PAC-MDP Learning with Knowledge-based Admissible Models

Marek Grześ and Daniel Kudenko
Department of Computer Science, University of York
Heslington, YO10 5DD York, UK
{grzes, kudenko}@cs.york.ac.uk

ABSTRACT

PAC-MDP algorithms approach the exploration-exploitation problem of reinforcement learning agents in an effective way which guarantees that with high probability, the algorithm performs near optimally for all but a polynomial number of steps. The performance of these algorithms can be further improved by incorporating domain knowledge to guide their learning process. In this paper we propose a framework to use partial knowledge about effects of actions in a theoretically well-founded way. Empirical evaluation shows that our proposed method is more efficient than reward shaping which represents an alternative approach to incorporate background knowledge. Our solution is also very competitive when compared with the Bayesian Exploration Bonus (BEB) algorithm. BEB is not PAC-MDP, however it can exploit domain knowledge via informative priors. We show how to use the same kind of knowledge in the PAC-MDP framework in a way which preserves all theoretical guarantees of PAC-MDP learning.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Experimentation, Theory

Keywords

Domain knowledge, Heuristics, Reinforcement learning

1 INTRODUCTION

Reinforcement learning represents a natural approach to implementing learning capability in autonomous agents and multi-agent systems. One of the key research challenges in the area of reinforcement learning (RL) is how to balance the exploration-exploitation tradeoff. In RL, learning and acting are intertwined and the exploration strategy has an immense impact on the learning speed (often also on the quality of the final solution) and the data which is provided for learning. This paper introduces knowledge-based extensions to the class of PAC-MDP RL methods (PAC stands for Probably Approximately Correct) and shows theoretical guarantees that proposed modifications do not violate theoretical assumptions of PAC-MDP learning.

Cite as: PAC-MDP Learning with Knowledge-based Admissible Models, Marek Grześ and Daniel Kudenko, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 349-356

One of the best approaches to exploration in RL, which has good theoretical properties, is the so called PAC-MDP approach. State-of-the-art examples of this idea are E^3 [12] and $Rmax$ [7]. This approach defines the exploration strategy which guarantees that with high probability the algorithm performs near optimally for all but a polynomial number of time steps (i.e., polynomial in the relevant parameters of the underlying process).

Most of RL research has focused on the situation when knowledge about the mathematical model of the underlying process is very limited. This is however not always the case in practical applications where some domain knowledge may exist. For example, Poupart et al. [15] indicate that some knowledge can be easily available in navigation scenarios. They also give a concrete example from the area of assistive technology where, in the RL-based hand-washing device, the transition dynamics are known except behaviour probabilities of people with dementia who use the system [6]. In this paper, the use of such partial knowledge about actions of the underlying controlled process is considered to improve the performance of PAC-MDP learning. This is only partial knowledge because it is not sufficient to design the analytical model of the underlying process and (reinforcement) learning is still necessary to solve the problem. The resulting approach is shown to preserve theoretical properties of PAC-MDP learning.

Bayesian techniques can be naturally enhanced with background knowledge through informative priors. A relevant Bayesian approach to the problem of exploration in RL has been recently introduced in [13]. This algorithm applies slightly greedier exploitation than the one which is in PAC-MDP algorithms. This may lead to improvements in some practical situations, however this greedier exploitation makes it not PAC-MDP (see proof in [13]). In this paper, we want to show that knowledge which in Bayesian approaches can be used to define informative priors can be also used in the PAC-MDP framework in a relatively straightforward way and yields a very good empirical improvement of the state-of-the-art algorithms. This allows for the use of background knowledge, obtaining an algorithm which is competitive with the Bayesian approach, and most importantly is still PAC-MDP. The lack of such techniques was one of the points of criticism against PAC-MDP algorithms in [13].

2 MARKOV DECISION PROCESSES AND REINFORCEMENT LEARNING

The underlying mathematical model of the RL methodology is the Markov Decision Process (MDP). An MDP is defined as a tuple $(\mathbb{S}, \mathbb{A}, T, R, \gamma)$, where $s \in \mathbb{S}$ is the state space, $a \in \mathbb{A}$ is the action space, $T(s, a, s')$ is the probability that action a when executed in state s will lead to state s', $R(s, a, s')$ is the immediate

reward received when action a taken in state s results in a transition to state s', and γ is the discount factor which determines how the long-term reward is calculated from immediate rewards [16]. The problem of solving an MDP is to find a policy (i.e., mapping from states to actions) which maximises the accumulated reward. A Bellman equation defines optimality conditions for the situation when the environment dynamics (i.e., transition probabilities and a reward function) are known. In such a case the problem of finding the policy becomes a planning problem which can be solved using iterative approaches like policy and value iteration [3]. These algorithms take $(\mathbb{S}, \mathbb{A}, T, R, \gamma)$ as an input and return a policy which determines which action should be taken in each state so that the long term reward is maximised. In algorithms which represent the policy via the value function $V(s)$ or $Q(s,a)$, such a long term reward is expressed explicitly through $V(s)$ or $Q(s,a)$, where $Q(s,a)$ reflects the expected long term reward when action a is executed in state s.

The policy and value iteration methods require access to an explicit, mathematical model of the environment, that is, transition probabilities, T, and the reward function, R, of the controlled process. When such a model is not available, there is a need for algorithms which can learn from experience. Algorithms which learn the policy from the simulation in the absence of the MDP model are known as reinforcement learning [20, 4]. In many practical situations, even if an explicit, mathematical model of the MDP cannot be constructed, the system can be simulated either directly or via a generative model (it is often easier to build a generative mathematical model than an explicit model of system dynamics [21]).

The first approach to RL is to estimate the missing model of the environment using, e.g., statistical techniques. The repeated simulation is used to approximate or average the model. Once such an estimation of the model is available, standard techniques for solving MDPs, like policy and value iteration, are again applicable. This approach is known as model-based RL [19]. This paper investigates a special type of model-based RL which is known as PAC-MDP learning.

An alternative approach to RL which is not considered in this paper does not attempt to estimate the model of the environment, and because of that is called model-free RL. Algorithms of this type directly estimate the value function or a policy [14] from repeated simulation. The standard examples of this approach constitute Q-learning and SARSA algorithms [20].

3 PAC-MDP ALGORITHMS

PAC-MDP learning represents one of the approaches to exploration in RL. Such algorithms are based on the technique known as optimism in the face of uncertainty [12, 7]. Like in standard model-based learning, the dynamics of the underlying MDP are estimated from data. If a certain state-action pair has been experienced enough times (parameter m), then the Hoeffding bound ensures that the estimated dynamics are close to the true values. Let p be the probability of success in the Binomial distribution, and $\hat{p}$ its empirical estimate. The Hoeffding bound shows that the probability of $|\hat{p} - p| > \epsilon$ for some ϵ is bounded by $\exp(-2m\epsilon^2)$ where m is the number of trials used to estimate $\hat{p}$ [11]. It means that for a given value of ϵ and desired probability, it is possible to determine analytically the value of the number of required trials, m. The optimism under uncertainty plays a crucial role when dealing with state-action pairs which have not been experienced m times. For such pairs, the algorithm assumes the highest possible value of their Q-values. State-action pairs for which $n(s,a) < m$ are named unknown and known when $n(s,a) \geq m$ where $n(s,a)$ is the number of times the state-action pair was experienced. When a new state action pair becomes known, the existing approximation, $\hat{M}$, of the true model, M^*, is used to compute the corresponding optimal policy for $\hat{M}$ which when executed will encourage the algorithm to try unknown actions and learn their dynamics. Such an exploration strategy guarantees that with high probability the algorithm performs near optimally for all but a polynomial number of time steps (i.e., polynomial in the relevant parameters of the underlying MDP).

The precise implementation of this idea is different in existing algorithms. One of the differences comes from the fact how the planning step is implemented. The general equation for performing value iteration for computing the policy, $\hat{\pi}$, for the model $\hat{M}$ can be as follows:

$$\hat{Q}(s,a) = \hat{R}(s,a) + B(s,a) + \gamma \sum_{s'} \hat{T}(s,a,s') \max_{a'} \hat{Q}(s',a'), \quad (1)$$

where $B(s,a)$ is an algorithm specific exploration bonus. In the Rmax algorithm, $B(s,a) = 0$ for all state-action pairs. In the Model Based Interval Estimation with Exploration Bonus (MBIE-EB) algorithm, which is also PAC-MDP, $B(s,a) = \beta/\sqrt{n(s,a)}$ where β is a parameter [18].

The proofs and theoretical analysis of PAC-MDP algorithms can be found in the relevant literature [10, 18]. In our analysis one specific property of such algorithms is advocated, i.e., the optimism under uncertainty which requires that inequality $\hat{V}(s) \geq V^*(s)$ is always satisfied during learning, where $V^*(s)$ is the optimal value function which corresponds to the true MDP model M^*.

4 NEAR BAYESIAN LEARNING

A relevant algorithm which tackles the exploration problem in a way which originates from optimism in the face of uncertainty has been recently proposed in [13]. The Bayesian Exploration Bonus (BEB) algorithm implements computationally tractable approximation of the Bayesian exploration. An important fact from our point of view is that this algorithm differs from the PAC-MDP algorithm mainly in a way the exploration bonus, $B(s,a)$, is computed. In the BEB algorithm $B(s,a) = \beta/(1+n(s,a))$. Because n increases faster than $\sqrt{n}$, it has been proven [13] that BEB is not PAC-MDP. The exploration bonus decays too fast in BEB and does not allow for enough exploration to satisfy the PAC-MDP requirements. Thus, BEB is not guaranteed to converge to the optimal solution. A useful property of the Bayesian approach is however the fact that it can use domain knowledge in a straightforward way through informative priors. The issue of using such knowledge in the PAC-MDP framework has not been analysed and was considered to be its weakness [13]. In our work we are investigating how to use the same knowledge in PAC-MDP algorithms.

5 DOMAIN KNOWLEDGE AND ADMISSIBLE MODELS

The standard scenario when RL is applicable is the situation when dynamics of the controlled stochastic decision process are not available. Actions of such a process can be formally specified with the use of the Probabilistic Planning Domain Description Language (PPDDL) [23]. Each action a is specified in this notation by defining the probability, p_i, of each probabilistic effect, e_i, in the following way:

$$(a\ p_1\ e_1\ ...\ p_n\ e_n).$$

When this notation is used to describe the probabilistic planning problem in which the entire model is known beforehand, all p_i and e_i have to be specified for all actions. In the standard RL scenario,

neither p_i nor e_i are known. However, it is often the case, that even if the entire model is not available, some elements can be determined beforehand (see reference to [6, 15] in Section 1). This paper aims at improving PAC-MDP learning when partial action knowledge is available. Before going into details on what kind of knowledge will be considered, the definition bellow introduces the notion of admissible MDP models which will determine theoretical requirements on application of this knowledge.

DEFINITION 1. *A model $\hat{M}$ is admissible iff the corresponding value function, $\hat{V}$, satisfies inequality $\hat{V}(s) \geq V^*(s)$, that is, $\hat{V}(s)$ is admissible.*

An initial model, $\hat{M}$, has to be admissible in order to preserve PAC-MDP properties when used with existing algorithms which are PAC-MDP [7, 18].

5.1 Optimistic Determinization

In this section we are dealing with RL problems for which all possible outcomes, e_i, of each action, $a \in \mathbb{A}$, can be determined by the designer of the system whereas probabilities of outcomes, p_i, still remain unknown as in the classical RL scenario (such a situation exists, e.g., in the area of assistive technology [6, 15]). Under this condition, learning from simulation is still one of the potential solutions and RL is applicable to solve sequential decision making problems of this kind. The question now is how to use PAC-MDP algorithms with such knowledge about the possible effects of actions. Our solution is motivated by probabilistic planners which apply determinization of stochastic domains [5, 22]. One of the approaches in [22] which yields an admissible deterministic model is based on all-outcomes determinization (AO). In this case, the determinization process creates a new deterministic action $a_d \in \mathbb{A}_d(a)$ for each possible effect, e_i, of a given action a. The new set of deterministic actions replaces the original action. The obtained model is in this case admissible with regard to the original probabilistic one. When this type of determinization is applied in the FF-Replan algorithm [22], probabilities of action outcomes are ignored. In our case this situation is ideal, since we do not have those probabilities in our RL settings anyway. The fact that the AO model is admissible can be easily proven in the same manner as Lemma 6 in [18] proves that models with the upper bound of the estimated interval guarantee admissibility of the value function with high probability. The proof for the case with the AO model is similar and proves that the corresponding value function is admissible with probability one.

LEMMA 1. *For any state s and action a, the condition $\hat{Q}(s,a) \geq Q^*(s,a)$ is satisfied after value iteration on the MDP $\hat{M}$ which is obtained from AO knowledge.*

PROOF. Value iteration solves the MDP $\hat{M}$ defined according to AO knowledge. We prove the claim by induction on the number of steps of value iteration which is stopped after finite number of iterations. For the base case, assume that the Q values are initialised to $Rmax$ when $\gamma = 1$ or $Rmax/(1-\gamma)$ otherwise, for all s. Now, for the induction, suppose that the claim holds for the current value function $\hat{Q}(s,a)$. By assumption, the reward $R(s,a)$ is known exactly and $\hat{T}(s,a,s') = \hat{T}(s,a_d \in \mathbb{A}_d(a), s') = 1$ and $\hat{T}(s,a,s') = 0$ only when $T(s,a,s') = 0$ for sure (according to AO knowledge). The term $Q(s',a')$ on the right-hand side of Equation 1 is the result of the previous iteration and is used to compute the new Q-value $\hat{Q}(s,a)$ on the left-hand side of the equation. By our assumption we know $R(s,a)$ exactly and

$$\sum_{s'} \hat{T}(s,a,s') \max_{a'} \hat{Q}(s',a') = \max_{a_d}[T(s,a_d,s') \max_{a'} \hat{Q}(s',a')]$$
$$= \max_{a_d} \max_{a'} \hat{Q}(s',a') \geq \sum_{s'} T(s,a,s') \max_{a'} \hat{Q}(s',a')$$
$$\geq \sum_{s'} T(s,a,s') \max_{a'} Q^*(s',a')$$

The first step is from the definition how to use a_d to determine values of a. The second and the third steps follow from the assumption that $\hat{T}(s,a,s') = 1 \geq \max_{s'} T(s,a,s')$ or $\hat{T}(s,a,s') = 0$ only when $T(s,a,s') = 0$ for sure, and the fourth from the induction assumption. □

Normally, probabilistic effects reduce the value function via transitions to lower value states and AO determinization leads to higher values in such situations because only the state with the highest value (which is achieved with probability 1.0 in the modified model) is used in the Bellman update.

5.2 Free Space Assumption

The free space assumption (FSA) is an approach to define an initial model of the environment which assumes that all transitions in the environment are possible (in robotic navigation environments it would assume, e.g., that there are no walls between all adjacent states or, in the case of the hand washing device of Boger et al. [6], the person with dementia always behaves like a rational healthy person), and that all actions are deterministic and always lead to a corresponding expected state [17]. In the PPDDL notation, it would mean that a real action a is replaced by outcome e_i for which $e_i = \arg\max_{e_i} p_i$, whereas all p_i may stay unknown. The best outcome selected in this way may fail in the real environment (with $p_i = 0$) however by our assumption other outcomes of this action have their highest p_i in a different action. Thus, without much loss of generality and for the sake of theoretical properties of our solution, we require that for each possible outcome which does not correspond to the most expected outcome of a given action, there is another action which has this outcome as the most expected one. This requirement is necessary to guarantee the admissibility of such a model because all outcomes have to be tested during learning (unless they are blocked). In the hypothetical robotic environment, the formulation of the FSA model would mean, e.g., that an action move forward, always moves the robot form a given state to the state in front of the robot, ignoring any existing walls and probabilistic effects of actions like, e.g., slippery surface which would slow down robot's movement or change the direction of its motion.

The fact that the FSA model is admissible with probability one can be also proved in a similar way as Lemma 1. Admissibility of $\hat{V}(s) = \max_a \hat{Q}(s,a)$ guarantees optimistic behaviour when the highest Q-value is used greedily.

LEMMA 2. *For any state s and action a, the condition $\hat{V}(s) \geq V^*(s)$ is satisfied after value iteration on the MDP $\hat{M}$ which is obtained from FSA knowledge.*

PROOF. Value iteration solves the MDP $\hat{M}$ defined according to FSA knowledge. We prove the claim by induction on the number of steps of value iteration which is stopped after finite number of iterations. For the base case, assume that the Q values are initialised to $Rmax$ when $\gamma = 1$ or $Rmax/(1-\gamma)$ otherwise, for all s. Now, for the induction, suppose that the claim holds for the current value function $\hat{V}(s)$. By assumption, the reward $R(s,a)$ is known exactly and $\hat{T}(s,a_{FSA},s') = 1 \geq \max_{s'} T(s,a,s')$. If

$T(s, a_{FSA}, s') = 0$, then another effect is better which is FSA of another action, so another action will be better for such an effect.

Equation 1 can be expressed also in terms of the value function V. The term $\hat{V}(s')$ on the right-hand side of such an equation is the result of the previous iteration and is used to compute the new V-value $\hat{V}(s)$ on the left-hand side of the equation. By our assumption we know $R(s, a)$ exactly and

$$\max_a[\gamma \sum_{s'} \hat{T}(s, a, s')\hat{V}(s')] = \max_a[\gamma \hat{T}(s, a_{FSA}, s')\hat{V}(s')]$$
$$= \max_a[\gamma \hat{V}(s')] \geq \max_a[\gamma \sum_{s'} T(s, a, s')\hat{V}(s')]$$
$$\geq \max_a[\gamma \sum_{s'} T(s, a, s')V^*(s')] \geq V^*(s)$$

The first step removes summation because each FSA action of a given action a is deterministic. The second step follows from the property $\hat{T}(s, a, s'_{FSA}) = 1$, the third from $\max_{s'} T(s, a, s') \leq 1$, and the fourth is from the induction assumption. □

Informally, the value function computed with the FSA model is always at least as high as the true value because the FSA model will utilise shorter optimistic paths (shorter because of unblocked transitions in the FSA model) and assume that corresponding transitions have always probability 1 (in the real model $p_i \leq 1$).

5.3 Maximal Probability

Two previous sections rely on knowledge about e_i in the PPDDL specification without information about exact values of probabilities p_i. Useful information may be available however in the form of $\varrho = \max p_i$, where max is over the entire state-action space. So, ϱ represents the maximal possible p_i which can occur in a given MDP. The value of ϱ is useful when $\varrho < 1$ as it can be used, e.g., for more accurate evaluation of the admissible potential function for reward shaping in PAC-MDP algorithms [1]. In our analysis it will be shown how to use knowledge about ϱ in our algorithm and how to enhance existing algorithms with which we are comparing our solution in order to obtain more fair comparison and gain better insight into the problem.

There is one more noteworthy issue about domain knowledge in RL. One, well established existing way of incorporating domain knowledge into RL is reward shaping [1]. It requires however knowledge about a sufficiently accurate admissible heuristic in order to preserve PAC-MDP properties of the algorithm. The problem is that in many practical applications it is difficult to define such a heuristic manually, which is the case in domains with symbolic PPDDL-like representations. The use of knowledge which is discussed in this section aims also at dealing with situations when such heuristics cannot be designed. The aim is to use knowledge from this section in an alternative way to reward shaping which is not applicable when there is no admissible heuristic. For better understanding of the problem, our solution will be compared with the reward shaping technique on a domain where the required heuristics can be easily defined.

The main contribution of the paper is a method to apply AO (Section 5.1) and FSA (Section 5.2) knowledge in PAC-MDP algorithms while preserving the PAC-MDP property. We are also showing how to effectively use knowledge about ϱ in various RL algorithms.

	Admissible Model	Estimated Model
$n(s, a) < m$	✓	
$n(s, a) \geq m$		✓

Table 1: The use of knowledge-based admissible models.

6 PAC-MDP LEARNING WITH ADMISSIBLE MODELS

The extension to PAC-MDP learning which is proposed in this section can be applied to any PAC-MDP algorithm. In this paper we are focusing on the Rmax [7] and MBIE-EB [18] algorithms. These algorithms apply the standard procedure of PAC-MDP learning, that is, model estimation, optimism in the face of uncertainty when dealing with unknown state-action pairs and planning according to Equation 1. Our modification is associated mainly with how the process of estimating the MDP model is handled. The use of the AO model requires also specification on how actions are selected for acting and how updates in Equation 1 are performed when AO actions are in the model.

The special treatment of the model used during learning is required because background knowledge needs to be incorporated. In standard Rmax learning, one can distinguish two stages of learning the dynamics of a particular state-action pair (s, a). Initially, when there are no previous executions of (s, a) or when the number of executions does not exceed the value of m, optimism under uncertainty is applied. The second stage is about (s, a) pairs which have been executed at least m times. The use of background knowledge which is considered in this paper improves the way unknown state-action pairs are dealt with (the first stage). Instead of using standard optimism under uncertainty which uniformly rewards each state-action pair with the analytically highest value function, we are proposing using domain knowledge to deal with this particular detail of PAC-MDP learning in a more informative way. The solution which we propose is based on combining two MDP models during learning and using them to estimate one Q-function. The first model is the knowledge-based admissible model which can be designed before the learning process is executed. The second model is the standard model used for estimating transition probabilities in PAC-MDP algorithms. The key idea is to use the knowledge-based model for all state-action pairs which are still not known and the true estimation from experience for all state-action pairs for which $n(s, a) \geq m$. This procedure is summarised in Table 1. It is worth reminding here that $m = 1$ in MBIE-EB. Summarising our idea and explaining Table 1, our approach is to always use 1) either the knowledge-based model (AO or FSA) for unknown state-action pairs, or 2) the estimated model for known state-action pairs (i.e., those for which $n(s, a) \geq m$). Because both these models are admissible, the overall model will be also admissible and it does not violate properties of PAC-MDP learning. The description bellow explains how to implement this idea with AO and FSA knowledge respectively.

6.1 The AO Model

The application of the AO model in the general approach presented above requires some specific changes. The first modification is that the Q-table has to contain all real actions and all AO actions for each state. Q-values of real actions and their AO determinizations are used exclusively for both planning and acting (according to Table 1). The second issue concerns the management of actions in the model. Initially the AO model contains each separate action corresponding to every outcome of each real action. When a corresponding real action becomes known during learning in the es-

timated model, every AO action which corresponds to the learned action is removed from the model (unless there are other unknown real actions which share a given outcome). The second issue is the planning part of the algorithm with the AO model. When value iteration is performed according to Equation 1, real actions are used in the evaluation in the right part of this equation. The AO model is however composed also of actions obtained from determinization. Each real action a has corresponding set of deterministic actions $a_d \in \mathbb{A}_d(a)$. Our solution to planning with the AO model, is to determine the state-action value of the unknown action a using $Q(s,a) = \max_{a_d \in \mathbb{A}_d(s)} Q(s,a_d)$. Each $Q(s,a)$ of a real action is equal to maximal $Q(s,a_d)$ of all its determinizations, $a_d \in \mathbb{A}_d(a)$. A similar approach is necessary also for selecting the best action to execute. When the highest Q-value is due to AO action a_d, the real action a for which $a_d \in \mathbb{A}_d(a)$ is executed (ties are broken randomly).

6.2 The FSA Model

The application of the FSA model requires less modifications. The handling of the model is easier, because there is exactly one action in the FSA model which corresponds to the real action. Thus, the computation of Q-values does not require any modifications as well as the selection of actions to execute, and these elements may remain unchanged. The handling of the FSA model can be encapsulated entirely in procedures for learning and representing the overall model. Q-table represents values for real actions only and FSA actions are used until a corresponding real action becomes known in the estimated model.

6.3 Maximal Probability Knowledge

Knowledge about the maximal probability, ϱ, can be used both with AO and FSA models. By default, these two models are deterministic and the blocked transitions are not only assumed be open, but also they can be successful with probability 1. When the value of ϱ is known, these models can be made more accurate while their admissibility will be preserved. The information about ϱ can be incorporated via the modification of the reward, r, in the model, which becomes $r = r/\varrho$ when $r < 0$ and $r = r\varrho$ when $r > 0$. This modification is performed for updates of all non-real actions. A similar trick was applied in [1] for more accurate evaluation of the potential function for reward shaping where it was shown that it leads to an optimistic value function. This knowledge will be also incorporated into other algorithms for a fair comparison in the experimental section.

7 EXPERIMENTAL VALIDATION

The proposed technique to incorporate background knowledge into the considered family of RL algorithms was evaluated empirically on the navigation maze task that is shown in Figure 1. Because our aim was to compare our approach with a range of relevant algorithms, in particular with reward shaping which represents alternative methods for knowledge incorporation, a scaled up (from 15×15 to 25×25 states) version of the domain from [1], where reward shaping for Rmax was proposed, is used. It is a stochastic domain for which relatively accurate heuristics for potential-based reward shaping can be manually designed. Each action can result in its expected outcome with probability 0.8, and slip into one of two perpendicular directions with probability 0.1 for each of these directions. The reward of -1 is given for the execution of each action. The start state is marked with S. Blocked transitions (walls) between states are marked as solid lines between corresponding states. The RL agent has to learn the highest reward path from the start state S to the goal state G without knowing in advance transition probabilities of the environment. Without much loss of generality the reward model is assumed be known to the agent, which is commonly assumed in the relevant literature [1]. The MDP discount factor γ is 1 in this task.

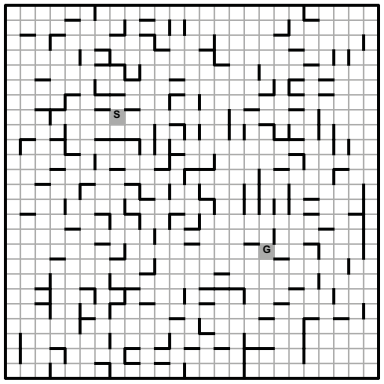

Figure 1: The stochastic navigation maze domain.

Experiments were conducted on a number of algorithms. The details of their settings and parameters are as follows:

- Rmax: The Rmax algorithm with $m = 5$. The versions of this algorithm with AO, Rmax-AO, and FSA, Rmax-FSA, models comprise the major contribution of this paper.
- Rmax with reward shaping [1]: The Manhattan, *RS(Manhattan)*, and straight line, *RS(Line)*, heuristics are used. The potential function was evaluated as $\Phi(s) = r_s \times h(s)/\varrho + r_g$, where $r_s \leq 0$ is the step reward, $h(s)$ is the heuristic estimation of the distance from state s to the goal G, and r_g the reward given when the goal state is reached. The Manhattan heuristic, which is more accurate, was evaluated also with AO and FSA knowledge in the following way: Instead of using uniform optimistic values for Q-values of unknown state-action pairs, with the use of AO and FSA knowledge they can be assigned values which differ within a given state and are more informative. Thus, with AO knowledge $Q(s,a) = V_{max} + max_{a_d \in \mathbb{A}(a)} F(s,a_d,s')$, where $a_d \in \mathbb{A}(a)$ are all determinizations of action a, and with FSA knowledge $Q(s,a) = V_{max} + F(s,a_{FSA},s')$, where a_{FSA} is the FSA action of a. Algorithms, Rmax-AO and Rmax-FSA, which comprise the major contribution of this paper use the same knowledge, and for fair comparison reward shaping was also enhanced with this knowledge.
- MBIE-EB [18]: Before evaluating, this algorithm was tuned for optimal values of the β parameter and the best value was selected for comparisons (such a methodology was also used in [13, 18]). This parameter, β, was evaluated for each configuration separately. This algorithm is also evaluated with AO, MBIE-EB-AO, and FSA, MBIE-EB-FSA, knowledge-based models as proposed in this paper. The algorithm was also tested against one additional improvement which we propose in this paper. In particular, the use of the bonus, $B(s,a)$, can be discarded once the pair has been visited enough times. In our case it was $m_B = 5$. All parameter tuning experiments were performed with the standard version of MBIE and with $m_B = 5$ and for each case the best configuration was selected for comparisons.
- BEB [13]: As in the previous case, the β and the m_B parameters were tuned in the same way. Additionally for fair comparisons with knowledge-based approaches this algorithm was en-

hanced with informative priors based on the AO model knowledge, BEB-AO, and the FSA model knowledge, BEB-FSA. In the first case, $n(s,a) = 1$ for all effects of action a, and $n(s,a) = 3$ when FSA knowledge is available.

- Greedy Optimistic (GO): This algorithm when used without domain specific knowledge corresponds to BEB and MBIE with $\beta = 0$. It can be also considered as a special case of Asynchronous Real Time Dynamic Programming (ARTDP) when used with greedy optimistic exploration [2]. This is not a PAC-MDP algorithm. In our experiments GO-AO (GO-FSA) corresponds to the BEB version of this algorithm with AO (FSA) knowledge.

All algorithms for which knowledge about ϱ is relevant were tested with and without this knowledge. When ϱ is known, its value is 0.8 in our case (because it is the highest possible p_i in the tested domain), otherwise its default value of 1 is used. In all graphs all evaluations were computed for 10 runs of all algorithms. The cumulative score of each algorithm is reported as a function of the number of learning episodes. The error bars of the standard error of the mean (SEM) are also presented [8].

8 RESULTS

The goal of the first series of experiments is to compare our Rmax-based approaches (i.e., Rmax-AO and Rmax-FSA) with basic Rmax and related reward shaping algorithms when they use the same knowledge. Rmax is the most 'cautious' PAC-MDP algorithm in its exploration strategy and our proposed technique should be especially suitable for Rmax. Figure 2 shows results with AO knowledge and $\varrho = 1$. The standard Rmax is the slowest to learn. Reward shaping obtains better learning ratio which is further improved by a more accurate heuristic in RS(Manhattan). RS(Manhattan)-AO is better initially however it does not yield better results than pure RS(Manhattan). This is due to the fact that AO knowledge does not give almost any differentiation of initial Q-values in this case. Our approach, Rmax-AO, showed the best performance. The fact that its performance is better than learning with reward shaping can be explained as follows: In the case of informative models (AO in this experiment and FSA bellow) the knowledge is injected into our Rmax-AO algorithm starting from the very early stages of learning when all state-action pairs are still unknown. Reward shaping yields improvements only when planning takes place, that is, the improvement has an impact only on those state-action pairs which are known. Figure 3 shows the same experiment with $\varrho = 0.8$. Results of all knowledge-based algorithms are better than in Figure 2 and Rmax-AO is still the best. RS(Manhattan)-AO showed slightly better initial learning than RS(Manhattan).

Results with FSA knowledge are shown in Figures 4 and 5. The improvement of the tested algorithms is similar as in the case of AO knowledge. This time however the reward shaping with FSA knowledge, RS(Manhattan)-FSA, is much better than pure reward shaping RS(Manhattan). In this case, the FSA knowledge yields very good differentiation of initial Q-values, because only one outcome for each action is considered and in effect in most cases there is exactly one state-action pair with $\max Q(s,a)$ in a given state. The use of $\varrho = 0.8$ led to further improvement of RS(Manhattan). Though other algorithms improve with the use of FSA knowledge, our Rmax-FSA algorithm was the most efficient in all cases.

A crucial outcome of the experiments discussed so far is the encouraging performance of our Rmax-based technique (Rmax-AO and Rmax-FSA). The overall advantage of the Rmax algorithm in general is that there is only one parameter m and the setting of this parameter is rather straightforward: the higher value of m the more accurate the estimated model is - where this accuracy is formally

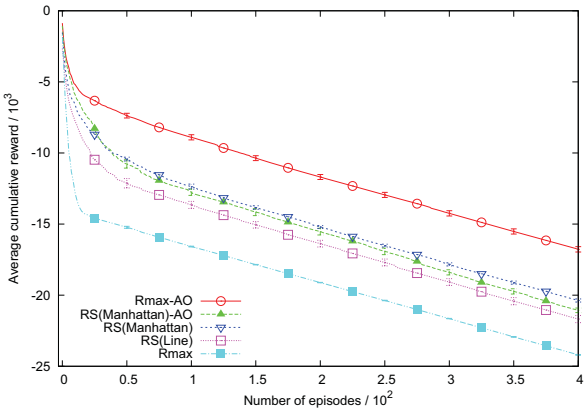

Figure 2: AO knowledge and $\varrho = 1$.

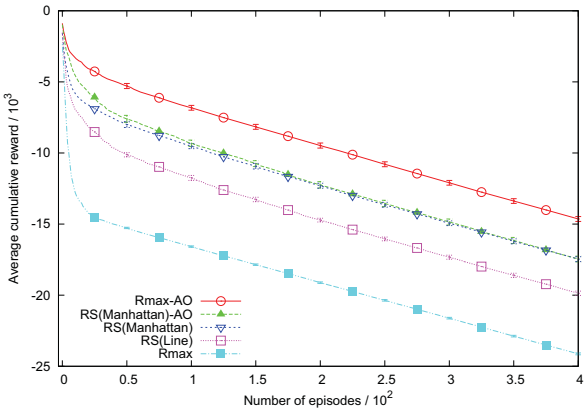

Figure 3: AO knowledge and $\varrho = 0.8$.

specified by the Hoeffding bound introduced in Section 3. In Rmax the planning step is also performed at easily identified milestones (i.e., after a new state-action pair becomes known) whereas MBIE and BEB, which are evaluated bellow, require constant replanning. In the presented results, the Rmax algorithm worked very well with all kinds of knowledge, and reached the same best asymptotic performance in the long term. Obtained results were stable and appropriately improved by provided knowledge.

Our technique introduced in this paper can be also applied to the MBIE-EB algorithm. The next series of experiments is to compare the performance of Rmax and MBIE-EB when these two algorithms apply our technique to use knowledge and to compare them with the BEB algorithm. MBIE-EB and BEB are relatively new algorithms and there are no systematic evaluations and comparisons between these two approaches nor against Rmax. Thus, before doing a final analysis of these algorithms with our extension to incorporate domain knowledge, we are evaluating the performance of basic Rmax, MBIE-EB and BEB on our test domain. With the best parameter configurations for BEB ($\beta = 0.3$ and $m_B = 5$) and MBIE-EB ($\beta = 0.4$ and $m_B = 5$) these two algorithms learned significantly faster than Rmax (BEB faster then MBIE-EB but the difference was not statistically significant), but were not able to reach the same asymptotic convergence as Rmax. The GO algorithm, though performing well initially, obtained the worst asymp-

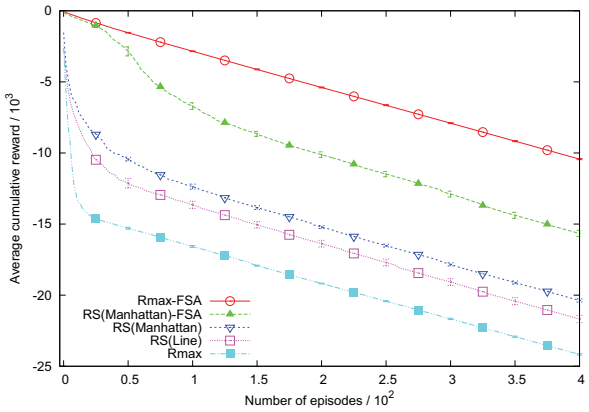

Figure 4: FSA knowledge and $\varrho = 1$.

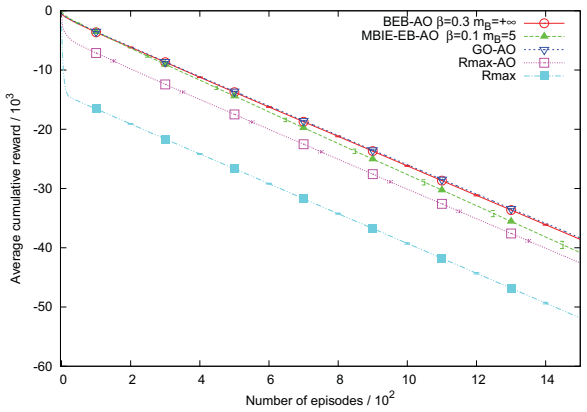

Figure 6: AO knowledge and $\varrho = 0.8$.

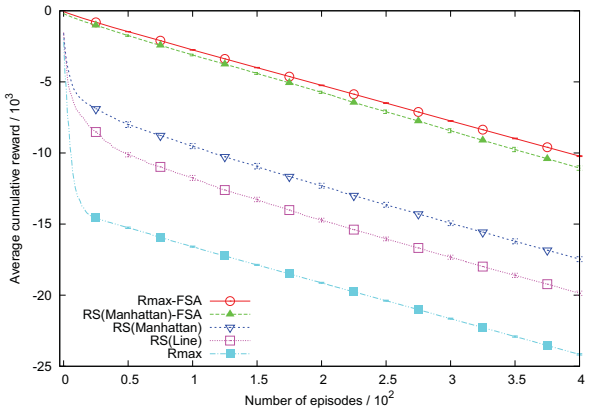

Figure 5: FSA knowledge and $\varrho = 0.8$.

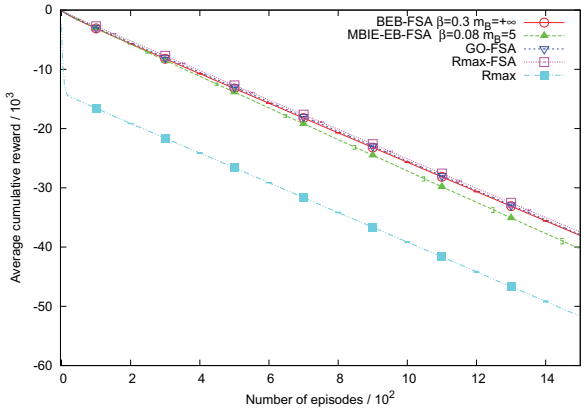

Figure 7: FSA knowledge and $\varrho = 0.8$.

totic performance. In comparisons in [13] both BEB and MBIE-EB do not converge to the optimum as well, but there are no results on Rmax in that comparison. Our results indicate that Rmax is very competitive in terms of finding the optimal solution.

Figures 6 and 7 present the target comparison of BEB, MBIE-EB, GO and Rmax algorithms when they have access to the same knowledge, i.e., AO or FSA knowledge. Results on basic Rmax is also included for reference. As already mentioned, AO knowledge is generally weaker than FSA knowledge because it leads to more optimistic and thus less informative MDP models. This fact is reflected in the performance of Rmax-AO in Figure 6. Rmax is the most 'cautious' and when using less informative knowledge it still has to explore more than other more greedy algorithms. The best performance was due to BEB and GO. The use of knowledge allowed these two algorithms reaching the asymptotic result as good as Rmax.When comparing all algorithms with the use of FSA knowledge (Figure 7), this knowledge yields much more informative model for the Rmax-FSA algorithm and its performance is as good (slightly better in this experiment) as BEB and GO. The MBIE-EB algorithm in this case as well reaches lower asymptotic performance (we present here the results with the best configuration of the parameters of this algorithm). These results are very promising for our extension, because the Rmax algorithm is PAC-MDP, it is easy to tune, and it can reach the learning speed of much more greedy learning algorithms which are not PAC-MDP. The last claim can be explained as follows: Due to its rigourous requirements, Rmax has to explore more than other more greedy non-PAC-MDP algorithms like BEB, which means that it learns slower in many domains as is the case in our experiments. This paper shows that a competitive performance can be obtained under all rigourous requirements of Rmax when it is used with our knowledge-based extension. Another practical remark is that FSA knowledge should be preferred over AO knowledge when our approach is used with the Rmax algorithm.

9 CONCLUSION

Exploration-exploitation is the crucial challenge for autonomous agents which learn from environment feedback using reinforcement learning. PAC-MDP algorithms are particularly effective in practice and interesting from an analytical point of view because they approach the exploration-exploitation problem in a way which guarantees that with high probability, the algorithm performs near optimally for all but a polynomial number of steps. The performance of these algorithms can be further improved by incorporating domain knowledge to guide the learning process. The lack of such methods was shown to be a weak point of PAC-MDP algorithms and alternative non-PAC-MDP methods were proposed and shown to be competitive [13]. In this paper we propose a frame-

work to use partial knowledge about effects of actions in a theoretically well-founded way. The contribution of the paper can be summarised as follows:

- With the use of a symbolic specification of the MDP action in the PPDDL formulation, potentially available domain knowledge was distinguished and it was shown how to use this knowledge with PAC-MDP algorithms in a way which preserves theoretical properties of these algorithms.
- The empirical evaluation shows that our proposed method is more efficient than reward shaping which represents an alternative approach to incorporate background knowledge. Reward shaping requires an admissible heuristic which has to be designed manually (e.g., in domains represented symbolically via PPDDL, it is difficult to design such admissible heuristics). Our solution uses only local action knowledge and can be applied when such heuristics cannot be designed or when designed are not accurate. Our results show that even if such heuristics exist, our approach can be more efficient. Informally, it can be argued that our approach will be always better than reward shaping, because in our case domain knowledge is used when state action pairs are still unknown. Knowledge injected via reward shaping is used only with known state-action pairs. In this comparison, our paper gives also a good insight into the significance of different kinds of knowledge on the learning performance of various PAC-MDP algorithms.
- Our solution is also very competitive when compared with the Bayesian Exploration Bonus (BEB) algorithm. BEB is not PAC-MDP, however it can exploit domain knowledge via informative priors. We show how to use the same kind of knowledge in the PAC-MDP framework in a way which preserves all theoretical guarantees of PAC-MDP learning.
- The presented results indicate also that FSA knowledge leads to more informative admissible models and should be preferred to AO knowledge when applied to PAC-MDP algorithms such as Rmax.

This work was motivated by our goals of advancing further the use of symbolic representations (e.g., PPDDL) in RL. The technique presented in this paper will be considered in our future work on such representations. Applicability of our approach is relatively straight-forward in domains with a PPDDL description. This representation is behind the theoretical explanation of our solution and exists in many practical RL/planning domains [9]. In this paper, we focused experiments on the Maze domain because it allowed for detailed comparison with reward shaping approaches, which rely on admissible heuristics. For the Maze domain, we could design such heuristics and obtained detailed comparisons. PPDDL search spaces are massively broader than those in mazes, and we expect more significant improvements on this kind of problems in the future work. Additionally, it is extremely difficult to design admissible heuristics for such domains, so reward shaping cannot guarantee PAC-MDP properties. In contrast, our solution is proven to be PAC-MDP for the broad family of PPDDL domains.

As shown in this paper, AO and FSA knowledge displayed encouraging speed-up of PAC-MDP learning. This kind of knowledge is directly based on the PPDDL representation, therefore it is easy to acquire and understand. It would be interesting to see in future work if different (either more general or more specific) types of domain knowledge would meet requirements of PAC-MDP learning. This could be, e.g., 'feature-based heuristics' which indicate that certain actions are more promising than other actions.

10 ACKNOWLEDGMENT

We would like to thank members of the RL^3 group from Rutgers University for useful discussions on PAC-MDP learning.

11 References

[1] J. Asmuth, M. L. Littman, and R. Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proc. of AAAI*, 2008.

[2] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, 1995.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control (2 Vol Set)*. Athena Scientific, 3rd ed., 2007.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5] A. L. Blum and J. C. Langford. Probabilistic planning in the graphplan framework. In *Proc. of ECP*, pages 8–12, 1998.

[6] Boger, J., Poupart, P., Hoey, J., Boutilier, C., Fernie, G., and Mihailidis, A. A decision-theoretic approach to task assistance for persons with dementia. In *Proc. of IJCAI*, pages 1293–1299, 2005.

[7] R. I. Brafman and M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *JMLR*, pages 213–231, 2002.

[8] P. R. Cohen. *Empirical methods for artificial intelligence*. MIT Press, 1995.

[9] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning, Theory and Practice*. Morgan Kaufmann, 2004.

[10] S. M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College, London, 2003.

[11] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, 1994.

[12] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *ML*, pages 209–232, 2002.

[13] J. Z. Kolter and A. Ng. Near-bayesian exploration in polynomial time. In *Proc. of ICML*, 2009.

[14] A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *In Proc. of UAI*, pages 406–415, 2000.

[15] Poupart, P., Vlassis, N., Hoey, J., and Regan, K. An analytic solution to discrete bayesian reinforcement learning. In *Proc. of ICML*, pages 697–704, 2006.

[16] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[17] D. C. Rayner, K. Davison, V. Bulitko, K. Anderson, and J. Lu. Real-time heuristic search with a priority queue. In *Proc. of IJCAI*, pages 2372–2377, 2007.

[18] A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for markov decision processes. *JCSS*, 74:1309–1331, 2008.

[19] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proc. of ICML*, pages 216–224, 1990.

[20] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[21] G. J. Tesauro. TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[22] S. W. Yoon, A. Fern, and R. Givan. Ff-replan: A baseline for probabilistic planning. In *Proc. of ICAPS*, pages 352–359, 2007.

[23] H. L. S. Younes and M. L. Littman. PPDDL1.0: An extension to PPDDL for expressing planning domains with probabilistic effects. Techn. Rep. CMU-CS-04-162, 2004.