	-814 ± 151

Table 1. Average response time and rewards for the EARTHOBSERVATION domain, the average of 100 runs is shown. Time values for MYND include offline planning time.

	Response Time [s]			Reward			
	Eff⋆	*eff*$_{\max}$	MYND	*Eff*⋆	*eff*$_{\max}$	MYND	PROST
1	3 ± 1	2 ± 0	1	-27 ± 1	-27 ± 1	-27 ± 1	-75 ± 44
2	16 ± 6	7 ± 3	1	-82 ± 7	-81 ± 4	-84 ± 4	-218 ± 92
3	109 ± 34	34 ± 9	143	-134 ± 11	-132 ± 8	-156 ± 7	-419 ± 18
4	125 ± 45	113 ± 54	3	-123 ± 7	-127 ± 6	-146 ± 6	-396 ± 72
5	367 ± 187	218 ± 88	4	-190 ± 13	-191 ± 9	-217 ± 10	-805
6	530 ± 195	335 ± 127	5	-283 ± 16	-279 ± 11	-286 ± 14	-1005
7	312 ± 165	223 ± 109	3	-187 ± 14	-184 ± 7	-240 ± 8	-805
8	470 ± 193	299 ± 132	11	-307 ± 32	-278 ± 11	-310 ± 8	-1505
9	613 ± 259	492 ± 175	14	-413 ± 18	-421 ± 18	-460 ± 17	-1805
10	1106 ± 244	611 ± 291	134	-604 ± 59	-967 ± 656	-645 ± 18	-2105

Table 2. Average response time and rewards for the MOBILEMANIPULATION domain, the average of 100 runs is shown. Time values for MYND include offline planning time.

5.3 Experiments

In this paper we compare three planning paradigms that not only differ in their expressivity but also have unique plan representations. To find common ground to evaluate the quality of the produced plans we use Scott Sanner's rddlsim[1], the simulator of the International Probabilistic Planning Competition (IPPC). For each of the three planning domains we provide between 10 and 20 instances to be solved. Every instance is simulated 100 times with a timeout of 30 minutes per instance. For each domain and planning formulation we record the number of dead ends, the average response time and the average reward for each instance when no dead end was reached. The average response time is the accumulated time to produce an action for a simulation run. For the Classical formalization solved by FAST DOWNWARD we denote the all-outcome determinization as *Eff*⋆ and the most-likely determinization by *eff*$_{\max}$ in tables.

The EARTHOBSERVATION domain does not have dead ends. The all-outcome determinization ran into memory limitations and thus was not applicable. Response times and rewards are given in Tab. 1. Since PROST is an online planner, its planning time per decision is a parameter and hence not illustrated in any of our Tables (we set it to one second per step). Classical and FOND planning are more dependent on the problem structure as they have to find a plan first. The most-likely determinization here suffers from the problem that the most likely result for taking an observation with bad weather is that no picture is taken. Thus as long as there is one patch with bad weather no proper plans are found. The rewards in Tab. 1 illustrate this clearly. MYND and PROST perform better than the most-likely determinization.

We see a different effect for the MOBILEMANIPULATION setting. Obviously, MYND never ends up in a dead-end (see Tab. 4), but the most-likely deter-

[1] https://code.google.com/p/rddlsim/

	Response Time [s]			Reward			
	Eff⋆	*eff* $_{\max}$	MYND	*Eff*⋆	*eff* $_{\max}$	MYND	PROST
1	0	0	0	-2	-2	-5	-5 ± 1
2	0	0	0	-4	-4	-12	-9 ± 1
3	1	1	0	-6	-6	-19 ± 2	-14 ± 1
4	1	1	2	-8	-8	-25 ± 1	-21 ± 2
5	2	1	5	-10	-10	-31 ± 2	-28 ± 2
6	2	1	10	-12	-12	-37 ± 1	-31 ± 2
7	5	2	78	-14	-14	-44 ± 2	-36 ± 2
8	16	2	n/a	-16	-16	n/a	-41 ± 3
9	60	3	n/a	-18	-18	n/a	-45 ± 2
10	94	3	n/a	-20	-20	n/a	-52 ± 3
11	97	5	n/a	-22	-22	n/a	-57 ± 3
12	99	7	n/a	-24	-24	n/a	-60 ± 3
13	100	8	n/a	-26	-26	n/a	-66 ± 3
14	102	10	n/a	-28	-28	n/a	-70 ± 3
15	103	13	n/a	-30	-30	n/a	-76 ± 3

Table 3. Average response time and rewards for the TRIANGLETIREWORLD domain, the average of 100 runs is shown. Time values for MYND include offline planning time.

minization is also well-suited to avoid this case. This is due to the fact that the most likely outcome of 'risky' actions always leads to a dead-end, so FAST DOWNWARD never considers this option. PROST performs comparably bad in the MOBILEMANIPULATION domain since its heuristic maximizes the reward in the next couple of steps rather than lead it to a goal state. Unlike the other domains, a large number of actions must be performed until there is a positive effect on the reward formula. Starting with instance five, it doesn't perform any meaningful actions. Response times in Tab. 2 show that the MYND planner is consistently faster besides problem three, while the rewards are similar for all planners but PROST.

For the TRIANGLETIREWORLD domain we observe that classical planning runs into dead-ends early on, while PROST solves most problems such that no or only a few runs end in a dead end state (see Tab. 4). The response times in Tab. 3 show the expected behavior on this domain for those runs not leading to a dead end. No FOND strategy could be found for problem instances eight and higher, while FAST DOWNWARD still produces optimistic plans and PROST still manages to find a safe path to the goal in most instances. FAST DOWNWARD is fastest finding optimal weak plans efficiently being extremely optimistic. The rewards in Tab. 3 show this explicitly. If a plan was produced the reward always was maximal. The FOND planner must be pessimistic and thus gains lower rewards and solves fewer instances. However, there are no dead ends for the solved problems. In this domain, the tradeoff between dead-end avoidance and planning time is most obvious.

	Eff⋆	*eff*$_{\max}$	myND	Prost
1	0	0	0	0
2	29	0	0	0
3	56	0	0	0
4	68	0	0	0
5	88	0	0	0
6	67	0	0	0
7	20	0	0	0
8	52	0	0	0
9	76	0	0	0
10	78	0	0	0

	Eff⋆	*eff*$_{\max}$	myND	Prost
1	15	15	0	0
2	41	41	0	0
3	62	62	0	0
4	79	79	0	0
5	86	86	0	1
6	90	90	0	2
7	95	95	0	0
8	96	96	n/a	4
9	97	97	n/a	0
10	98	98	n/a	20
11	99	99	n/a	9
12	99	99	n/a	9
13	99	99	n/a	2
14	99	99	n/a	2
15	99	99	n/a	10

Table 4. These tables show, how many deadends were reached in 100 iterations for the MobileManipulation (left) and TriangleTireworld (right) domains.

6 Conclusion

We evaluated three different planning approaches on distinctly different domains with probabilistic outcomes. Our results indicate that more expressive planning formulations are not necessarily slower for realistic problem sizes. Each planner showed different behavior dependent on the specific domain. It should be noted that classical planning performs faster, when most actions are deterministic. However, in the presented challenging settings a clear advantage based on computation times cannot be seen. If more expressive features are desired a richer planning formalism does not need to be prohibitively slow; nondeterministic or probabilistic planning formalisms can show better overall performance.

Acknowledgements

This work was supported by the German Aerospace Center (DLR) as part of the Kontiplan project (50 RA 1010), by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centers SFB/TR 14 AVACS and SFB/TR 8 Project R7, and the PACMAN project within the HYBRIS research group (NE 623/13-1).

References

1. Boutilier, C., Dearden, R., Goldszmidt, M.: Stochastic Dynamic Programming with Factored Representations. Artificial Intelligence (AIJ) 121(1–2), 49–107 (2000)

2. Culberson, J.C., Schaeffer, J.: Searching with pattern databases. In: Advances in Artificial Intelligence. pp. 402–416. LNCS, Springer-Verlag (1996)
3. Fu, J., Ng, V., Bastani, F.B., Yen, I.L.: Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In: Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). pp. 1949–1954 (2011)
4. Gaschler, A., Petrick, R.P.A., Kröger, T., Knoll, A., Khatib, O.: Robot task planning with contingencies for run-time sensing. In: ICRA Workshop on Combining Task and Motion Planning (2013)
5. Hansen, E.A., Zilberstein, S.: LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence 129(1–2), 35–62 (2001)
6. Haslum, P., Botea, A., Helmert, M., Bonet, B., Koenig, S.: Domain-independent construction of pattern database heuristics for cost-optimal planning. In: AAAI Conference on Artificial Intelligence (AAAI 2007). pp. 1007–1012 (2007)
7. Helmert, M.: The fast downward planning system. Journal of Artificial Intelligence Research (JAIR) 26, 191–246 (2006)
8. Kaelbling, L., Lozano-Perez, T.: Hierarchical task and motion planning in the now. In: IEEE Conference on Robotics and Automation (ICRA) (2011)
9. Kaelbling, L., Lozano-Perez, T.: Integrated task and motion planning in belief space. International Journal of Robotics Research (2013)
10. Keller, T., Eyerich, P.: A Polynomial All Outcomes Determinization for Probabilistic Planning. In: Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011). pp. 331–334. AAAI Press (June 2011)
11. Keller, T., Eyerich, P.: PROST: Probabilistic Planning Based on UCT. In: Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012). pp. 119–127. AAAI Press (2012)
12. Keller, T., Eyerich, P., Nebel, B.: Task planning for an autonomous service robot. In: Proceedings on the 33rd Annual German Conference on Artificial Intelligence (KI). pp. 358–365. Springer Verlag (2010)
13. Keller, T., Helmert, M.: Trial-based Heuristic Tree Search for Finite Horizon MDPs. In: Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS 2013). pp. 135–143. AAAI Press (2013)
14. Kuter, U., Nau, D.S., Reisner, E., Goldman, R.P.: Using classical planners to solve nondeterministic planning problems. In: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS). pp. 190–197 (2008)
15. Little, I., Thiébaux, S.: Probabilistic planning vs replanning. In: In ICAPS Workshop on IPC: Past, Present and Future (2007)
16. Mattmüller, R., Ortlieb, M., Helmert, M., Bercher, P.: Pattern database heuristics for fully observable nondeterministic planning. In: International Conference on Automated Planning and Scheduling (ICAPS). pp. 105–112 (2010)
17. Muise, C.J., McIlraith, S.A., Beck, J.C.: Improved non-deterministic planning by exploiting state relevance. In: Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012) (2012)
18. Nebel, B., Dornhege, C., Hertle, A.: How much does a household robot need to know in order to tidy up your home? In: AAAI Workshop on Intelligent Robotic Systems (2013)