Pattern-Database Heuristics for Partially Observable Nondeterministic Planning

Manuela Ortlieb and Robert Mattmüller

Research Group Foundations of AI, University of Freiburg, Germany
{ortlieb,mattmuel}@informatik.uni-freiburg.de

Abstract. Heuristic search is the dominant approach to classical planning. However, many realistic problems violate classical assumptions such as determinism of action outcomes or full observability. In this paper, we investigate how – and how successfully – a particular classical technique, namely informed search using an abstraction heuristic, can be transferred to nondeterministic planning under partial observability. Specifically, we explore pattern-database heuristics with automatically generated patterns in the context of informed progression search for strong cyclic planning under partial observability. To that end, we discuss projections and how belief states can be heuristically assessed either directly or by going back to the contained world states, and empirically evaluate the resulting heuristics internally and compared to a delete-relaxation and a blind approach. From our experiments we can conclude that in terms of coverage and guidance, it is preferable to represent both nondeterminism and partial observability in the abstraction (instead of relaxing them), and that the resulting abstraction heuristics significantly outperform both blind search and a delete-relaxation approach where nondeterminism and partial observability are also relaxed.

Keywords: AI planning, nondeterministic planning, partial observability, heuristic search, pattern databases

1 Introduction

Classical planning is a well-understood problem that has been successfully approached over the past decades. Both for satisficing and for optimal planning, there are algorithms in the recent literature that scale well beyond simple toy problems [18,11]. Although lately the focus of research in classical planning has shifted towards algorithmic enhancements and pruning techniques orthogonal to planning heuristics, accurate domain-independent heuristics were the main driving factor in the progress of classical planning for many years. However, not all planning tasks fit into the framework of classical planning. Often, action outcomes are nondeterministic and the environment is only partially observable. We would still like to capitalize on the advances made in classical planning when solving such problems. In previous work [1,2,16], we already handled nondeterminism, but only full observability. As part of our efforts to get closer to real-world problems with the approach developed before, in this work, we consider the problem of finding so called strong cyclic plans [6] for partially observable nondeterministic planning tasks, i.e., policies that are guaranteed to never lead into states where they are undefined, and always maintain the possibility to reach a goal state. There exist various approaches to finding strong cyclic plans (for fully observable problems, but in principle adaptable to partially observable problems), including a symbolic nested fixpoint algorithm [6,13], repeated invocations of a classical planner in the all-outcomes determinization of the given nondeterministic planning task until no unsolved leaf nodes remain in the generated subgraph of the transition system [14,8], and (informed) forward search in the nondeterministic transition system induced by the planning task guided by an appropriate heuristic function [4,16]. In this paper, we study the latter approach, more specifically, LAO* search [9]. Whereas a similar study has been performed by Bryce et al. [4] before, that study only considers delete-relaxation heuristics to guide the search. Here, we want to complement that study with an investigation of domain-independent *abstraction* heuristics, more specifically, pattern-database (PDB) heuristics [7]. When Bryce et al. [4] studied relaxation heuristics, they investigated how to evaluate a belief state by either (a) sampling world states from that belief state, evaluating them using a relaxation heuristic assuming full observability, and aggregating the estimates across samples, or (b) directly evaluating the belief state by extending the relaxed planning graph from the fully observable setting to a so-called labeled uncertainty graph (LUG) for the partially observable setting. In this work, we perform a similar comparison between a sampling-based approach and a more direct evaluation of belief states for pattern-database heuristics. Our main research question is whether there is a significant difference in how well approaches (a) and (b) perform empirically, and if so, which performs better.

2 Preliminaries

2.1 Nondeterministic Planning under Partial Observability

We formalize nondeterministic planning tasks under partial observability using a *finite-domain representation* for the state variables and separate *causative and sensing actions*, extending our previous formalization for fully observable nondeterministic planning tasks [16]. A *partially observable nondeterministic 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 such that its *scope* $\mathrm{scope}(s) = \{v \in \mathcal{V} \mid s(v) \neq \bot\}$ is $\mathcal{V}$. The set of all states s over $\mathcal{V}$ is denoted as $\mathcal{S}$, and the set of all *belief states* B over $\mathcal{V}$ is denoted as $\mathcal{B} = 2^{\mathcal{S}}$. 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 partial state (i.e., a compact encoding of a compactly encodable belief 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, and a belief state B is a goal belief state iff each state $s \in B$ is a goal state. $\mathcal{O}$ is a finite set of *actions* partitioned into *causative actions* $\mathcal{O}_\mathrm{c}$ and *sensing actions* $\mathcal{O}_\mathrm{s}$. Causative actions are of the form $a_\mathrm{c} = \langle \mathit{Pre}, \mathit{Eff} \rangle$, where the *precondition* Pre is a partial state, and the *effect* Eff is a finite set of partial states eff, the *nondeterministic outcomes* of a. The *application* of a nondeterministic outcome eff to a state s is the state $\mathit{app}(\mathit{eff}, s)$ that results from updating s with eff. The application of an effect Eff to 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. Sensing actions are of the form $a_\mathrm{s} = \langle \mathit{Pre}, \mathit{Obs} \rangle$, where the *precondition* Pre is a partial state, and the *observed variables* Obs are a subset of $\mathcal{V}$. An action is *applicable* in a state s iff its precondition is satisfied in s, and it is applicable in a belief state B if it is applicable in all $s \in B$. Actions are applied in belief states and result in *sets* of belief states. The application of an action in a belief state B is undefined if the action is inapplicable in B. Otherwise, the application of a causative action $a_\mathrm{c} = \langle \mathit{Pre}, \mathit{Eff} \rangle$ to B is the singleton set $\mathit{app}(a_\mathrm{c}, B) = \{\{\mathit{app}(\mathit{eff}, s) \mid \mathit{eff} \in \mathit{Eff}, s \in B\}\}$, and the application of a sensing action $a_\mathrm{s} = \langle \mathit{Pre}, \mathit{Obs} \rangle$ to B is the set of nonempty belief states that result from splitting B according to possible observations, i.e., $\mathit{app}(a_\mathrm{s}, B) = \{\{s \in B \mid s' \subseteq s\} \mid s' \text{ partial state with scope}(s') = \mathit{Obs}\} \setminus \{\emptyset\}$. All actions have unit cost. Partially observable nondeterministic planning tasks as defined above induce nondeterministic transition systems where the nodes are the (reachable) belief states and where there is an arc from a belief state B to a belief state B' labelled with an action a iff a is applicable in B and $B' \in \mathit{app}(a, B)$. Given a partially observable nondeterministic planning task, we seek a strong cyclic plan solving the task, i.e., a partial mapping π from belief states to applicable actions such that for all belief states B reachable from the initial belief state $B_0 = \{s \in \mathcal{S} \mid s \text{ satisfies } s_0\}$ following π, B is either a goal belief state, or π is defined for B and at least one goal belief state is reachable from B following π. Later, we will occasionally simplify a partially observable problem by *assuming full observability*. In that case, the induced transition system will be defined slightly differently: First, all nodes will be world states instead of belief states, second, sensing actions will be ignored (since sensing occurs implicitly), and third, applying a causative action $a_\mathrm{c} = \langle \mathit{Pre}, \mathit{Eff} \rangle$ to a node representing a state s will no longer lead to a unique successor node, but rather to one successor node for each successor state in $\{\mathit{app}(\mathit{eff}, s) \mid \mathit{eff} \in \mathit{Eff}\}$ (i.e., AND nodes in the transition system are caused by nondeterministic actions instead of splitting of belief states). Also, we will sometimes simplify partially observable nondeterministic problems by *determinizing* them. In that case, we replace each causative action $a_\mathrm{c} = \langle \mathit{Pre}, \mathit{Eff} \rangle$, $\mathit{Eff} = \{\mathit{eff}_1, \dots, \mathit{eff}_n\}$, by n causative actions $a_\mathrm{c}^i = \langle \mathit{Pre}, \{\mathit{eff}_i\} \rangle$ for $i = 1, \dots, n$. Together with a unique initial state, this essentially leads to a classical planning problem.

2.2 Pattern-Database Heuristics

In classical planning, pattern-database heuristics work by projecting the planning task to a set of variables $P \subseteq \mathcal{V}$, the pattern, solving the resulting simplified planning task optimally, storing the optimal goal distances of all abstract states in a pattern database, and eventually using these abstract distances as heuristic values during search [7]. In addition, one often uses more than one pattern and maximizes over non-additive patterns and adds heuristic values from provably additive patterns. The most accurate admissible heuristic obtainable from a set of patterns is the so-called canonical heuristic function [10], which we will also use in this work. Formally, the projection of a partially observable nondeterministic planning task $\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{O} \rangle$ to a pattern $P \subseteq \mathcal{V}$ is defined component- and element-wise: $\Pi|_P = \langle P, s_0|_P, s_\star|_P, \mathcal{O}|_P \rangle$, where $s|_P(v) = s(v)$ for all $v \in \mathrm{scope}(s) \cap P$, and $s|_P(v) = \bot$, otherwise; where $\mathcal{O}|_P = \{a|_P \mid a \in \mathcal{O}\}$, $a_\mathrm{c}|_P = \langle \mathit{Pre}|_P, \{\mathit{eff}|_P \mid \mathit{eff} \in \mathit{Eff}\} \rangle$ for each causative action $a_\mathrm{c} = \langle \mathit{Pre}, \mathit{Eff} \rangle$, and $a_\mathrm{s}|_P = \langle \mathit{Pre}|_P, \mathit{Obs} \cap P \rangle$ for each sensing action $a_\mathrm{s} = \langle \mathit{Pre}, \mathit{Obs} \rangle$. To ensure that projections preserve action applications and goal states, we require that for each pattern P to which we project, all variables in P are either observable by some sensing action, or do not occur in any precondition or goal condition, or their value is known initially and never becomes uncertain through any action application.

3 Simplifying a Partially Observable Nondeterministic Planning Task

The only problem simplification that occurs in pattern-database heuristics for classical planning is the projection to the pattern. Our partially observable nondeterministic problems differ from classical problems in two respects: nondeterministic actions and partial observability. Our main research question is to investigate the best way how to deal with these two aspects when computing pattern databases. Both nondeterminism and partial observability can easily be retained in the abstraction: If an action a leads from state s to one of two different states s'_1 and s'_2 in the original problem, and s'_1 and s'_2 can still be distinguished in the abstraction under consideration, then a will still be nondeterministic in the abstract problem, leading from the abstraction of s to either the abstraction of s'_1 or the abstraction of s'_2. Similarly, if two states s_1 and s_2 cannot be distinguished in the original problem and belong to some common belief state B, their abstract versions cannot be distinguished in the abstract problem and the abstraction of B will contain the abstractions of s_1 and of s_2. Thus, besides abstracting to a pattern, we have four possibilities how to further simplify the abstract problem, namely all combinations of determinizing or not determinizing the problem and assuming or not assuming full observability in the abstraction. The resulting abstract problem will fall into one of the four categories in the following table:

		determinization	
		yes	*no*
observability	*full*	**(A)** FO-Det PSPACE-complete [5]	**(B)** FO-NDet EXPTIME-complete [15]
	partial	**(C)** PO-Det EXPSPACE-complete [19]	**(D)** PO-NDet 2-EXPTIME-complete [19]

This suggests that the abstract problem will be easier to solve the more sources of complexity (partial observability, nondeterminism) we abstract away. On the other hand, we expect better-informed heuristics and better guidance the fewer of these sources we abstract away:

(A) *Full observability, determinization (FO-Det):* This leads to a classical abstract problem that we can solve with classical regression search as it is usually done when computing PDB heuristics for classical problems. Information about nondeterministic outcomes belonging to the same original nondeterministic action is lost. Therefore, we implicitly minimize over possible action outcomes and thus underestimate true worst case or expected case costs. The resulting optimistic goal distances are stored in PDBs for all patterns in the pattern collection under consideration. During LAO* search, when PDB values are retrieved, since the PDBs contain (projected) world states as keys, we cannot directly look up a heuristic value for the (projection of the) belief state B we want to evaluate. Rather, we have to consider the (projections of the) world states s contained in B individually. This poses two challenges: The practical challenge lies in the fact that B can contain exponentially many world states in the number of state variables, which leads to prohibitively many PDB lookups for a single heuristic evaluation of a belief state. We resolve this by experimenting with different numbers of world state samples from B (sampling 1, 5, 10, 15, or all states). The conceptual challenge is the question how to aggregate heuristic values for individual states $s \in B$. Summing costs corresponds to assuming that all $s \in B$ have to be solved independently without positive interactions of the individual plans for each, whereas maximizing corresponds to assuming maximal positive interaction, where an optimal plan for the most expensive state $s \in B$ happens to solve all other states in B along the way. We experimented with both possible aggregation rules.

(B) *Full observability, no determinization (FO-NDet):* In this case, we end up with an AND/OR graph (a nondeterministic transition system) in the abstraction with splitting over causative action outcomes instead of over sensing action outcomes. This leads to the question of which cost measure to use in the abstract transition system. If we used weak (*optimistic*) goal distances, this would be the same as the FO-Det case above. We cannot use strong (*pessimistic*) goal distances, since this would assign cost values of ∞ to belief states that actually admit a strong cyclic solution. Instead, we perform value iteration on the resulting abstract AND/OR graph to label the abstract states with *expected* costs, i.e., expected numbers of steps to the nearest goal state. Lacking information about probabilities of different action outcomes, we assign them uniform probabilities. Moreover, the remarks about sampling of world states from the belief state under consideration from the FO-Det case still apply.

(C) *Partial observability, determinization (PO-Det):* In this case, the only uncertainty in any reachable belief state comes from initial state uncertainty. We end up with an AND/OR graph with splitting over sensing action outcomes. If the initial state happens to be unique (fully observable), PO-Det amounts to FO-Det, and the complexity reduces from Expspace-complete to Pspace-complete.

(D) *Partial observability, no determinization (PO-NDet):* In this case, states in the abstract transition system – and therefore the keys in the PDBs – are still belief states (not world states). As in the other cases where AND nodes in the abstract transition systems are involved, we have to choose an aggregation rule for interior nodes (optimistic, pessimistic, or expected costs). Again, we use expected costs under the assumption that each successor of an AND node has the same weight. We leave the idea of weighing successor nodes (belief states) by cardinality for future work.

In the experiments below we compare three of these four approaches among each other and to (a) a delete-relaxation approach with additional determinization and assumption of full observability and (b) the blind heuristic.

4 Implementation Details

We implemented a tool in Java that computes strong cyclic plans for partially observable nondeterministic planning tasks using LAO* search [9] guided by FO-Det, FO-NDet and PO-NDet PDB heuristics.[1] In all cases, we use the canonical heuristic function induced by a pattern collection computed using Haslum et al.'s local search in the space of pattern collections [10]. In the case where we preserve partial observability in the abstraction (PO-NDet), we consider two different ways of computing a pattern collection: the one where we also assume partial observability during pattern search, and the one where we avoid searching for a suitable pattern collection in the *belief* space by assuming *full* observability during pattern search. After that local search terminates, we use the resulting pattern collection to create pattern databases under *partial* observability. Within LAO*, we use a nonstandard expansion strategy in the case when there is no unexpanded non-goal leaf node in the most promising partial solution graph: We alternate between expanding an unexpanded non-goal leaf node with *minimal h value* outside the most promising partial solution graph and expanding

[1] We disregard PO-Det for the following reasons: (a) The additional simplification over PO-NDet appears minor and in PO-Det we would still have to deal with an AND/OR graph (instead of simply an OR graph) in the abstraction, and (b) two of the three benchmarks domains we consider (Fr and Blocks, see below) have fully observable initial states, i.e., in these benchmarks PO-Det and FO-Det would collapse anyway.

an unexpanded non-goal leaf node outside the most promising partial solution graph which was *created earliest* among all such nodes. Moreover, our LAO* implementation uses maximization and discounting to aggregate cost estimates at interior nodes. Belief states and transitions between them are represented symbolically using Binary Decision Diagrams (BDDs) [3]. Sampling of world states from belief states represented as BDDs is done uniformly with replacement.

5 Experiments

We ran our planner on a compute server equipped with AMD Opteron 2.3 GHz CPUs. For each single planner run, we imposed a 4GB memory limit and a 30 minute time limit on the JRE. The time for the search for a pattern collection was limited to 10 minutes. When that limit was reached, the best pattern collection found so far was used. If the pattern collection search terminated in less than 10 minutes, the main LAO* search was allowed to use up the remainder of the original 30 minutes.

5.1 Benchmark Domains

We adapted the FirstResponders (Fr) and Blocksworld (Blocks) domains from the fully observable nondeterministic track of the International Planning Competition 2008 by requiring active sensing for certain state variables:

- Fr: The task is to plan a rescue operation where fires have to be extinguished and victims have to be treated on-scene or at a hospital. We required active sensing for victims' health statuses and for whether fires are still burning or already extinguished.
- Blocks: Unlike in the classical Blocks domain, where towers of blocks have to be reconfigured using deterministic block movement actions, in our formalization there are actions that can fail, like transferring a block from one tower to another. We require active sensing for block positions.

In addition, we experimented with a variant of the Canadian Traveler Problem:

- Ctp: The Canadian Traveler Problem [17] is originally a probabilistic planning problem which we transformed into a partially observable nondeterministic one. It consists of a road map where an agent has to travel from a start to a goal location. In the original formalism, each road is passable with a specific probability and driving roads has different costs. In our transformation, driving a road has unit costs and there are roads which are definitively passable, definitively not passable or for which it is unknown if they are passable. Sensing actions are used to determine if an incident road is passable or not.

5.2 Belief State Sampling

When we assume full observability in the abstractions, in order to evaluate a belief state B during LAO* search, we need to sample world states from B, evaluate them individually, and aggregate the resulting heuristic values into a heuristic value for B. Before comparing FO-Det, FO-NDet, and PO-NDet, we first want to find suitable parameters for the numbers of belief state samples (we experimented with 1, 5, 10, 15, all) and aggregation methods (we experimented with maximizing and adding) used in FO-Det and FO-NDet. The results are summarized in Table 1. For all sampling methods except for "all", sampling is with replacements. Sampling "all" considers each state from B exactly once. In this experiment, for each problem instance, we use the same pattern collection for all configurations of sample numbers and aggregation methods to improve comparability of the results. Therefore, preprocessing times (pattern collection computation times) are the same for all configurations and hence omitted from the table. In the Fr domain with summation, coverage and guidance tend to increase with the number of samples with the exception of sampling all states. With summation, it is not a good idea to sum over *all* world states, because this introduces an unjustified bias of the search towards low-cardinality belief states. With maximization, we get mixed guidance, time and coverage results for different numbers of samples, with a small advantage of maximizing over *all* world states. Overall, in Fr FO-NDet has a higher coverage than FO-Det. In the Blocks domain, it turns out that it is often cheaper to sample *all* world states (without replacement) than to use a fixed number of samples (with replacement), since the cardinalities of the encountered belief states are very small (typically less than 10). When sampling is used, guidance and search time tend to improve with the number of samples. That means that time spent for sampling is compensated by better guidance, and in terms of coverage, FO-Det slightly outperforms FO-NDet. Overall, in Blocks FO-Det tends to outperform FO-NDet. In the Ctp domain, it was not possible to enumerate all world states of the belief states encountered during search because of their exponential cardinality. There is no significant difference between FO-Det and FO-NDet or between maximizing and summing in the Ctp domain. In conclusion, except for a few Blocks instances, coverage is slightly higher with FO-NDet than with FO-Det, and in both cases, the sweet spot of the sample number seems to be around 10 or 15. Summing over samples appears a bit more promising than maximizing over them.

5.3 Pattern Selection

When we assume partial observability in the abstraction, we are faced with different ways of performing the search for suitable pattern collections. In Table 2, we report on an experiment for the PO-NDet case with the following three configurations: In configuration "steps 0", we perform no pattern collection search at all, but rather use a collection of singleton patterns with one pattern for each goal variable. In configuration "pop mip0.5", we assume partial observability also

Table 1. Coverage (cov) and guidance (number of node expansions, exp) and search times (time, in seconds) on commonly solved problems 30 (FO-Det) and 28 (FO-Ndet) in Fr, 10 in Blocks, 26 in Ctp) for different numbers of samples (1, 5, 10, 15, all) and different aggregation methods (maximizing and adding).

Domain		FO-Det						FO-NDet					
		max			sum			max			sum		
	n	cov	exp	time	cov	exp	time	cov	exp	time	cov	exp	time
Fr	1	*42*	*13835*	*995*	41	*13835*	1357	*40*	*11084*	1125	*40*	*11084*	*1077*
(75 tasks)	5	54	6161	291	*58*	*3644*	*156*	58	6599	855	*60*	*4868*	*206*
	10	56	12194	755	*62*	*2716*	*162*	55	11097	494	*64*	*3338*	*117*
	15	51	11267	579	*62*	*4481*	*320*	56	11420	631	*65*	*4998*	*341*
	all	*54*	*11085*	*395*	32	27048	1900	*59*	*9810*	*309*	31	12751	665
Blocks	1	*12*	*3573*	*24*	*12*	*3573*	46	*14*	*4024*	*49*	*14*	*4024*	76
(30 tasks)	5	*14*	2766	50	12	*2214*	*34*	*13*	*2647*	*52*	*13*	3261	89
	10	13	2509	*34*	*14*	*1863*	37	*12*	*1699*	*25*	*12*	3532	77
	15	*14*	1922	*31*	*14*	*1796*	33	12	*1271*	*25*	*13*	2495	60
	all	13	2392	22	*14*	*1618*	*16*	*14*	*2731*	61	12	3007	*49*
Ctp	1	*26*	*751*	*28*	*26*	*751*	31	*26*	*728*	*29*	*26*	*728*	32
(46 tasks)	5	*26*	494	*76*	*26*	*460*	79	*26*	507	*74*	*26*	*488*	86
	10	*26*	560	154	*26*	*428*	*143*	*26*	561	147	*26*	*391*	*121*
	15	*26*	518	196	*26*	*401*	*195*	*26*	523	202	*26*	*408*	*198*
	all	*0*	—	—	*0*	—	—	*0*	—	—	*0*	—	—

during pattern collection search and use a minimal improvement threshold [10] of 0.5 (i.e., we only perform a local search step in the pattern collection search if the fraction of samples for which the canonical heuristic value is improved is at least 0.5). Similarly, in configuration "fop mip0.5", we assume *full* observability during pattern collection search and use a minimal improvement threshold of 0.5 as well. From the data in Table 2, we conclude that it is typically preferable to search for better pattern collections than the trivial singleton pattern collections, and that assuming full observability during that search tends to improve total time because the preprocessing time is significantly decreased, whereas assuming partial observability generates better patterns at a higher preprocessing cost, but leads to faster (better informed) LAO* search.

5.4 Internal Comparison of FO-Det, FO-NDet, and PO-NDet

To determine the overall best PDB configuration, we compare the best configurations of FO-Det, FO-NDet, and PO-NDet side by side in Table 3. For FO-Det and FO-NDet, we use the configurations with summation over 15 belief state samples, and for all three configurations, we use a minimal improvement threshold of 0.5. We observe that the additional informedness of PO-NDet over the more simplistic FO-Det and FO-NDet configurations mostly translates into fewer node expansions as well as lower search and overall times. Although there is only a small resulting increase in coverage, altogether PO-NDet appears dominant.

Table 2. Coverage (cov) and guidance (number of node expansions, exp), search times (stm, in seconds), and total times (ttm, in seconds, including pattern collection search) on commonly solved problems (39 in Fr, 11 in Blocks, 23 in Ctp) for different configurations of the pattern collection search for PO-NDet.

	PO-NDet											
Domain	steps 0				pop mip0.5				fop mip0.5			
	cov	exp	stm	ttm	cov	exp	stm	ttm	cov	exp	stm	ttm
Fr	40	25278	3079	3111	70	5887	*218*	1058	*73*	*5819*	262	*588*
Blocks	*13*	6560	630	*644*	12	*5343*	*423*	673	12	6902	779	866
Ctp	*26*	526	9	*15*	23	*461*	*4*	862	*26*	480	5	314
OVERALL	79	32364	3718	3770	105	*11691*	*645*	2593	*111*	13201	1046	*1768*

Table 3. Coverage (cov) and guidance (number of node expansions, exp), search times (stm, in seconds), and total times (ttm, in seconds, including pattern collection search) on commonly solved problems (69 in Fr, 10 in Blocks, 26 in Ctp) for LAO* search with best FO-Det, FO-NDet, and PO-NDet configuration.

Domain	FO-Det sum15 mip0.5				FO-NDet sum15 mip0.5				PO-NDet fop mip0.5			
	cov	exp	stm	ttm	cov	exp	stm	ttm	cov	exp	stm	ttm
Fr	70	40159	9330	10320	72	28938	9140	11327	*73*	*26414*	*3851*	*6095*
Blocks	*14*	1796	33	85	13	2558	59	113	12	*1670*	*19*	*78*
Ctp	*26*	*607*	281	*849*	*26*	*607*	270	1004	*26*	630	*7*	923
OVERALL	110	42562	9644	11254	*111*	32103	9469	12444	*111*	*28714*	*3877*	*7096*

5.5 Comparison to Delete Relaxation and Baseline

In order to assess how well our best PDB configuration (PO-NDet fop mip0.5) does in comparison to an established technique (FF heuristic [12] under assumption of full observability and determinization) and a trivial baseline (blind heuristic), we provide a direct comparison in Table 4. We can conclude that PDBs outperform FF and blind heuristic in the Fr and Ctp domains in terms of coverage, guidance and runtime, whereas they perform slightly worse than FF in the Blocks domain. A comparison to a cleverer delete-relaxation approach like LUGs [4] that could shift the picture in favor of delete relaxation again, is left for future work. We remark that we do not expect a completely reversed picture with LUGs, since the PDB approach that is most comparable to the

FF approach under full observability and with determinization, namely FO-Det, still leads to a higher coverage than FF (102 vs. 78 solved problems).

Table 4. Coverage (cov) and guidance (number of node expansions, exp), search times (stm, in seconds), and total times (ttm, in seconds, including pattern collection search) on commonly solved problems (16 in Fr, 6 in Blocks, 13 in Ctp) for LAO* search with blind heuristic, FF heuristic under assumption of full observability and determinization, and the best PDB configuration.

Domain	blind			FF			PO-NDet fop mip0.5			
	cov	exp	stm=ttm	cov	exp	stm=ttm	cov	exp	stm	ttm
Fr	16	18716	1337	47	4381	239	*73*	*662*	12	*95*
Blocks	6	15937	488	*15*	*241*	*20*	12	276	2	37
Ctp	13	36124	2128	16	13714	735	*26*	*152*	1	*88*
OVERALL	35	70777	3954	78	18336	993	*111*	*1090*	16	*219*

6 Conclusion and Future Work

We have demonstrated that abstraction heuristics can successfully guide LAO* search for strong cyclic plans for partially observable nondeterministic planning problems towards goal belief states, and that the guidance is at least competitive with the guidance provided by a delete-relaxation heuristic. We argued experimentally that preserving partial observability and nondeterminism in the abstraction leads to more informative heuristics at the cost of more expensive preprocessing. From a global perspective, the better accuracy of such heuristics pays off with better overall planner performance.

Future work includes a comparison of our results to those of Bryce et al. [4] that also takes their labelled uncertainty graph (LUG) into account as an efficient data structure for the direct evaluation of belief states with a delete-relaxation approach without going back to their constituent world states. Moreover, we plan to investigate more realistic benchmark problems arising from robotic applications.

Acknowledgements. This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, see `http://www.avacs.org`).

References

1. Bercher, P., Mattmüller, R.: A planning graph heuristic for forward-chaining adversarial planning. In: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008). pp. 921–922 (2008)

2. Bercher, P., Mattmüller, R.: Solving non-deterministic planning problems with pattern database heuristics. In: Proceedings of the 32nd Annual Conference on Artificial Intelligence (KI 2009). pp. 57–64 (2009)
3. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers 35(8), 677–691 (1986)
4. Bryce, D., Kambhampati, S., Smith, D.E.: Planning graph heuristics for belief space search. Journal of Artificial Intelligence Research 26, 35–99 (2006)
5. Bylander, T.: The computational complexity of propositional strips planning. Artificial Intelligence 69(1–2), 165–204 (1994)
6. Cimatti, A., Pistore, M., Roveri, M., Traverso, P.: Weak, strong, and strong cyclic planning via symbolic model checking. Artificial Intelligence 147(1–2), 35–84 (2003)
7. Culberson, J.C., Schaeffer, J.: Searching with pattern databases. In: Advances in Artificial Intelligence. pp. 402–416. LNCS, Springer-Verlag (1996)
8. 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)
9. Hansen, E.A., Zilberstein, S.: LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence 129(1–2), 35–62 (2001)
10. Haslum, P., Botea, A., Helmert, M., Bonet, B., Koenig, S.: Domain-independent construction of pattern database heuristics for cost-optimal planning. In: Proc. 22nd AAAI Conference on Artificial Intelligence (AAAI 2007). pp. 1007–1012 (2007)
11. Helmert, M., Röger, G., Seipp, J., Karpas, E., Hoffmann, J., Keyder, E., Nissim, R., Richter, S., Westphal, M.: Fast downward stone soup (planner abstract). In: Seventh International Planning Competition (IPC 2011), Deterministic Part. pp. 38–45 (2011)
12. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14, 253–302 (2001)
13. Kissmann, P., Edelkamp, S.: Solving fully-observable non-deterministic planning problems via translation into a general game. In: Proc. 32nd German Annual Conference on Artificial Intelligence (KI 2009). LNCS, vol. 5803, pp. 1–8. Springer-Verlag (2009)
14. Kuter, U., Nau, D.S., Reisner, E., Goldman, R.P.: Using classical planners to solve nondeterministic planning problems. In: Proc. 18th International Conference on Automated Planning and Scheduling (ICAPS 2008). pp. 190–197 (2008)
15. Littman, M.L.: Probabilistic propositional planning: Representations and complexity. In: Proc. 14th National Conference on Artificial Intelligence (AAAI 1997). pp. 748–754. MIT Press (1997)
16. Mattmüller, R., Ortlieb, M., Helmert, M., Bercher, P.: Pattern database heuristics for fully observable nondeterministic planning. In: Proc. 20th International Conference on Automated Planning and Scheduling (ICAPS 2010). pp. 105–112 (2010)
17. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theoretical Computer Science 84, 127–150 (1991)
18. Richter, S., Westphal, M., Helmert, M.: Lama 2008 and 2011 (planner abstract). In: Seventh International Planning Competition (IPC 2011), Deterministic Part. pp. 50–54 (2011)
19. Rintanen, J.: Complexity of planning with partial observability. In: Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS 2004). pp. 345–354 (2004)