PROST-DD - Utilizing Symbolic Classical Planning in THTS

Florian Geißer and **David Speck**
University of Freiburg, Germany
{geisserf, speckd}@informatik.uni-freiburg.de

Abstract

We describe PROST-DD, our submission to the International Probabilistic Planning Competition 2018. Like its predecessor PROST, which already participated with success at the previous IPPC, PROST-DD is based on the trial-based heuristic tree search framework and applies the $UCT^{\star}$ algorithm. The novelty of our submission is the heuristic used to initialize newly encountered decision nodes. We apply an iterative symbolic backward planning approach based on the determinized task. Similarly to the SPUDD approach and recent work in symbolic planning with state-dependent action costs, we encode costs and reachability of states in a single decision diagram. During initialization, these diagrams are then used to query a state for its estimated expected reward. One benefit of this heuristic is that we can optionally interweave the standard heuristic of PROST, the IDS heuristic.

Introduction

The 6th edition of the International Probabilistic Planning competition initially consisted of three different tracks: the discrete MDP track, the continous MDP track, and the discrete SSP track. In this paper, we will discuss our submission to the discrete MDP track, which consists of a novel heuristic implemented into the PROST planner (Keller and Eyerich 2012), the winner of the previous IPPC. The goal of the discrete SSP track is to come up with a policy for a factored Markov decision process (MDP) with fixed initial state and fixed horizon. The reward is state-dependent and there are no dead-ends. As in the previous IPPC, the language to model the planning tasks is the Relational Dynamic Influenc Diagram (RDDL) language (Sanner 2010), and planners are evaluated by executing 75 runs per instance and comparing the average accumulated reward.

The PROST planner is based on the trial-based heuristic tree search framework (THTS) (Keller and Helmert 2013) which allows to mix several ingredients to compose an anytime optimal algorithm for finite-horizon MDPs. One of these ingredients is the state-value initialization (or: heuristic) used to give an initial estimate for previously unknown states. Our submission exchanges the iterative deepening search (IDS) heuristic, the original heuristic implemented in PROST, with a heuristic based on backward symbolic search (BSS) on the determinized task. On the one hand, this approach can be compared to SPUDD (Hoey et al. 1999), a stochastic planning approach using decision diagrams. On the other hand, it can be compared to recent work on symbolic planning for tasks with state-dependent action costs (Speck, Geißer, and Mattmüller 2018).

Before we describe our heuristic, we quickly introduce the THTS framework and the setup of the PROST planner in the previous IPPC. The next section then explains the BSS heuristic, before we finally sketch how we can interweave BSS and IDS, to come up with a stronger heuristic for more challenging tasks.

Trial-based Heuristic Tree Search

The trial-based heuristic tree search (THTS) framework (Keller and Helmert 2013) allows to model several well-known probabilistic search algorithms in one common framework. It is based on the following ingredients: *heuristic function*, *backup function*, *action selection*, *outcome selection*, *trial length*, and *recommendation function*. Independent of the specific ingredients, the general tree search algorithm maintains a tree of alternating decision and chance nodes, where a decision node contains a state s and a state-value estimate based on previous trials. A chance node contains a state s, an action a and a Q-value estimate, which estimates the expected value of action a applied in state s. The algorithm performs so-called trials, until it either computed the optimal state-value estimate of the state in the root node of the tree, or until it is out of time. A THTS trial consists of different phases: the *selection phase* traverses the tree according to action and outcome selection until a previously unvisited decision node is encountered. Then, in the *expansion phase*, this selected node is expanded, where for each action a child node is added to the tree and initialized with a heuristic value according to the heuristic function. The trial length parameter decides if the selection phase starts again, or if the *backup phase* is initiated. In this phase, the visited nodes are updated in reversed order according to the backup function. A trial finishes when the backup function is called on the root node. In the case that the algorithm is out of time, the *recommendation function* recommends which action to take, based on the values of the child nodes. For more information on the THTS algorithm we recommend the original THTS paper (Keller and Helmert 2013), as well as the PhD thesis of T. Keller (Keller 2015) which introduces recommendation functions and contains a thorough theoretical

and empirical evaluation of a multitude of algorithms realized within this framework.

Our submission is based on the PROST configuration of the IPPC 2014, together with a novel heuristic function. Before we describe this heuristic, we quickly mention the other ingredients used in our configuration. The action selection function is based on the well-known UCB1 formula (Auer, Cesa-Bianchi, and Fischer 2002) which has a focus on balancing exploration versus exploitation. The outcome selection is based on Monte-Carlo sampling and samples outcomes according to their probability, with the additional requirement that the outcome was not already marked as solved by the backup function. This backup function is a combination of Monte-Carlo backups and Full Bellman backups, and weights outcomes proportionally to their probability. It allows for missing (i.e. non-explicated) outcomes and also for labeling nodes as solved, where a node is solved if its optimal value estimation is known. These ingredients are the same used in the IPPC 2014. For the recommendation function, we apply the *most played arm* recommendation (Bubeck, Munos, and Stoltz 2009), which recommends one of the actions that have been selected most often in the root node (uniformly at random). This recommendation function was shown (Keller 2015) to be superior in combination with the other ingredients.

Backward Symbolic Search Heuristic (BSS)

The *Backward Symbolic Search Heuristic (BSS)* exploits the efficiency of symbolic search and the compactness of symbolic data structures in form of decision diagrams. More precisely, we use Algebraic Decision Diagrams as the underlying symbolic data structure. *Algebraic Decision Diagrams* (Bahar et al. 1997) represent algebraic functions of the form $f : \mathcal{S} \to \mathbb{R} \cup \{-\infty, \infty\}$. Formally, an ADD is a directed acyclic graph with a single root node and multiple terminal nodes. Internal nodes correspond to binary variables, and each node has two successors. The *low edge* represents that the current variable is false, while the *high edge* represents that the current variable is true. Evaluation of a function then corresponds to the traversal of the ADD according to the assignment of the variables.

The main idea of BSS is to determinize a given MDP while representing the MDP as decision diagrams with a subsequent backward exploration of the state space. Recently, Speck, Geißer, and Mattmüller (2018) showed how symbolic search can be applied to deterministic planning tasks with state-dependent action costs. Similar to their work, we perform a symbolic backward search on the determinized MDP which corresponds to a classical planning task with state-dependent action costs. This backward search results in multiple ADDs, where each ADD represents states associated with the maximum reward that can be achieved from the corresponding states. More precisely, BSS can be divided into three parts. First, a given MDP is determinized (all-outcome or most-likely). Second, a backward breadth-first search is performed, with the number of backward planning steps equal to the horizon. In each backward planning step we obtain an ADD which maps reachable states to rewards (*symbolic layers*). Finally, during the actual search the precomputed rewards (stored in decision diagrams) are used to evaluate state actions pairs, i.e. Q-values. In the following, we will explain each step in more detail.

Let a be an action of a given MDP. Action a has an empty precondition and effects $p(x' := \neg x) = 1, p(y' := 1) = 0.6$ and $p(y' := 0) = 0.4$. In other words, action a always negates the value of x and sets y to 1 (0) with probability of 0.6 (0.4). Finally, the reward function of action a is defined as $R(s, a) = 2 + 5 \cdot s(y)$, where $s(y)$ is the value of y in state s. In the initial step, action a is determinized (here: most-likely determinization) and represented as a transition relation in form of an ADD mapping state pairs consisting of predecessors S and successors S' to 1 (true) or 0 (false). Figure 1 depicts the ADD which represents action a as transition relation after applying the most-likely determinization, i.e. only outcomes with a probability of 0.5 are considered [1]. Finally, the reward function is added to the transition relation of a. If we transform the reward to a negative value, we obtain a transition relation representing costs which is analogous to the formalization of Speck, Geißer, and Mattmüller (2018). This transformation of an action is applied to each action which results in a determinized planning task with state-dependent action costs.

The symbolic backward search starts with all states associated with zero costs as shown on the left side of Figure 2. We perform h backward steps where h is equal to the horizon. Each backward step creates a symbolic Layer L_i, which stores for each state the maximal reward which can be achieved in the remaining i steps. In Figure 2, after one backward step (L_1) we can obtain a reward of 2 or 7 by applying an action. Note that there can be states where no action can be applied which is represented by a reward of $-\infty$. Finally, we initialize the value of a state s with action a as follows: let i be the number of remaining steps and let $S'_{s,a}$ be the set of possible successor states of applying action a in state s, i.e. $S'_{s,a} = \{s' | p(s'|s, a) > 0\}$. The initial Q-value of a state action pair is defined as

$$Q^{\text{init}}(s, a) = \frac{\sum_{s' \in S'_{s,a}} L_{i-1}(s')}{|S'_{s,a}|}.$$

In other words, we take the average of the precomputed rewards of all successor states of predecessor state s with respect to action a. In the following, we present how we can combine this heuristic with the usual forward search heuristic applied by the PROST planner.

Combining explicit forward and symbolic backward search heuristics

One property of the symbolic backward search heuristic is that it explores the whole deterministic task and collects all states and rewards reachable from the end of the horizon. While this is certainly easier for the determinized task it still is a hard problem and as a result we might only have results for parts of the horizon. In this case, we can make use of

[1] This may contrast with some other notions of most-likely, namely that most-likely usually means only accepting the most-likely outcome.

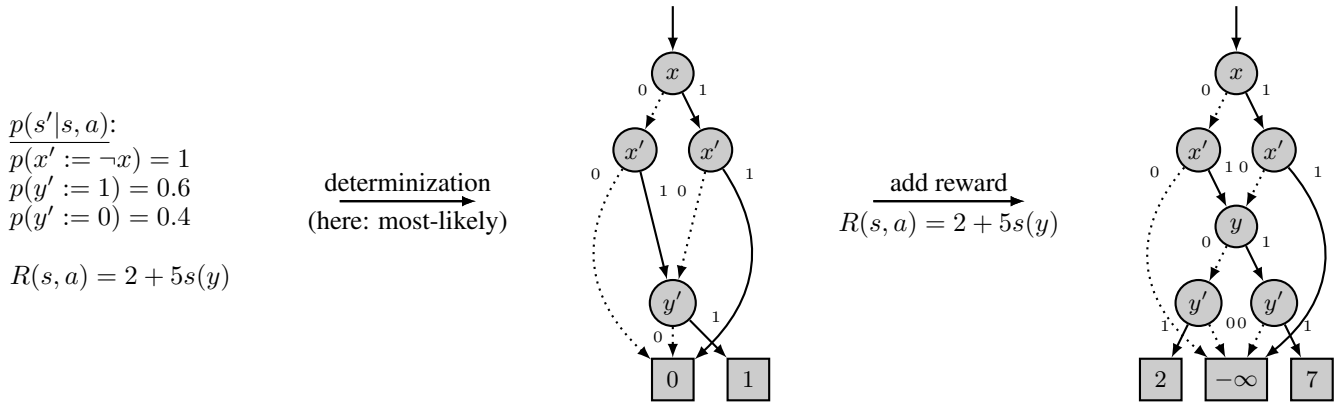

Figure 1: Transformations of action a. Action a has an empty precondition and effects $p(x' := \neg x) = 1$, $p(y' := 1) = 0.6$ and $p(y' := 0) = 0.4$. Functions related to action a are depicted as ADDs. In the middle the determinized transition relation and on the right the final transition relation with rewards.

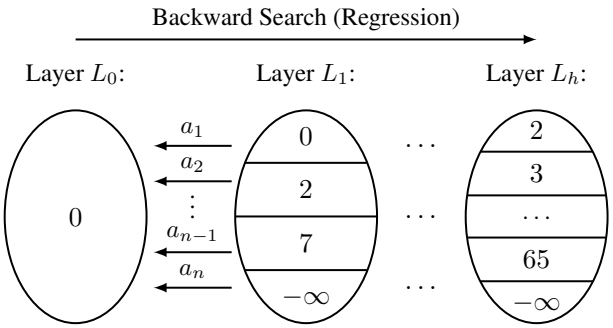

Figure 2: Visualization of symbolic backward search starting with all states associated with zero costs. At each backward step, all states leading to the previous state s are stored in an ADD called Layer L_i and mapped to the maximum reward which can be achieved from s in the remaining i steps.

the original heuristic implemented in the PROST planner, which is based on iterative deepening search (IDS).

The IDS heuristic also performs a determinization of the probabilistic task, and then conducts a depth-first search to compute the maximal reward reachable in the *next* d steps. The value of d is computed before search starts and usually depends on the complexity of the domain and its transition functions. The IDS heuristic can therefore be seen as an explicit forward search algorithm which complements our symbolic backward search approach. While our heuristic computes maximal rewards from the end of the horizon up to some step i, IDS computes the maximal reward reachable in the next d steps. Therefore, whenever we are not able to compute all layers, we combine both heuristics by querying the backward search value of the last layer and add the value estimated by IDS (with depth corresponding to the maximal layer). This can also be seen as a portfolio approach: if we are able to build all layers in the symbolic search we rely solely on our heuristic. If the task is so complex that we are not even able to build a single layer we only rely on the IDS heuristic (and have a setup very similar to the previous IPPC configuration). In all other cases we interweave both heuristics in order to generate an initial state value estimate which is better than when we would solely rely on a single heuristic.

Competition Analysis

Now that the competition is over we present a brief analysis of some of the results (Keller 2018b). The two versions of our planner differ only in the heuristics. Version 1 computes the BSS heuristic based on a most-likely determinization, while version 2 is based on an all-outcome determinization. We are especially interested in a comparison to the baseline planners (the PROST planner configurations of 2011 and 2014), since our planner mostly differs in the heuristic.[2] This year's IPPC consisted of eight domains with 20 instances each, resulting in a total of 160 instances. It turns out that grounding was a major challenge this year. For example, the hardest Academic Advising instance has more than 11 *billion* grounded actions, due to the combinatorial blowup (5 out of 269 actions are applicable concurrently). In total, our planner was unable to ground 31 instances. This certainly warrants investing more research effort into the grounding of concurrent actions. Regarding the remaining instances, the PROST-DD planner crashed during search in 16 instances, mainly in the Earth observation domain due to a bug. Table 1 shows the average rewards of our planner (bug fixed) compared to the PROST versions of the IPPCs 2011 and 2014 on the Earth Observation domain. The differences in performance are minor.

[2]We additionally fixed some bugs of the PROST planner which were mostly concerned with not exceeding the memory limit (the baseline planners only used 2GB RAM). Unfortunately, we introduced a bug which led to a crash in most of the Earth Observation domain instances; otherwise our planner's score would have exceeded the baseline score.

	PROST-DD		PROST	
ID	most-likely (v1)	all-outcome (v2)	2011	2014
1	-8.84	-8.92	-15.95	**-8.49**
2	-486.75	-483.91	**-478.11**	-484.93
3	-704.89	-709.97	**-697.33**	-714.44
4	**-1574.65**	-1600.60	-1591.32	-1616.31
5	-649.13	**-644.63**	**-643.91**	-672.15
6	-239.01	**-236.27**	-237.88	-248.41
7	**-39.40**	**-39.65**	-40.79	-42.04
8	-431.67	**-429.09**	-455.09	-455.47
9	-1355.19	-1355.19	-1291.91	**-1278.41**
10	-3203.95	-3222.25	**-3167.59**	**-3163.08**
11	**-820.28**	**-819.43**	-833.20	-825.21
12	-1668.00	-1693.24	**-1657.35**	-1665.31
13	-1919.39	-1922.29	**-1841.96**	**-1839.17**
14	-10099.60	-10103.80	-9957.04	**-9827.79**
15	-2645.88	-2627.17	**-2588.33**	-2775.01
16	-353.41	**-351.64**	-380.27	-368.68
17	-1875.48	-1875.48	-1791.63	**-1736.52**
18	-3186.06	-3186.60	-2989.44	**-2843.33**
19	-5170.25	-5114.09	-4954.21	**-4825.60**
20	-12702.90	-12665.50	-12731.80	**-12622.90**

Table 1: Average reward of the PROST-DD planner (version 1 and version 2) compared to the PROST versions of the IPPCs 2011 and 2014 on the Earth Observation domain.

The key question remains: has the BSS heuristic paid off? To answer this question, we analyze the number of tasks for which it was possible to compute at least one layer, which meant that the BSS heuristic could also be used during the search. Unfortunately, it turns out that it was only possible to compute at least one layer in 23 instances. This is certainly due to the fact that the domains of this years IPPC were more challenging compared to previous problems of former IPPCs. The BSS heuristic was mostly succesful in the domains Academic Advising and Push Your Luck. In both domains, the performance of both configurations was evenly good, and superior to other planners. The heuristic computation took around 10% of the search time. In the Manufacturer and Cooperative Recon domains the heuristic was unable to generate a single layer and thus consumed time in the precomputation phase without providing useful information. This might be a reason for the low performance. However, once computed the BSS heuristic is informative and helpful. This certainly shows that the heuristic has potential, but needs to be more efficient, especially when faced with large and difficult problems. We already have some ideas for such improvements. Interestingly, we also outperformed other planners in Wildlife Preserve, even though we use the same heuristic as the baseline planner in this case. This may be due to the additional memory we use, but also due to some modifications to the grounding of actions which differs slightly from the baseline.

In summary, the heuristic presented here has paid off in some domains and has affected the planner's performance in others due to loss of time. PROST-DD proved to be a competitive planner and the BSS heuristic showed promising results.

Our planner submission is available in the official IPPC repositories on Bitbucket (Keller 2018a). We fixed the bug which led to crashes in the Earth Observation domain in the branch ipc2018-disc-mdp. The original competition version is available on the branch ipc2018-disc-mdp-competition.

Acknowledgments

David Speck was supported by the German National Science Foundation (DFG) research unit FOR 1513 on Hybrid Reasoning for Intelligent Systems (`http://www.hybrid-reasoning.org`).

References

[Auer, Cesa-Bianchi, and Fischer 2002] Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finitetime Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47:235–256.

[Bahar et al. 1997] Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1997. Algebraic decision diagrams and their applications. In *Proceedings of the International Conference on Computer Aided Design (ICCAD 1993)*, volume 10, 171–206.

[Bubeck, Munos, and Stoltz 2009] Bubeck, S.; Munos, R.; and Stoltz, G. 2009. Pure Exploration in Multiarmed Bandits Problems. In *Algorithmic Learning Theory, 20th International Conference (ALT 2009)*, 23–37.

[Hoey et al. 1999] Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 279–288.

[Keller and Eyerich 2012] Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 119–127.

[Keller and Helmert 2013] Keller, T., and Helmert, M. 2013. Trial-based Heuristic Tree Search for Finite Horizon MDPs. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 135–143.

[Keller 2015] Keller, T. 2015. *Anytime Optimal MDP Planning with Trial-based Heuristic Tree Search*. Ph.D. Dissertation, University of Freiburg.

[Keller 2018a] Keller, T. 2018a. Bitbucket repository of the ippc 2018 planners. `https://bitbucket.org/account/user/ipc2018-probabilistic/projects/EN`. [Online; accessed 08-October-2018].

[Keller 2018b] Keller, T. 2018b. Presentation slides of the ippc 2018. `https://ipc2018-probabilistic.bitbucket.io/results/presentation.pdf`. [Online; accessed 08-October-2018].

[Sanner 2010] Sanner, S. 2010. Relational Dynamic Influence Diagram Language (RDDL): Language Description.

[Speck, Geißer, and Mattmüller 2018] Speck, D.; Geißer, F.; and Mattmüller, R. 2018. Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 250–258.