Probabilistic Backward and Forward Reasoning in Stochastic Relational Worlds

Tobias Lang LANG@CS.TU-BERLIN.DE
Marc Toussaint MTOUSSAI@CS.TU-BERLIN.DE
TU Berlin, Machine Learning and Robotics Group, Franklinstraße 28/29, 10587 Berlin, Germany

Abstract

Inference in graphical models has emerged as a promising technique for planning. A recent approach to decision-theoretic planning in relational domains uses forward inference in dynamic Bayesian networks compiled from learned probabilistic relational rules. Inspired by work in non-relational domains with small state spaces, we derive a backpropagation method for such nets in relational domains starting from a goal state mixture distribution. We combine this with forward reasoning in a bidirectional two-filter approach. We perform experiments in a complex 3D simulated desktop environment with an articulated manipulator and realistic physics. Empirical results show that bidirectional probabilistic reasoning can lead to more efficient and accurate planning in comparison to pure forward reasoning.

1. Introduction

Intelligent agents have to accomplish two tasks to act autonomously in complex worlds: First, they need to learn how their environment works. Second, they have to use the acquired knowledge efficiently to decide which actions to take to achieve the goals and maximize the expected rewards. Research on the first task has led to probabilistic relational knowledge representations: these can deal with stochastic actions, cope with noise and generalize over objects and situations. In spite of considerable progress over recent years, designing efficient algorithms for reward maximization in such learned world models is still a major concern.

Action decision-making (Boutilier et al., 1999) in stochastic relational domains is often cast in a relational reinforcement learning framework (Dzeroski et al., 2001) which investigates the use of compact relational representations for state and action spaces. We are interested in model-based approaches enabling agents with changing goals to plan for the goal at hand by internal simulation. Regarding the learning of such world models, Pasula et al. (2007) have proposed noisy indeterministic deictic (NID) rules. These rules extend probabilistic STRIPS operators (Kushmerick et al., 1995) by two special constructs essential for learning: a noise outcome to avoid modeling of rare and overly complex outcomes and deictic references which refer to objects other than the action arguments and allow for more compact rule-sets. Actions are modeled by different rules whose preconditions do not have to be mutually exclusive; in a specific situation, we require a unique rule with satisfied preconditions and unique deictic references to avoid contradicting predictions.

To plan with these learnt rules, one can formalize the problem as a relational Markov decision process (MDP). The field of Symbolic Dynamic Programming (Boutilier et al., 2001) investigates methods to compute policies over complete state and action spaces working in the lifted abstract representation without grounding or referring to particular object instances (Kersting et al., 2004; Sanner & Boutilier, 2009). As an alternative, several methods exist for reasoning in the grounded domain, which makes it straightforward to account for the noise outcome and the uniqueness requirement of NID rules. The forward planners SST (Kearns et al., 2002) and UCT (Kocsis & Szepesvari, 2006) sample outcomes to cope with the stochasticity of grounded actions. PRADA (Lang & Toussaint, 2009) converts NID rules into a grounded Dynamic Bayesian net (DBN) representation and propagates action effects forward by means of a factored frontier filter (Murphy & Weiss, 2001). Backward planning in grounded domains, in contrast, has largely focussed on non-probabilistic domains where action outcomes are not probabilistically determined. Classical approaches use regression techniques to map action operators and

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

state variables to formulas describing the conditions under which a variable becomes true (Rintanen, 2008).

Our idea in this paper is to derive a backpropagation method to enable bidirectional reasoning in probabilistic relational domains. This may drastically prune the search space of action sequences: instead of search spaces which are exponential in the length d of the complete plans, we only have to consider search spaces which are exponential in $\frac{d}{2}$. Furthermore, backward reasoning is particularly useful in problems where the number of possible actions close to goal states is small in comparison to the start state. Consider for instance the goal to build a tower from objects which are initially scattered over a table. Unfortunately, previous approaches such as the forward-backward algorithm in Hidden Markov models (HMMs) (Rabiner, 1989) and the planning by inference paradigm (Toussaint & Storkey, 2006) in non-relational MDPs work in limited small state spaces and are not applicable in DBNs for grounded relational domains, where exact inference is infeasible. The factored frontier inference of PRADA cannot be used directly for backpropagation, either. A core challenge is how to condition the state distribution at the last time-step on receiving a high reward when using approximate inference. This problem arises in particular for complex abstract reward dependencies such as partial goal descriptions.

We make three contributions to overcome these problems: *(i)* We show how to use NID rules to learn a probabilistic backward model. *(ii)* We model arbitrary (partial) goal descriptions with a mixture state distribution and derive a probabilistic backward reasoning procedure. *(iii)* We introduce a two-filter (Solo, 1982) inference method to use bidirectional reasoning for planning in stochastic relational domains. The remainder of this paper is organized as follows: First, we review the theoretical background, namely NID rules and the PRADA planning algorithm. In Section 3, we present our two-filter using backward reasoning with NID rules. In Section 4, we introduce the bidirectional planning approach. Then, we show our empirical results before we conclude.

2. Background

2.1. State and Action Representation

A relational domain is represented by a relational logic language $\mathcal{L}$: the set of logical predicates $\mathcal{P}$ and the set of logical functions $\mathcal{F}$ contain the relationships and properties that can hold for domain objects. We distinguish between primitive and *derived* concepts. The latter are defined in terms of formulas over primitive or other derived concepts. The set of logical predicates $\mathcal{A}$ comprises the possible actions in the domain.

A concrete instantiation of a relational domain is made up of a finite set of objects $\mathcal{O}$. If the arguments of a predicate or function are all concrete, i.e. taken from $\mathcal{O}$, we call it *grounded*. A concrete world state s is fully described by all grounded predicates and functions. Concrete actions a are described by positive grounded predicates from $\mathcal{A}$. The arguments of predicates and functions can also be abstract logical variables which can represent any object. If a predicate or function has only abstract arguments, we call it *abstract*. We will speak of grounding an abstract formula ψ if we apply a substitution σ that maps all of the variables appearing in ψ to objects in $\mathcal{O}$.

2.2. Noisy Indeterministic Deictic Rules

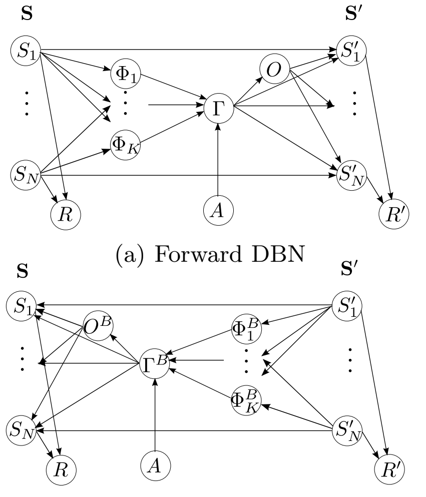
Noisy Indeterministic Deictic (NID) (Pasula et al., 2007) rules are a model of the transition dynamics in relational domains. Table 1 shows an exemplary NID rule for a desktop environment. A NID rule r is given as follows

$$a_r(\mathcal{X}) : \ \Phi_r(\mathcal{X}) \quad \rightarrow \quad \begin{cases} p_{r,1} & : \ \Omega_{r,1}(\mathcal{X}) \\ & \vdots \\ p_{r,m_r} & : \ \Omega_{r,m_r}(\mathcal{X}) \\ p_{r,0} & : \ \Omega_{r,0} \end{cases} , \quad (1)$$

where $\mathcal{X}$ is a set of logic variables in the rule (which represent a (sub-)set of abstract objects). The rule r consists of preconditions, namely that action a_r is applied on $\mathcal{X}$ and that the state context Φ_r is fulfilled, and m_r+1 different outcomes with associated probabilities $p_{r,i} > 0$, $\sum_{i=0} p_{r,i} = 1$. Each outcome $\Omega_{r,i}(\mathcal{X})$ describes which predicates and functions change when the rule is applied. The context $\Phi_r(\mathcal{X})$ and outcomes $\Omega_{r,i}(\mathcal{X})$ are conjunctions of literals constructed from the predicates in $\mathcal{P}$ as well as equality statements comparing functions from $\mathcal{F}$ to constant values. In contrast to the outcomes, the context may contain derived predicates and functions, enabling complex and abstract situation descriptions. The so-called *noise outcome* $\Omega_{r,0}$ subsumes all possible action outcomes which are not explicitly specified by one of the other

Table 1. Example NID rule for a desktop world, which models dropping an object X over an object Y in a situation where Y is below an object Z (deictic referencing). With high probability, X will land on Z, but might also fall on the table. With a small probability something unpredictable happens.

$$\mathit{dropabove}(X,Y) : \quad \mathit{inhand}(X),\ \mathit{on}(Z,Y),\ \mathit{table}(T)$$
$$\rightarrow \begin{cases} 0.6 & : \ \mathit{on}(X,Z),\ \neg \mathit{inhand}(X) \\ 0.3 & : \ \mathit{on}(X,T),\ \neg \mathit{inhand}(X) \\ 0.1 & : \ \text{noise} \end{cases}$$



(b) Backward DBN

Figure 1. PRADA converts NID rules into a forward DBN (a) to predict the effects of action sequences. For backward reasoning, we use a second DBN (b) with the same state, action and reward variables, but different rule variables (Φ_i^B, Γ^B and O^B) according to the backward rules.

$\Omega_{r,i}$. The arguments of the action $a(\mathcal{X}_a)$ may be a true subset $\mathcal{X}_a \subset \mathcal{X}$ of the variables $\mathcal{X}$ of the rule. The remaining variables are called *deictic references* $\mathcal{D} = \mathcal{X} \setminus \mathcal{X}_a$ and denote objects relative to the agent or action being performed.

Let σ denote a substitution that maps variables to constant objects, $\sigma : \mathcal{X} \rightarrow \mathcal{O}$. Applying σ to an abstract rule $r(\mathcal{X})$ yields a *grounded rule* $r(\sigma(\mathcal{X}))$. We say a grounded rule r *covers* a state s and a grounded action a if $s \models \Phi_r$ and $a = a_r$. If r is the *unique covering rule* for a in s, it defines $P(s'|a, s)$, the probability of a successor state s' if action a is performed in state s, according to r's outcomes and their probabilities. If there is no unique covering rule, the effects of a are explained as noise by a default rule. NID rules can be learned from experience triples (s, a, s') using a batch algorithm that trades off the likelihood of these triples with the complexity of the learned rule-set.

2.3. Planning using Probabilistic Inference

The PRADA algorithm (Lang & Toussaint, 2009) plans *forward* in stochastic relational domains using a set of NID rules that predict state transitions. It copes efficiently with stochastic action outcomes by means of probabilistic inference. It converts NID rules into a structured DBN representation (Fig. 1(a)). To clarify notation, we denote random variables by upper case letters (e.g. S), their values by the corresponding lower case letters (e.g. $s \in dom(S)$), variable tuples by bold upper case letters (e.g. $\mathbf{S} = (S_1, S_2, S_3)$) and value tuples by bold lower case letters (e.g. $\mathbf{s} = (s_1, s_2, s_3)$). We also use column notation (e.g. $\mathbf{s}_{3:5} = (s_3, s_4, s_5)$).

PRADA represents each grounded predicate or function at time t as a random variable S_i^t with value s_i^t. Random variables A for actions, Γ for rules, Φ for preconditions and O for outcomes model the transitions between states $\mathbf{S}^t$ and $\mathbf{S}^{t+1}$ at subsequent timesteps. PRADA uses a factored frontier (Murphy & Weiss, 2001) approximating the posterior $P(\mathbf{s}^t \mid \mathbf{a}^{0:t-1})$ to efficiently propagate the effects of action sequences forward. The reward gained in a state is represented by a binary variable R which can be used to express arbitrary reward expectations. For planning, i.e., to find an action sequence $\mathbf{a}$ with large $P(R \mid \mathbf{a})$, PRADA samples action-sequences in an informed way, taking predicted state distributions into account: action a is sampled at time-step t with a probability proportional to the probability that a has a unique covering rule – in which case one can make meaningful predictions.

3. Two-Filter Smoothing using Backward NID Rules

We propose to adapt PRADA for backward reasoning using NID rules. Then, we can exploit the knowledge about the goal for planning.

3.1. Backward Messages for a Two-Filter

Existing approaches for combining probabilistic backward and forward reasoning calculate smoothed state posteriors conditioned on a sequence of observed variables, e.g. the forward-backward algorithm in HMMs (Rabiner, 1989) or Expectation-Maximization used for planning by inference in small state spaces (Toussaint & Storkey, 2006). Here, we want to calculate posteriors $P(s^t, R^T \mid \mathbf{a}^{0:T-1})$ where T is the last time-step which we can use to evaluate an action-sequence by

$$P(R^T \mid \mathbf{a}^{0:T-1}) = \sum_{\mathbf{s}^t} P(\mathbf{s}^t, R^T \mid \mathbf{a}^{0:T-1}) \,. \qquad (2)$$

These posteriors can be calculated by means of forward messages $\alpha_{\mathbf{a}^{0:t-1}}(\mathbf{s}^t) := P(\mathbf{s}^t \mid \mathbf{a}^{0:t-1})$ and backward messages $\beta_{\mathbf{a}^{t:T-1}}(\mathbf{s}^t) := P(R^T \mid \mathbf{s}^t, \mathbf{a}^{t:T-1})$ such that $P(R^T, \mathbf{s}^t \mid \mathbf{a}^{0:T-1}) = \alpha_{\mathbf{a}^{0:t-1}}(\mathbf{s}^t) \cdot \beta_{\mathbf{a}^{t:T-1}}(\mathbf{s}^t)$.

It is intractable to calculate these messages exactly in relational domains due to the immense state spaces. PRADA calculates the forward messages α approximately using a factored frontier filter. Unfortunately, PRADA's specific factored frontier equations only work for forward reasoning and cannot be applied for calculating the likelihood backward messages β. We might use rejection sampling in PRADA's forward DBN, but this is highly inefficient. It is in general unclear how to calculate the β even approximately in a tractable way in all but the smallest state spaces. Therefore, as an alternative we propose a filtering approach for backward reasoning. We use PRADA in reversed order, providing us with messages $\hat{\beta}_{\mathbf{a}^{t:T-1}}(\mathbf{s}^t) := P(\mathbf{s}^t \,|\, \mathbf{a}^{t:T-1}, R^T)$. This requires a set of backward NID rules from which we can build a backward DBN (Fig. 1(b)) and apply PRADA's factored frontier inference (see next section).

A *two-filter* (Solo, 1982; Briers et al., 2009) uses the resulting messages $\hat{\beta}$ to approximate the likelihood messages β,

$$\begin{aligned}
\beta_{\mathbf{a}^{t:T-1}}(\mathbf{s}^t) &= P(R^T \,|\, \mathbf{s}^t, \mathbf{a}^{t:T-1}) \qquad (3)\\
&= \frac{P(\mathbf{s}^t \,|\, R^T, \mathbf{a}^{t:T-1}) P(R^T \,|\, \mathbf{a}^{t:T-1})}{P(\mathbf{s}^t \,|\, \mathbf{a}^{t:T-1})}\\
&= \frac{P(\mathbf{s}^t \,|\, R^T, \mathbf{a}^{t:T-1}) P(R^T \,|\, \mathbf{a}^{t:T-1})}{P(\mathbf{s}^t)}\\
&\approx \frac{P(\mathbf{s}^t \,|\, R^T, \mathbf{a}^{t:T-1}) P(R^T)}{P(\mathbf{s})}\\
&\propto P(\mathbf{s}^t \,|\, R^T, \mathbf{a}^{t:T-1}) = \hat{\beta}_{\mathbf{a}^{t:T-1}}(\mathbf{s}^t) \; .
\end{aligned}$$

The approximations of this two-filter are due to the intractable state distributions $P(\mathbf{s}^t)$ at specific time-steps t, also required to account for the dependencies of R^T on $\mathbf{a}^{t:T-1}$. Since in planning we are interested in ranking different action sequences $\mathbf{a}$, we can drop the likewise intractable reward marginal $P(R^T)$, as we can drop $P(\mathbf{s})$ assuming a uniform state prior.

3.2. Backward Rules

To use PRADA for backward filtering, we require a set of backward NID rules to define a distribution $P(s \,|\, s', a)$ over precessor state s if an action a was applied before the current state s'. These rules take exactly the same form as in the forward case, only the semantics are changed. Given a forward model, one might try to invert the according rules. How to account for the special characteristics of NID rules such as uniqueness and noise outcomes in this case is unclear, however. As our forward models are learned and thus in any event approximations of the true underlying dynamics, we propose to learn the backward rules directly from data as well. This has the advantage that we can use the same algorithm which we already use for learning the forward rules. We only have to provide the experience triples in reversed order (s', a, s).

Depending on the domain, state transitions may be easier to model in one direction than the other. This may affect the deictic references in NID rules which may be unique only in one direction. Consider for instance the unary predicate *pickup*(X) used by Pasula et al. (Fig. 2). A forward rule could use a deictic reference Y to describe where X was taken from which is required to conclude $\neg on(X,Y)$ for the successor state. When reasoning backward, looking only at the successor state s', it is impossible to determine Y. This can be solved by increasing the arity of the action predicate so that less deictic references need to be resolved. For this reason, in our experiments we will use binary action predicates $takefrom(X,Y)$ and $dropabove(X,Y)$ instead of $grab(X)$ and $puton(Y)$ – while still allowing for deictic referencing to third or fourth objects.

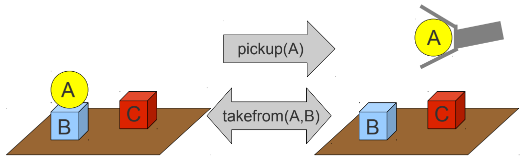


Figure 2. Using a unary action predicate *pickup*(A), it is impossible to deduce from the successor state (via a deictic reference) whether A was taken from B or C. This information is captured explicitely in the binary action predicate $takefrom(A,B)$. Due to its action sampling strategy, extending the arity of actions does not influence PRADA's planning efficiency.

At first glance, one might suspect that extending the action predicate arity increases the planning complexity due to the increased action space. This is resolved, however, when using a policy that only considers actions with unique rules. As we regularize our rule learning procedure, the learned rules model typical state transitions. Thus, a planner using these rules takes actions only in frequently observed contexts into account, effectively pruning large parts of the action space in a given situation. Furthermore, determining all unique covering rules has the same computational cost, independently of whether Y is used as a second action argument or as a deictic reference.

4. Backward-Forward Planning

We use a two-filter to plan in stochastic relational domains. Given a backward model $B \equiv P(s \,|\, s', a)$ in form of NID rules for a state s, an action a and successor state s', we apply PRADA first to reason backward to estimate a distribution of states backward-reachable from goal states. Then, we use a second set of NID rules specifying a forward model $A \equiv P(s' \,|\, s, a)$ to reason forward from the initial state to find action sequences leading to states close to a goal state.

4.1. Goal State Distributions

For probabilistic backward reasoning, we require a state distribution at the last time-step T. We are interested in states achieving a high reward, namely

$$\hat{\beta}^T(\mathbf{s}) := P(\mathbf{s} \,|\, R^T) \; . \qquad (4)$$

In contrast to previous work on planning by inference, we cannot calculate $\hat{\beta}^T(\mathbf{s})$ exactly in relational domains due to the large state spaces. Thus, we approximate it with a factored frontier $\hat{\beta}^T(\mathbf{s}) \approx \prod_i \hat{\beta}^T(s_i)$.

If the goal fully specifies the final state, setting the marginals $\hat{\beta}^T(s_i)$ to their deterministic values is straightforward. If the goal is defined in terms of a partial state description in form of a conjunction ς of primitive predicates and functions, only some state attributes $\mathbf{s}_\varsigma \subset \mathbf{s}$ have deterministic values. The situation becomes more difficult to deal with if the goal is specified in terms of a derived predicate, corresponding to formulas over primitive predicates, such as existentially quantified goals. Consider for instance the goal to stack the cubes $\{a, b, c\}$ in any order. In this case, the clearly dissimilar states $\mathbf{s}_1 = \{on(a, b), on(b, c)\}$ and $\mathbf{s}_2 = \{on(c, b), on(b, a)\}$ yield the same reward. If we approximate the final state belief by marginals, we lose the crucial correlations among the variables. This is a general problem in backward planning and arises likewise in non-probabilistic and propositional domains. A common strategy there is to pick arbitrary grounded forms of the goal, e.g. choosing $\mathbf{s}_1$ in the example above. This has the pitfall that some goal groundings may not be reachable or more costly to reach from the given state. To avoid these problems and achieve a closer approximation of the goal state distribution $\hat{\beta}^T$, we approximate it by means of a mixture model with individual components $\hat{\beta}_c^T$,

$$\hat{\beta}^T(\mathbf{s}^T) \approx \frac{1}{C}\sum_{c=1}^{C} \hat{\beta}_c^T(\mathbf{s}^T). \tag{5}$$

The components c are built from conjunctions ς_c over grounded primitive predicates and functions which partially describe world states achieving high reward. For instance, ς_c might define the tower in $\mathbf{s}_2$ above. We choose these formulas ς_c without taking the initial state $\mathbf{s}_0$ or knowledge about actions in terms of rules into account – to separate this clearly from planning. Concerning unspecified properties of the final state, we use a prior $P^F(\mathbf{s})$ and define the component $\hat{\beta}_c^T$ as

$$\hat{\beta}_c^T(\mathbf{s}) \propto \delta_{\mathbf{s},\varsigma_c} P^F(\mathbf{s}), \quad \delta_{\mathbf{s},\varsigma_c} = \begin{cases} 1 & \text{if } \ \mathbf{s} \Vdash \varsigma_c \\ 0 & \text{otherwise} \end{cases} . \tag{6}$$

We choose $P^F(\mathbf{s})$ such that states close to the initial state $\mathbf{s}^0$ are highly probable. This is inspired by traditional A.I. backward planning: there, state variables not in ς_c are left unspecified until either they need to be set as required by the preconditions of a rule during backward search or until the initial state is achieved in which case all unspecified variables in the final state implicitely get set to their values in the initial state. This assumption is also advantageous for the factored frontier as the repeated multiplication of small probabilities (such as uninformed 0.5 for binary variables) may lead to very small rule context probabilities, decreasing the probabilities of unique rules.

4.2. Backward Messages

For each component $\hat{\beta}_c^T$ of the mixture model approximation of $\hat{\beta}^T$ given in Eq. (5), we sample N backward action sequences $\mathbf{b}_{ci} = (b_{ci}^{T-1}, \dots, b_{ci}^{T-D_\leftarrow})$ of horizon $D_\leftarrow$ using PRADA and the backward model B, where we set $\hat{\beta}_c^T$ as the initial state distribution. One such sample $\mathbf{b}_{ci}$ results in the distribution $\hat{\beta}^t_{\mathbf{b}_{ci}}(\mathbf{s}^t) = P(\mathbf{s}^t \mid \mathbf{b}_{ci}, \varsigma_c, R^T)$. We do not want to evaluate a forward action sequence $\mathbf{a}$ with each backward action sequence $\mathbf{b}_{ci}$ individually. Hence, we approximate state posteriors $\hat{\beta}^t(\mathbf{s}^t) = P(\mathbf{s}^t \mid R^T)$ generalizing over concrete action sequences as

$$\hat{\beta}^t(\mathbf{s}^t) \approx \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N}\sum_{i=1}^{N}\hat{\beta}^t_{\mathbf{b}_{ci}}(\mathbf{s}^t) \ . \tag{7}$$

The resulting $\hat{\beta}$ define the probability of states according to backward reasoning from the goal state mixture distribution $\hat{\beta}^T$ using PRADA's action sampling strategy. They quantify which states are actually backward reachable from goal states.

4.3. Evaluating Forward Sequences

Having calculated the backward messages $\hat{\beta}^t(\mathbf{s}^t)$, we sample forward action sequences $\mathbf{a}^{0:t-1}$ using the forward model A and PRADA yielding messages $\alpha_{\mathbf{a}^{0:t-1}}(\mathbf{s}^t) = P(\mathbf{s}^t \mid \mathbf{a}^{0:t-1})$. For each $\mathbf{a}$ we are interested in its suitability to achieve a goal state at time T with $t \leq T$. We use the two-filter of Sec. 3.1 with the backward state distribution of Sec. 4.2 to calculate

$$P(R^T \mid \mathbf{a}^{0:t-1}) = \sum_{\mathbf{s}^t} P(\mathbf{s}^t \mid \mathbf{a}^{0:t-1}) P(R^T \mid \mathbf{s}^t) \tag{8}$$

$$\underset{\sim}{\propto} \sum_{\mathbf{s}^t} \alpha_{\mathbf{a}^{0:t-1}}(\mathbf{s}^t)\hat{\beta}^{T-t}(\mathbf{s}^t) \ . \tag{9}$$

Representing the messages by means of factored frontiers $\alpha(\mathbf{s}) = \prod_i \alpha(s_i)$ and $\hat{\beta}(\mathbf{s}) = \prod_i \hat{\beta}(s_i)$ (dropping indices for clarity), where i ranges over the individual state attributes, we calculate this sum over messages products as

$$\sum_{\mathbf{s}} \alpha(\mathbf{s})\hat{\beta}(\mathbf{s}) = \sum_{\mathbf{s}}\prod_i \alpha(s_i)\hat{\beta}(s_i) \tag{10}$$

$$= \prod_i \sum_{s_i} \alpha(s_i)\hat{\beta}(s_i) \tag{11}$$

$$= \prod_i \sum_{s_i} \alpha(s_i)\frac{1}{C}\sum_{c=1}^{C}\hat{\beta}_c(s_i) \tag{12}$$

$$= \prod_i \frac{1}{C}\sum_{s_i} \alpha(s_i)\sum_{c=1}^{C}\hat{\beta}_c(s_i) \ . \tag{13}$$

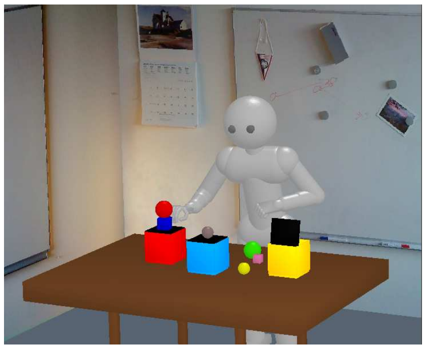


Figure 3. In our experiments, a robot has to master different tasks in a 3D simulated complex desktop environment involving cubes, balls and boxes of different sizes.

4.4. Action Selection

For a set $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_M\}$ of forward action sequence samples of length $D_{\rightarrow}$, we determine the best action sequence $\mathbf{a}^*$ defined as

$$\mathbf{a}^* = \operatorname*{argmax}_{\mathbf{a}\in\mathbf{A}} P(R \mid \mathbf{a}) \tag{14}$$

$$= \operatorname*{argmax}_{\mathbf{a}\in\mathbf{A}} \max_{0<t\le D_{\rightarrow}} \sum_T P(T) P(R^T \mid \mathbf{a}^{0:t-1}), \tag{15}$$

$$\approx \operatorname*{argmax}_{\mathbf{a}\in\mathbf{A}} \max_{0<t\le D_{\rightarrow}} \sum_T \gamma^T \sum_{\mathbf{s}_t} \alpha_{\mathbf{a}^{0:t-1}}(\mathbf{s}^t) \hat{\beta}^{T-t}(\mathbf{s}^t) \ , \tag{16}$$

where we take different horizons T to achieve a goal state into account, discounting them with $P(T) = \gamma^T$ with $0 < \gamma < 1$ to favour smaller horizons.

5. Evaluation

We compare our backward-forward planning approach PRADA$\leftrightarrows$ to the purely forward approaches Upper Confidence Bounds on Trees (UCT) (Kocsis & Szepesvari, 2006), PRADA$\rightarrow$ and A-PRADA$\rightarrow$ (Lang & Toussaint, 2009). To estimate action values, UCT grows lookahead trees based on sampling successor states using an adaptive policy to cut suboptimal parts of the tree early. If rules contain probabilistically dominant outcomes, UCT can be viewed as almost determinizing the corresponding state transitions. A-PRADA$\rightarrow$ examines the best plan found by PRADA$\rightarrow$ and decides by simulation whether omitting some actions will increase its utility.

Our test domain is a simulated complex desktop environment where a robot manipulates cubes, balls and boxes scattered on a table (Fig. 3). We use a 3D rigid-body dynamics simulator (ODE) that enables a realistic behavior of the objects. For instance, piles of objects may topple over or objects may even fall off the table (in which case they become out of reach for the robot). Depending on their type, objects show different characteristics. For example, it is almost impossible to successfully put an object on top of a ball, and building piles with small objects is more difficult. The robot can grab objects, try to put them on top of other objects, in a box or on the table. Boxes have a lid; special actions may open or close the lid; taking an object out of a box or putting it into it is possible only when the box is opened. The actions of the robot are affected by noise so that resulting object piles are not straight-aligned. We assume full observability of triples (s, a, s') that specify how the world changed when an action was executed in a certain state. We represent the data with predicates $\mathit{cube}(X)$, $\mathit{ball}(X)$, $\mathit{box}(X)$, $\mathit{table}(X)$, $\mathit{on}(X,Y)$, $\mathit{contains}(X,Y)$, $\mathit{out}(X)$, $\mathit{inhand}(X)$, $\mathit{upright}(X)$, $\mathit{closed}(X)$, $\mathit{clear}(X) \equiv \forall Y.\neg \mathit{on}(Y,X)$, $\mathit{inhandNil}() \equiv \neg\exists X.\mathit{inhand}(X)$ and function $\mathit{size}(X)$ for state descriptions and $\mathit{dropabove}(X,Y)$, $\mathit{takefrom}(X,Y)$, $\mathit{openBox}(X)$, $\mathit{closeBox}(X)$ and $\mathit{doNothing}()$ for actions. All compared methods consider only actions with unique rules, so that for the reasons discussed in Sec. 3.2 also the purely forward methods are not affected by extending the action predicate arities. If there are o objects and f different object sizes and colors, the state space is huge with $f^{2o}2^{2o^2+7o}$ different states (not excluding states one would classify as impossible given some intuition about real world physics). This shows the potential of backward reasoning to prune large parts of the search space, leading to faster, more accurate planning. Existing planning by inference approaches on non-factored representations, however, are clearly not applicable in such spaces.

We employ the rule learning algorithm of Pasula et al. with the same parameter settings to learn forward and backward action models in form of fully abstract NID rules from training sets of 500 experience triples each. Training data to learn rules are generated in a world of two boxes, six cubes and four balls of two different sizes by performing random actions with a slight bias to build high piles. Learning a backward model is more difficult as deictic references can often not be uniquely resolved (cf. Sec. 3.2). We suspect this to be a domain-specific characteristic rather than a general directional bias (see Massey (1999) for a discussion of the metaphysics of directionality in planning). The resulting backward models are compact and cover the standard situations that arise in the tasks (such as lifting a clear object). We learn one backward model (9 abstract rules) and three different forward models (12-14 abstract rules) from independent training data.

We perform three experiments. In each experiment, we investigate different worlds with varying numbers of objects. Thus, we transfer the knowledge gained in the training world to different, but similar worlds

Table 2. Clearance problem. Obj. denotes the object number (cubes/balls and table) and *Reward* the discounted total reward, which is 0 for performing no actions. PRADA⇆ is the proposed bidirectional planning approach.

Obj.	Planner	Reward	Trial time (s)
6+1	UCT	32.13±0.41	31.85±1.47
6+1	PRADA→	53.76±0.45	**7.64±1.34**
6+1	A-PRADA→	53.11±0.35	17.11±1.34
6+1	PRADA⇆	**54.10±0.48**	14.48±1.41
8+1	UCT	15.05±0.70	166.05±6.36
8+1	PRADA→	31.33±0.94	**65.90±1.00**
8+1	A-PRADA→	32.23±0.97	76.97±1.47
8+1	PRADA⇆	**33.12±1.09**	**65.91±2.01**
10+1	UCT	32.15±0.97	1148.81±29.83
10+1	PRADA→	97.25±1.96	426.84±16.41
10+1	A-PRADA→	88.40±1.75	444.46±11.10
10+1	PRADA⇆	**111.34±1.96**	**399.04±6.04**

by using abstract NID rules. For each object number we create five start situations with different objects. Per rule-set combination and start situation, we perform three independent runs with different random seeds. For evaluation, we compute the mean planning times and performances over the fixed (but randomly generated) set of 45 test scenarios (3 *learned* forward rule-sets, 1 *learned* backward rule-set, 5 situations, 3 random seeds). In all experiments, we use deliberately overestimated planning horizons D as these can't be known apriori. For PRADA⇆, we set $D_{\leftarrow}$ and $D_{\rightarrow}$ each equal to $\frac{1}{2}D$.

Clearance The goal in our first experiment is to clear up the objects which are scattered over the desktop. An object is defined to be cleared if it is piled with all objects of the same color. In our experiments, 2-4 objects have the same color with at most 1 ball (to enable successful piling). The starting situations contain piles, but only with objects of different colors. We let the robot perform 20 actions in worlds of 6 objects (in addition to the table), 30 for 8 and 40 for 10 objects. We emphasize that we did not use any world knowledge to set the goal state mixture distribution for PRADA⇆. In particular, the mixtures also contain clearly impossible situations (as could be deduced from the rules), for examples piles where balls are the lowest objects. Table 2 shows our results. UCT performs worst even when admitted very long planning times. We controlled the other approaches to have about the same planning time. In this rather easy planning problem not requiring long horizons, the additional computational overhead of combing forward and backward reasoning starts to pay off in worlds of 10 objects. Then, the planning problem has achieved a certain level of complexity (a very large state space) and PRADA⇆ performs significantly better than the pure forward approaches.

Reverse Tower The goal is to reverse a tower of c cubes. This is a difficult planning task requiring a long planning horizon. (Depending on the available rule-sets, the minimum horizon may be less than $2c$ as the robot may predict that a cube on top of the grabbed cube may land on the table.) We set a limit of 50 actions on each trial. Table 3 presents our results. UCT cannot be used for this task requiring over an hour for a trial. To achieve about the same performance as PRADA⇆, the forward PRADA approaches need 25-70% more planning time. Backward reasoning prunes the search-space and hence speeds up planning.

Table 3. Reverse tower problem. Suc. is the success rate and *Actions* the number of used actions in case of success. PRADA⇆ is the proposed bidirectional planning approach.

Obj.	Planner	Suc.	Trial time (s)	Actions
5+1	UCT	0.0	> 1h	–
5+1	PRADA→	0.91	16.38±1.74	11.85±1.21
5+1	A-PRADA→	0.89	18.12±1.88	12.43±1.27
5+1	PRADA⇆	**0.93**	**10.69±0.47**	**10.12±0.47**
6+1	PRADA→	0.80	24.27±1.27	**12.06±0.67**
6+1	A-PRADA→	**0.89**	27.59±2.28	12.62±0.93
6+1	PRADA⇆	0.83	**18.20±0.80**	12.26±0.47
7+1	PRADA→	**0.62**	129.83±8.44	14.75±0.80
7+1	A-PRADA→	0.60	123.20±5.70	**13.70±0.60**
7+1	PRADA⇆	0.58	**99.91±5.23**	14.77±0.87

Box Tower The goal is to build a specific tower of cubes and balls on one of three available boxes, no matter which one. All boxes are closed in the beginning. One of them contains the object which shall be on top of the goal tower. All the other objects are scattered on the table. This is a difficult planning problem as the robot may erroneously start building the desired tower just on the filled box before taking out the required object. As above, no specific world knowledge is used to construct the goal state mixture of PRADA⇆ which may also contain components where the desired tower is built on the filled box, increasing planning difficulty. The minimum number of required actions is $1 + 2 \cdot o$ where o is the number of objects (besides the box) in the target tower. We set a limit of 50 actions on each trial. Table 4 presents our results. As above, UCT is not competitive. PRADA⇆ always has the highest success rate – while at the same time requiring the smallest planning times. Backward reasoning is particularly useful in this scenario as the number of possible actions is comparatively small in goal states in comparison to start states. This is also reflected in the smaller number of executed actions to achieve a goal state in worlds with many objects.

6. Conclusions

We have introduced an approach for bidirectional planning in stochastic relational domains based on probabilistic two-filter inference which combines forward and backward reasoning. Our empirical results show that by exploiting the knowledge about goal states, we can significantly increase both planning accuracy

Table 4. Box tower problem. Obj. denotes the number of objects (cubes/balls, boxes and table), *Suc.* the success rate and *Actions* the number of used actions in case of success. PRADA⇆ is the proposed bidirectional planning approach.

Obj.	Planner	Suc.	Trial time (s)	Actions
3+3+1	UCT	0.29	331.94±2.13	**7.62±0.27**
3+3+1	PRADA→	0.84	12.64±0.20	11.53±1.14
3+3+1	A-PRADA→	0.82	11.89±0.17	10.51±0.94
3+3+1	PRADA⇆	**0.91**	**10.09±0.22**	14.66±2.14
4+3+1	UCT	0.0	> 1h	–
4+3+1	PRADA→	0.67	29.33±0.51	17.50±1.94
4+3+1	A-PRADA→	0.67	31.52±0.39	14.90±1.21
4+3+1	PRADA⇆	**0.73**	**22.98±0.46**	**12.36±1.68**
5+3+1	PRADA→	0.40	72.09±1.39	25.22±2.35
5+3+1	A-PRADA→	0.38	64.58±1.68	21.88±2.75
5+3+1	PRADA⇆	**0.51**	**61.03±1.06**	**17.35±1.55**

and efficiency. We plan in the fully grounded domain and thus in case of many objects, we have to combine it with methods reducing state and space complexity in relational domains, see e.g. Gardiol & Kaelbling (2007). Finding appropriate mixture models to approximate goal state beliefs to account for partial goal descriptions is a major topic of future research. Also, in a more traditional planning sense one might investigate using the backward messages as proposal distribution for biasing the forward messages for search. Furthermore, we want to investigate the relationships to lifted backward reasoning as is done in Symbolic Dynamic Programming (Boutilier et al., 2001).

Acknowledgments

We thank the anonymous reviewers for their careful comments. This work was supported by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3.

References

Boutilier, Craig, Dean, Thomas, and Hanks, Steve. Decision-theoretic planning: Structural assumptions and computational leverage. *Artificial Intelligence Research*, 11:1–94, 1999.

Boutilier, Craig, Reiter, Ray, and Price, Bob. Symbolic dynamic programming for first-order MDPs. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pp. 690–700. Morgan Kaufmann, 2001.

Briers, M., Doucet, A., and Maskell, S. Smoothing algorithms for state-space models. *To appear in Annals of the Institute of Statistical Mathematics*, 2009.

Dzeroski, S., de Raedt, L., and Driessens, K. Relational reinforcement learning. *Machine Learning*, 43:7–52, 2001.

Gardiol, Natalia H. and Kaelbling, Leslie Pack. Action-space partitioning for planning. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, pp. 980–986, 2007.

Kearns, Michael J., Mansour, Yishay, and Ng, Andrew Y. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.

Kersting, Kristian, van Otterlo, Martijn, and de Raedt, Luc. Bellman goes relational. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 465–472, 2004.

Kocsis, Levente and Szepesvari, Csaba. Bandit based monte-carlo planning. In *Proc. of the European Conf. on Machine Learning (ECML)*, 2006.

Kushmerick, N., Hanks, S., and Weld, D. An algorithm for probabilistic planning. *Artificial Intelligence*, 78 (1-2):239–286, 1995.

Lang, Tobias and Toussaint, Marc. Approximate inference for planning in stochastic relational worlds. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 585–592, 2009.

Massey, Bart. *Directions In Planning: Understanding The Flow Of Time In Planning.* PhD thesis, University of Oregon, 1999.

Murphy, Kevin P. and Weiss, Yair. The factored frontier algorithm for approximate inference in DBNs. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pp. 378–385, 2001.

Pasula, Hanna M., Zettlemoyer, Luke S., and Kaelbling, Leslie Pack. Learning symbolic models of stochastic domains. *Artificial Intelligence Research*, 29:309–352, 2007.

Rabiner, Lawrence R. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 31, pp. 257–286, 1989.

Rintanen, Jussi. Regression for classical and nondeterministic planning. In *Proc. of the European Conf. on Artificial Intelligence (ECAI)*, pp. 568–572, 2008.

Sanner, Scott and Boutilier, Craig. Practical solution techniques for first-order MDPs. *Artificial Intelligence*, 173(5-6):748–788, 2009.

Solo, V. Smoothing estimation of stochastic processes: Two-filter formulas. *IEEE Transactions on Automatic Control*, 27(2):473–476, 1982.

Toussaint, Marc and Storkey, Amos. Probabilistic inference for solving discrete and continuous state Markov decision processes. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 945–952, 2006.