Sample-based abstraction for hybrid relational MDPs

Davide Nitti davide.nitti@cs.kuleuven.be
Department of Computer Science, KU Leuven, Belgium

Vaishak Belle vaishak.belle@cs.kuleuven.be
Department of Computer Science, KU Leuven, Belgium

Tinne De Laet tinne.delaet@kuleuven.be
Faculty of Engineering Science, KU Leuven, Belgium

Luc De Raedt luc.deraedt@cs.kuleuven.be
Department of Computer Science, KU Leuven, Belgium

Editor:

Abstract

We study planning in relational Markov Decision Processes involving discrete and continuous states and actions. This combination of hybrid relational domains has so far not received a lot of attention. While several symbolic approaches have been proposed for hybrid and relational domains separately, they generally do not provide an integrated approach and they often make restrictive assumptions to make exact inference possible. Removing those restrictions requires approximations such as Monte-Carlo methods. We propose HyBrel: a sample-based planner for hybrid relational domains that combines model-based approaches with state abstraction. HyBrel samples episodes and uses the previous episodes as well as the model to approximate the Q-function. Abstraction is performed for each sampled episode, this removes typical restrictions of symbolic approaches. In our empirical evaluations, HyBrel is shown to have a wide applicability, confirming the advantage of sampled-based abstraction.

1. Introduction

Markov Decision Processes (MDPs) are the underlying framework for many approaches for probabilistic planning and reinforcement learning. One important extension is that of relational MDPs (Wiering and van Otterlo, 2012), in which state descriptions correspond to conjunctions of facts representing a possible world, and for which symbolic methods can be used to compute the value function exactly. The value function is represented compactly at an abstract (lifted) level, as abstract states represent sets of specific ground states by means of logical expressions. Another extension is a hybrid state-space, in which there are both continuous and discrete variables. Under the assumption of deterministic state transitions, it is possible to solve such hybrid MDPs exactly (Sanner et al., 2011). However, exact solutions are intractable for more general problems, and one has to resort to approximated methods. Despite the interest in hybrid domains and in relational MDPs, their combination has received only little attention so far. In this paper we try to fill this gap. More specifically, we study planning in hybrid relational MDPs, which allows to describe objects, attributes (unary relations) as well as the relationships amongst them, and furthermore, the attributes and relations are not necessary binary but can also be categorical or continuous. In this paper we introduce the planner HybRel for hybrid relational MDPs. HybRel builds on the recent planner HYPE (Nitti et al., 2015). HYPE is a sample-based planner that exploits importance sampling to estimate the Q-function. While HYPE can solve hybrid relational MDPs by considering a grounding of the planning problem, it lacks the advantages of abstraction when the model is available. The proposed HybRel adds an abstraction layer to HYPE with a significant performance improvement. Abstraction is performed at the level of samples, therefore it has a wider applicability than exact symbolic methods. Thus, the contribution of this paper is to provide formal and empirical results of sample-based abstraction by logical reasoning on the model.

2. Background

An MDP (Sutton and Barto, 1998) consists of a set S of states, a set A of actions the agent can take, a state transition model $p(s_{t+1}|s_t, a_t)$ and a reward function $R(s_t, a_t)$, with $s_t \in S$ and $a_t \in A$. The goal of the agent is to maximize the expected (discounted) reward $\mathbb{E}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)] = \mathbb{E}[G_T(E)]$, where $\gamma \in [0, 1]$ is a discount factor, E is the state and action sequence called episode $E = <s_0, a_0, s_1, a_1, ..., s_T, a_T>$ and $G_T(E) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$. If T is finite we have a finite horizon MDP, otherwise an infinite horizon MDP. To maximize the expected reward, we want to find a policy $\pi_d(s)$ that assigns to each state s and the remaining d steps (horizon) the action to perform. In this paper we consider probabilistic policies $\pi_d(a_t|s_t)$ that provide for each state and horizon d a distribution over actions. Since the total number of steps $T = t + d$, we omit the horizon for the sake of clarity: $\pi(a_t|s_t)$. The value function (or V-function) in a state s_t with horizon d is defined as $V_d^\pi(s_t) = \mathbb{E}[\sum_{k=0}^{d} \gamma^k R(s_{t+k}, a_{t+k})|s_t, \pi] = \mathbb{E}[G_d(E_t)|s_t, \pi]$, with $E_t = <s_t, a_t, ..., s_T, a_T>$ and $G_d(E_t) = \sum_{k=0}^{d} \gamma^k R(s_{t+k}, a_{t+k})$. The action-value function (or Q-function) in a state/action s_t, a_t with horizon d is defined as

$$Q_d^\pi(s_t, a_t) = \mathbb{E}[\sum_{k=0}^{d} \gamma^k R(s_{t+k}, a_{t+k})|s_t, a_t, \pi] = \mathbb{E}[G_d(E_t)|s_t, a_t, \pi] = \int_{E_t} p(E_t|s_t, a_t, \pi) G_d(E_t) dE_t,$$

where $p(E_t|s_t, a_t, \pi) = p(s_{t+1:T}, a_{t+1:T}|s_t, a_t, \pi) = \prod_{k=t}^{T-1} \pi(a_{k+1}|s_{k+1}) p(s_{k+1}|s_k, a_k)$. In this paper, the state is hybrid relational, that is a set of ground relational atoms (true or false) and pairs (variable, value) with a discrete or continuous range. The number of variables in the state can grow or shrink over time and be countably infinite. Formal programming languages such as BLOG (Milch et al., 2005a) and (Dynamic) Distributional Clauses (Nitti et al., 2013; Gutmann et al., 2011) can be used to define proper distributions in such models.

A sample-based planner uses Monte-Carlo methods to solve an MDP. It samples episodes $E^i = <s_0^i, a_0^i, s_1^i, a_1^i, ..., s_T^i, a_T^i>$, using a policy π and updates $Q_d(s_t^i, a_t^i)$ according to a backup rule, e.g., averaging the total rewards obtained starting from (s_t^i, a_t^i). The policy is improved using a strategy that trades-off exploitation and exploration, e.g., ϵ-greedy.

3. Algorithm

Our approach is sketched in Algorithm 1. HybRel is an extension of HYPE (Nitti et al., 2015). The main contribution is the sample-based abstraction (line 10) and the computation of the Q-function from abstracted episodes (line 5). Superscripts i refer to a state, action or expression in the i-th episode E^i, while $\tilde{Q}$ and $\tilde{V}$ refer to an approximation of the

Algorithm 1 HybRel

```
1: function sampleEpisode($d, s_t^n, n$)    ▷ Horizon $d$, starting state $s_t^n$, index current episode $n$
2:     if $d = 0$ then return 0
3:     end if
4:     for each applicable action $a$ in $s_t^n$ do    ▷ $Q$-function estimation
5:         $\tilde{Q}_d^n(s_t^n, a) \leftarrow R(s_t^n, a) + \frac{\sum_{i<n} w^i \tilde{V}_{d-1}^i(\hat{s}_{t+1}^i)}{\sum_{i<n} w^i}$
6:     end for
7:     $a_t^n \leftarrow policy(\{\tilde{Q}_d^n(s_t^n, a)\})$    ▷ action policy, e.g., $\epsilon$-greedy
8:     sample $s_{t+1}^n \sim p(s_{t+1}|s_t^n, a_t^n)$
9:     $(\hat{s}_{t+1}^n, v) \leftarrow R(s_t^n, a_t^n) + \gamma$sampleEpisode$(d-1, s_{t+1}^n, n)$
10:    $\hat{s}_t^n \leftarrow$ Regress$(\{R(s_t^n, a_t^n) \wedge \hat{s}_{t+1}^n\}, \{s_t^n, a_t^n, \hat{s}_{t+1}^n\})$    ▷ state abstraction
11:    $\tilde{V}_d^n(s_t^n) \leftarrow \lambda v + (1-\lambda) max_a \tilde{Q}_d^n(s_t^n, a)$
12:    store $(\hat{s}_t^n, \tilde{V}_d^n(s_t^n), d)$
13:    return $(\hat{s}_t^n, \tilde{V}_d^n(s_t^n))$
14: end function
```

Q and V-function. Furthermore, $\hat{s}_t^n$ indicates an abstract state derived from the full state s_t^n. Briefly, in order to sample the n-th episode starting from s_0 with a maximum depth d, we estimate the Q-function for each applicable action[1] using the previous sampled episodes $i < n$ (line 5). Then the action is selected using a given policy (e.g., ϵ-greedy), the next state is sampled, and the planner is called recursively for the remaining steps. Afterwards, the state is abstracted and stored together with its estimated V-function and the horizon. The estimation of the V-function can be a combination of Monte-Carlo with dynamic programming methods (line 11) resembling TD(λ) algorithms. Without abstraction (line 10) the algorithm becomes HYPE. The weight in line 5 is defined as

$$w^i = \frac{p(\hat{s}_{t+1}^i | s_t^n, a)}{q(\hat{s}_{t+1}^i)} \alpha^{(n-i)}, \tag{1}$$

where $\hat{s}_{t+1}^i$ is the stored (abstracted) state from episode i, s_t^n is the current (full) state, a is the action we are trying to evaluate, $q(\hat{s}_{t+1}^i)$ is the probability with which $\hat{s}_{t+1}^i$ has been sampled, while $\alpha^{(n-i)}$ is used to give more weight to more recent episodes, with $0 \ll \alpha \leq 1$.

Consider the following example without abstraction. We have an object that can be pushed in a set of directions; the goal is to move the object close to a target point g. The state consists of the object position (x, y), the reward function is $R((x, y), a) = 100$ if $dist((x, y), g) < 0.1$ (terminal state), -1 otherwise. The state transition model is $p(x_{t+1}, y_{t+1}|x_t, y_t, d_x, d_y) = \mathcal{N}(x_{t+1}, y_{t+1}|x_t + d_x, y_t + d_x, \Sigma)$, where the action $a = d_x, d_y$ basically represents the displacement of the object to which Gaussian noise is added. Let us assume we have sampled already some episodes with depth $d = 10$, and we want to sample the n-th episode starting from $s_0^n = (0, 0)$. We start computing $\tilde{Q}_{10}^n((0, 0), a)$ for each action a (line 5). Thus, we compute the weights w^i using (1) for each stored sample $\tilde{V}_9^i(s_1^i)$. For example, Figure 1 shows the computation of $\tilde{Q}_{10}^n((0, 0), a)$ for action a' and a'', where we have 3 previous samples. A shadow represents the likelihood $p(s_1^i|s_0^n, a)$ (left for a' and right for a''). The weight (1) of each sample s_1^i is obtained by dividing this likelihood by $q(s_1^i)$ (assuming for simplicity $\alpha = 1$). If $q(s_1^i)$ is uniform, sample 2 with total reward 98 will have higher weight than sample 1, while sample 3 will be largely ignored. The situation

1. If the action space is continuous, we can discretize the action space or sample from it.

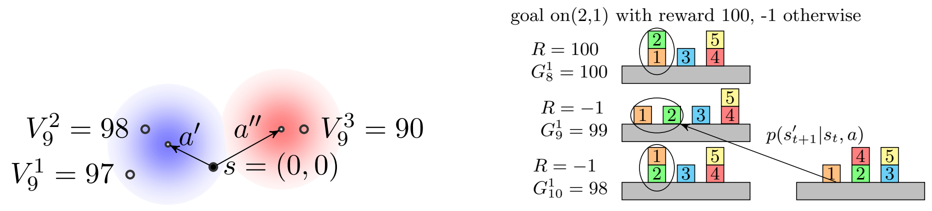

Figure 1: Weight computation for a continuous domain (left), and for the blocksworld with abstraction (right).

is reversed for a''. Note that we can estimate $\tilde{Q}_d^n(s_t^n, a)$ using episodes that may never encounter s_t^n, a provided that $p(s_{t+1}^i|s_t^n, a) > 0$ for some stored episodes i.

4. Q-function estimation without abstraction

This section motivates the algorithm without abstraction (HYPE) as proposed by Nitti et al. (2015). Let us consider the Monte-Carlo interpretation ($\lambda = 1$) starting from standard off-policy Monte-Carlo (Sutton and Barto, 1998) where the Q-function for a policy π and horizon d is estimated from episodes generated from another policy by importance sampling:

$$\begin{aligned} Q_d^\pi(s_t^n, a) &= \mathbb{E}[G_d(E_t)|s_t^n, a, \pi] = R(s_t^n, a) + \gamma\, \mathbb{E}[G_{d-1}(E_{t+1})|s_t^n, a, \pi] \\ &\approx R(s_t^n, a) + \frac{1}{\sum w(E_{t+1}^i)} \gamma \sum_i w(E_{t+1}^i) G_{d-1}(E_{t+1}^i). \end{aligned} \tag{2}$$

The weight $w(E_{t+1}) = \frac{p(E_{t+1})}{q(E_{t+1})}$ is the ratio of the target $p(E_{t+1}) = p(s_{t+1:T}, a_{t+1:T}|s_t^n, a, \pi)$ and the proposal distribution $q(E_{t+1})$. Standard methods use $q(E_{t+1}) = p(s_{t+1:T}, a_{t+1:T}|s_t^n, a, \pi')$, i.e., the episodes are generated with another policy π' starting from s_t^n, a. In contrast, to evaluate Q at episode n, Nitti et al. (2015) uses all previously sampled episodes E_{t+1}^i generated from s_0 that do not necessarily pass by s_t^n, a. Since the formula requires only E_{t+1}^i, i.e., episode subsets from time $t+1$, the proposal distribution is the marginal $q(E_{t+1}) = p(s_{t+1:T}, a_{t+1:T}|s_0, \pi^i) = q(s_{t+1})\pi^i(a_{t+1}|s_{t+1}) \prod_{k=t+1}^{T-1} \pi^i(a_{k+1}|s_{k+1}) p(s_{k+1}|s_k, a_k)$, where $q(s_{t+1}) = p(s_{t+1}|s_0, \pi^i)$. At episode n, the weight $w(E_{t+1}^i)$ in (2) for $i < n$ becomes:

$$\begin{aligned} w(E_{t+1}^i) &= \frac{p(s_{t+1}^i \mid s_t^n, a)\pi(a_{t+1}^i \mid s_{t+1}^i) \prod_{k=t+1}^{T-1} \pi(a_{k+1}^i \mid s_{k+1}^i) p(s_{k+1}^i \mid s_k^i, a_k^i)}{q(s_{t+1}^i)\pi^i(a_{t+1}^i \mid s_{t+1}^i) \prod_{k=t+1}^{T-1} \pi^i(a_{k+1}^i \mid s_{k+1}^i) p(s_{k+1}^i \mid s_k^i, a_k^i)} \\ &= \frac{p(s_{t+1}^i \mid s_t^n, a)}{q(s_{t+1}^i)} \frac{\prod_{k=t}^{T-1} \pi(a_{k+1}^i \mid s_{k+1}^i)}{\prod_{k=t}^{T-1} \pi^i(a_{k+1}^i \mid s_{k+1}^i)} && (3) \\ &\approx \frac{p(s_{t+1}^i|s_t^n, a)}{q(s_{t+1}^i)} \alpha^{(n-i)} && (4) \end{aligned}$$

The policy ratio evaluation (left fraction of (3)) is hard to compute without an explicit representation of the target policy, therefore Nitti et al. (2015) replaces[2] the policy ratio with $\alpha^{(n-i)}$ obtaining (1). Ignoring abstraction, formula (2) with weights defined by (4) is used in line 5, with $\lambda = 1$. Since we are performing policy improvement, each episode is sampled

2. This is similar to the standard recently-weighted average Sutton and Barto (1998).

from a different policy. Shelton (2001) showed that samples from different distributions can be considered as sampled from a single distribution that is the mixture of the true distributions. Thus, during the sampling of episode n, for each previous episode $i < n$: $q(s^i_{t+1}) = \frac{1}{n-1}\sum_{j<n} p(s^i_{t+1} \mid s_0, \pi_j)$. In general, $q(s^i_{t+1})$ is not available in closed form; Nitti et al. (2015) considered the following approximation, that we also use for abstracted states:

$$q(s^i_{t+1}) = \frac{1}{n-1}\sum_{j=1}^{n-1} p(s^i_{t+1}|s_0,\pi_j) = \frac{1}{n-1}\sum_{j=1}^{n-1}\int_{s_t,a_t} p(s^i_{t+1}|s_t,a_t)p(s_t,a_t|s_0,\pi_j)ds_t,a_t \approx \frac{1}{n-1}\sum_{j=1}^{n-1} p(s^i_{t+1}|s^j_t,a^j_t).$$

5. Abstraction

By exploiting the (relational) model, we can improve the algorithm by using abstract states, because often, only part of the state is relevant to determine the total reward. The idea is to generalize the specific states into abstract states by removing the irrelevant facts (for the outcome of the episode). This resembles symbolic methods to exactly solve MDPs in propositional and relational domains (Wiering and van Otterlo, 2012). However, symbolic methods are more challenging in hybrid relational MDPs, indeed exact inference is intractable in general. To overcome these difficulties, we propose to perform abstraction at the level of samples, as we will describe in the next section.

5.1 Derivation

In this section we formalize the sample-based abstraction. For an episode from time t $E_t = < s_t, a_t, ..., s_T, a_T >$, let us consider an arbitrary partition $E_t = \{\hat{E}_t, E'_t\}$ such that $G_d(E_t) = G_d(\hat{E}_t)$, i.e., the total reward depends only on $\hat{E}_t$. The relevant part of the episode has the form $\hat{E}_t = <\hat{s}_t, a_t, ..., \hat{s}_T, a_T>$, while $E'_t = E_t \setminus \hat{E}_t = <s'_t, ..., s'_T>$ is the remaining non-relevant part[3]. The partial episode $\hat{E}_t$ is called *abstract* because the irrelevant variables have been marginalized, in contrast E_t is called full or complete. At episode n, the Q-function estimation from (full) state s^n_t and action a can be reformulated as follows:

$$\begin{aligned}
Q^\pi_d(s^n_t, a) &= \int_{E_t} p(E_t|s^n_t, a, \pi)G_d(E_t)dE_t = \int_{\hat{E}_t}\left(\int_{E'_t} p(\hat{E}_t, E'_t|s^n_t, a, \pi)dE'_t\right)G_d(\hat{E}_t)d\hat{E}_t = \\
&= \int_{\hat{E}_t} p(\hat{E}_t|s^n_t, a, \pi)G_d(\hat{E}_t)d\hat{E}_t = R(\hat{s}^n_t, a) + \gamma\int_{\hat{E}_{t+1}} p(\hat{E}_{t+1}|s^n_t, a, \pi)G_{d-1}(\hat{E}_{t+1})d\hat{E}_{t+1} = \\
&= R(\hat{s}^n_t, a) + \gamma\int_{\hat{E}_{t+1}} \underbrace{\frac{p(\hat{E}_{t+1}|s^n_t, a, \pi)}{q(\hat{E}_{t+1})}}_{w(\hat{E}_{t+1})} q(\hat{E}_{t+1})G_{d-1}(\hat{E}_{t+1})d\hat{E}_{t+1} \\
&\approx R(\hat{s}^n_t, a) + \frac{1}{\sum_{i<n} w(\hat{E}^i_{t+1})}\gamma\sum_{i<n} w(\hat{E}^i_{t+1})G_{d-1}(\hat{E}^i_{t+1}). \qquad (5)
\end{aligned}$$

We use importance sampling as in the non-abstract case, but this time the samples correspond to the previous abstracted episodes $\hat{E}^i_{t+1}$ for $i < n$. This formula is valid for any partition such that $G_d(E_t) = G_d(\hat{E}_t)$, but computing the weights $w(\hat{E}_{t+1})$ for importance sampling might be hard in general. Let us assume that the state transition model obeys the Markov assumption on abstract states, i.e., $p(\hat{s}_{t+1} \mid s_{0:t}, a_t) = p(\hat{s}_{t+1}|\hat{s}_t, a_t)$.

3. We assumed that the actions are relevant, otherwise they will belong to E'.

To estimate $Q^{\pi}_{d}(s^n_t, a)$ (episode n), the weight for episode $i < n$ becomes the following:

$$
\begin{aligned}
w(\hat{E}_{t+1}) &= \frac{p(\hat{E}_{t+1}|s^n_t, a, \pi)}{q(\hat{E}_{t+1})} = \frac{\int_{E'_{t+1}} p(s_{t+1:T}, a_{t+1:T}|s^n_t, a, \pi)dE'_{t+1}}{\int_{E'_{t+1}} p(s_{t+1:T}, a_{t+1:T}|s_0, a_0, \pi^i)dE'_{t+1}} = \frac{p(\hat{s}_{t+1:T}, a_{t+1:T}|s^n_t, a, \pi)}{p(\hat{s}_{t+1:T}, a_{t+1:T}|s_0, a_0, \pi^i)} \\
&= \frac{p(\hat{s}_{t+1}|s^n_t, a)\pi(a_{t+1} \mid \hat{s}_{t+1}, s^n_t, a)\prod_{k=t+1}^{T-1}\pi(a_{k+1}|\hat{s}_{t+1:k+1}, s^n_t, a)p(\hat{s}_{k+1}|\hat{s}_k, a_k)}{q(\hat{s}_{t+1})\pi^i(a_{t+1} \mid \hat{s}_{t+1}, s_0, a_0)\prod_{k=t+1}^{T-1}\pi^i(a_{k+1}|\hat{s}_{t+1:k+1}, s_0, a_0)p(\hat{s}_{k+1}|\hat{s}_k, a_k)} \\
&= \frac{p(\hat{s}_{t+1}|s^n_t, a)}{q(\hat{s}_{t+1})}\frac{\prod_{k=t}^{T-1}\pi(a_{k+1}|\hat{s}_{t+1:k+1}, s^n_t, a)}{\prod_{k=t}^{T-1}\pi^i(a_{k+1}|\hat{s}_{t+1:k+1}, s_0, a_0)} && (6) \\
&\approx \frac{p(\hat{s}_{t+1}|s^n_t, a)}{q(\hat{s}_{t+1})}\frac{\prod_{k=t}^{T-1}\pi(a_{k+1}|\hat{s}_{t+1:k+1})}{\prod_{k=t}^{T-1}\pi^i(a_{k+1}|\hat{s}_{t+1:k+1})} && (7) \\
&\approx \frac{p(\hat{s}_{t+1}|s^n_t, a)}{q(\hat{s}_{t+1})}\alpha^{(n-i)}. && (8)
\end{aligned}
$$

Note that the proposal $q(\hat{E}_{t+1})$ for a generic abstracted episode is the probability used to sample such episode as in the non-abstracted case. However, this time we need to marginalize the non-relevant variables: $q(\hat{E}_{t+1}) = \int_{E'_{t+1}} q(E_{t+1})dE'_{t+1}$. After the marginalization of E'_{t+1}, the probabilities π, π^i of choosing an action (policy) might depend also on the previous states. If we assume that the action probabilities do not depend on the initial state we obtain (7). Since the policy is assumed to improve every episode, we replace the policy ratio with a quantity that favours recent episodes as in the propositional case (formula (8)). HybRel adopts formula (5) and weights (8) for Q-function estimation. Note that during episode sampling the states are complete, nonetheless, to compute $Q^{\pi}_{d}(s^n_t, a)$ at episode n all previous abstracted episodes $i < n$ are considered. Finally, when the sampling of episode n is terminated, it can be abstracted (line 10) and stored (line 12).

Before explaining HybRel abstraction in detail, let us consider an alternative solution that samples abstract episodes directly, instead of sampling a complete episode and performing abstraction afterwards. For this purpose, note that formula (5) refers to the Q-function at full state s^n_t and action a, nonetheless, the assumption $p(\hat{s}_{t+1}|s_{0:t}, a_t) = p(\hat{s}_{t+1}|\hat{s}_t, a_t)$ and approximation (7) make the weights, and thus the Q-function, dependent only on the partial state: $Q^{\pi}_{d}(s^n_t, a) = Q^{\pi}_{d}(\hat{s}^n_t, a)$. Therefore, if we are able to determine and sample partial states $\hat{s}^n_t$, we can sample abstract episodes directly and perform Q-function estimation. Sampling the relevant partial episode $\hat{E}_t$ can be easily performed using lazy instantiation, where given the query $G_d(E_t)$, relevant variables are sampled until the query is answered. Lazy instantiation can exploit context-specific independencies and be extended for distributions with a countably infinite number of variables, as in BLOG (Milch et al., 2005b,a). Similarly, Distributional Clauses search relevant variables using backward reasoning, while sampling is performed in a forward way. For example, to prove R_t the algorithm needs to sample the variables $\hat{s}_t$ relevant for R_t, $\hat{s}_t$ depends on $\hat{s}_{t-1}$ and the action a_{t-1}, the action depends on the admissible actions and the Q-function that again depend on $\hat{s}_{t-1}$, and so on. At some point variables can be sampled because they depend on known facts (e.g., initial state s_0). This procedure guarantees that $G_d(E_t) = G_d(\hat{E}_t)$, $p(\hat{s}_{t+1}|s_{0:t}, a_t) = p(\hat{s}_{t+1}|\hat{s}_t, a_t)$ and $\pi(a|s_t) = \pi(a|\hat{s}_t)$, thus (6) simplifies to $w(\hat{E}_{t+1}) = \frac{p(\hat{s}_{t+1}|\hat{s}_t,a)}{q(\hat{s}_{t+1})}\frac{\prod_{k=t}^{T-1}\pi(a_{k+1}|\hat{s}_{k+1})}{\prod_{k=t}^{T-1}\pi^i(a_{k+1}|\hat{s}_{k+1})}$, and the approximation of (7) is not needed. Unfortunately, this method avoids only sampling variables that are completely irrelevant, therefore in many practical domains it will sample (almost) the entire state. For example, evaluating the admissible actions often requires sampling the entire state.

5.2 Sample-based abstraction

In this section we describe the proposed sample-based abstraction. HybRel samples complete episodes and performs abstraction afterwards. The abstraction of $\hat{E}_t$ from E_t (line 10) is decomposed recursively employing backward reasoning (regression) from the last step $t = T$ and repeated backwards till reaching s_0. We first regress $R(s_T, a_T)$ using s_T to obtain the abstract state $\hat{s}_T = \hat{E}_T$ (computing the most general $\hat{s}_T$ such that $R(\hat{s}_T, a_T) = R(s_T, a_T)$). For $t = T-1, ..., 0$ we regress $R(s_t, a_t) \wedge \hat{s}_{t+1}$ using $a_t, s_t \in E_t$ to obtain the most general $\hat{s}_t \subseteq s_t$ that guarantees $R(\hat{s}_t, a_t) = R(s_t, a_t)$ and $p(\hat{s}_{t+1}|s_t, a_t) = p(\hat{s}_{t+1}|\hat{s}_t, a_t)$. Note that $\hat{E}_t = \hat{s}_t \cup \hat{E}_{t+1}$. This method assumes that the actions are given, thus it avoids to prove the admissible actions, keeping the abstract states smaller. For this reason, REGRESS guarantees only $G_d(E_t) = G_d(\hat{E}_t)$ and $p(\hat{s}_{t+1}|\hat{s}_{0:t}, a_t) = p(\hat{s}_{t+1}|\hat{s}_t, a_t)$, thus, approximations (7) and (8) are needed to compute the weight. We argue that this is a reasonable approximation, confirmed by empirical evaluation. Furthermore, each episode is sampled using complete states, thus every applicable action is evaluated and even the same action can produce episodes with different abstractions. Note that derivation (5) assumes a fixed partition, thus exploits only conditional independencies, but the idea can be extended to context-specific independencies. To illustrate the algorithm, consider the blocksworld example in Fig. 1. In this domain we can move objects with no objects on top, and the action succeeds with a certain probability or leaves the state unchanged. Let us consider the abstraction of the episode on the left. To prove the last reward we need to prove the goal, thus $\hat{s}_2 = \mathtt{on(2,1)}_2$. Now let us consider time step 1, the proof for the immediate reward is $\mathtt{not(on(2,1)}_1\mathtt{)}$, while the proof for the next abstract state $\hat{s}_2$ is $\mathtt{on(1,table)}_1, \mathtt{on(2,table)}_1, \mathtt{clear(1)}_1, \mathtt{clear(2)}_1$, therefore the abstract state becomes $\hat{s}_1 = \mathtt{on(1,table)}_1, \mathtt{on(2,table)}_1, \mathtt{clear(1)}_1, \mathtt{clear(2)}_1, \mathtt{not(on(2,1)}_1\mathtt{)}$. Analogously, $s'_0 = \mathtt{on(1,2)}_0\mathtt{,on(2,table)}_0\mathtt{,clear(1)}_0\mathtt{,not(on(2,1)}_0\mathtt{)}$. The same procedure is applicable to continuous variables. The abstraction helps to exploit the previous episodes in more cases, speeding up the convergence. For example, Fig. 1 shows the computation of a weight from the state depicted on the right to the abstract state pointed by the arrow. If the action is moving 4 on top of 5 we have $p(\hat{s}_{t+1}|s_t, a) > 0 \Rightarrow w > 0$, in contrast without abstraction all actions get weight 0, thus that episode cannot be used to compute the Q-function.

6. Related work

As showed, this work is an extension of Nitti et al. (2015). Other notable sample-based planners are UCT (Kocsis and Szepesvári, 2006) based on upper confidence bounds, and Sparse Sampling (Kearns et al., 2002) that works with no particular assumptions. Those and many other sample-based planners do not exploit the availability of the model (the actual probabilities) and do not perform abstraction. Among sample-based planners for probabilistic relational models there is PRADA (Lang and Toussaint, 2010), though it supports only discrete action-state spaces. Several exact methods for propositional and relational MDPs have been proposed. Symbolic methods that perform computation at the level of abstract states. This idea is used in propositional and relational domains. State-of-the-art algorithms can solve MDPs exactly with discrete action-states in propositional (see Mausam and Kolobov (2012) for a review) and relational MDPs (e.g., Kersting et al. (2004)). There are also propositional solutions for hybrid domains (Zamani et al., 2012), but they require a deterministic transition model to deal with the continuous variables.

7. Experiments

HYPE (Nitti et al., 2015) was tested in several domains, outperforming Sparse Sampling (Kearns et al., 2002). In this section we empirically evaluates HybRel with respect to HYPE, i.e., the effect of episode abstraction on performance. Dynamic Distributional Clauses were adopted for modelling and inference, and the algorithm was implemented in YAP Prolog and C++, and run on a Intel Core i7.

We performed experiments with the blocksworld (BW) and a continuous version of it (BWC) with an energy level of the agent and object weights. The energy decreases with a quantity proportional to the weight of the object moved plus Gaussian noise. If the energy becomes zero the action fails, otherwise the probability of success is 0.9. The reward is -1 before reaching the goal and $Energy$ if the goal is reached. For these experiments (Table 1, setting A) we sampled a total of 200 episodes to find a policy and execute it. The results are averaged over 100 runs. In addition, we repeated the experiments ignoring the horizon for episodes that reach the goal (setting B in Table 1). In the current implementation abstraction is not supported with negation. Therefore, the domain has been propositionalized to describe explicitly what is true and what is false. For HYPE (without abstraction) we show time performance in the propositionalized domain and in the relational version between brackets. The results highlight that abstraction improves the performance significantly.

Table 1: Experiments. d is the horizon, 'success' is the number of times the goal is reached. For blocksworld with 4 objects (BW 4, BWC 4) we used a goal of 3 facts and a goal of 4 facts when we use 6 objects (BW 6 and BWC 6).

Setting A					
domain	d	abstract	reward	success	time (s)
BW 4	10	NO	77.8	85%	175 (36)
BW 4	10	YES	**83.3**	**91%**	84
BW 6	16	NO	-16	0%	596 (126)
BW 6	16	YES	**57.9**	**71%**	280
BWC 4	10	NO	3.8	79%	312 (44)
BWC 4	10	YES	**5.4**	**95%**	100
BWC 6	18	NO	-18.0	0%	976 (170)
BWC 6	18	YES	**1.3**	**98%**	288

Setting B					
domain	d	abstract	reward	success	time (s)
BW 4	10	NO	90.9	97%	315 (75)
BW 4	10	YES	**94.1**	**100%**	262
BW 6	16	NO	-13.9	2%	613 (128)
BW 6	16	YES	**79.3**	**90%**	750
BWC 4	10	NO	3.8	81%	319 (70)
BWC 4	10	YES	**5.3**	**96%**	316
BWC 6	18	NO	-17.7	1%	985 (183)
BWC 6	18	YES	**2.2**	**98%**	1127

8. Conclusions

We proposed a sample-based planner for hybrid relational MDPs that extends the planner HYPE. We formally described how (context-specific) independence assumptions can be exploited to perform abstraction. This is valid for propositional as well as relational domains. In addition, empirical results showed that abstraction provides important improvements with respect to HYPE.

References

Bernd Gutmann, Ingo Thon, Angelika Kimmig, Maurice Bruynooghe, and Luc De Raedt. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 2011.

Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. *Machine Learning*, 49(2-3): 193–208, 2002.

Kristian Kersting, Martijn Van Otterlo, and Luc De Raedt. Bellman goes relational. In *Proc. ICML*, page 59, 2004.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-carlo Planning. In *Proc. ECML*, 2006.

Tobias Lang and Marc Toussaint. Planning with Noisy Probabilistic Relational Rules. *Journal of Artificial Intelligence Research*, 39:1–49, 2010.

Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* Morgan & Claypool Publishers, 2012.

Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic Models with Unknown Objects. In *Proc. IJCAI*, 2005a.

Brian Milch, Bhaskara Marthi, David Sontag, Stuart Russell, Daniel L. Ong, and Andrey Kolobov. Approximate Inference for Infinite Contingent Bayesian Networks. In *Proc. of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005b.

Davide Nitti, Tinne De Laet, and Luc De Raedt. A Particle Filter for Hybrid Relational Domains. *Proc. IROS*, 2013.

Davide Nitti, Vaishak Belle, and Luc De Raedt. Planning in discrete and continuous markov decision processes by probabilistic programming. *to appear in ECML/PKDD*, 2015.

Scott Sanner, Karina Valdivia Delgado, and Leliane Nunes de Barros. Symbolic Dynamic Programming for Discrete and Continuous State MDPs. In *Proc. UAI*, pages 643–652, 2011.

Christian Robert Shelton. *Importance Sampling for Reinforcement Learning with Multiple Objectives.* PhD thesis, MIT, 2001.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

M. Wiering and M. van Otterlo. *Reinforcement Learning: State-of-the-Art.* Adaptation, Learning, and Optimization. Springer, 2012.

Zahra Zamani, Scott Sanner, and Cheng Fang. Symbolic Dynamic Programming for Continuous State and Action MDPs. In *Proc. AAAI*, 2012.