In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence* (IJCAI 05), pp. 752-757, Edinburgh, Scotland, UK, August 2005.

State Abstraction Discovery from Irrelevant State Variables

Nicholas K. Jong
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
nkj@cs.utexas.edu

Peter Stone
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
pstone@cs.utexas.edu

Abstract

Abstraction is a powerful form of domain knowledge that allows reinforcement-learning agents to cope with complex environments, but in most cases a human must supply this knowledge. In the absence of such prior knowledge or a given model, we propose an algorithm for the automatic discovery of state abstraction from policies learned in one domain for use in other domains that have similar structure. To this end, we introduce a novel condition for state abstraction in terms of the relevance of state features to optimal behavior, and we exhibit statistical methods that detect this condition robustly. Finally, we show how to apply temporal abstraction to benefit safely from even partial state abstraction in the presence of generalization error.

1 Introduction

Humans can cope with an unfathomably complex world due to their ability to focus on pertinent information while ignoring irrelevant detail. In contrast, most of the research into artificial intelligence relies on fixed problem representations. Typically, the researcher must engineer a feature space rich enough to allow the algorithm to find a solution but small enough to achieve reasonable efficiency. In this paper we consider the *reinforcement learning* (RL) problem, in which an agent must learn to maximize rewards in an initially unknown, stochastic environment [Sutton and Barto, 1998]. The agent must consider enough aspects of each situation to inform its choices without spending resources worrying about minutiae. In practice, the complexity of this state representation is a key factor limiting the application of standard RL algorithms to real-world problems.

One approach to adjusting problem representation is *state abstraction*, which maps two distinct states in the original formulation to a single abstract state if an agent should treat the two states in exactly the same way. The agent can still learn optimal behavior if the Markov decision process (MDP) that formalizes the underlying domain obeys certain conditions: the relevant states must share the same local behavior in the abstract state space [Dean and Givan, 1997; Ravindran and Barto, 2003]. However, this prior research only applies in a planning context, in which the MDP model is given, or if the user manually determines that the conditions hold and supplies the corresponding state abstraction to the RL algorithm.

We propose an alternative basis to state abstraction that is more conducive to automatic discovery. Intuitively, if it is possible to behave optimally while ignoring a certain aspect of the state representation, then an agent has reason to ignore that aspect during learning. Recognizing that discovering structure tends to be slower than learning an optimal behavior policy [Thrun and Schwartz, 1995], this approach suggests a knowledge-transfer framework, in which we analyze policies learned in one domain to discover abstractions that might improve learning in similar domains. To test whether abstraction is possible in a given region of the state space, we give two statistical methods that trade off computational and sample complexity.

We must take care when we apply our discovered abstractions, since the criteria we use in discovery are strictly weaker than those given in prior work on safe state abstraction. Transferring abstractions from one domain to another may also introduce generalization error. To preserve convergence to an optimal policy, we encapsulate our state abstractions in *temporal abstractions*, which construe sequences of primitive actions as constituting a single abstract action [Sutton *et al.*, 1999]. In contrast to previous work with temporal abstraction, we discover abstract actions intended just to simplify the state representation, not to achieve a certain goal state. RL agents equipped with these abstract actions thus learn when to apply state abstraction the same way they learn when to execute any other action.

In Section 2, we describe our first contribution, an alternative condition for state abstraction and statistical mechanisms for discovery. In Section 3, we describe our second contribution, an approach to discovering state abstractions and then encapsulating them within temporal abstractions. In Section 4, we present an empirical validation of our approach. In Section 5, we discuss related work, and in Section 6, we conclude.

2 Policy irrelevance

2.1 Defining irrelevance

First we recapitulate the standard MDP notation. An MDP $\langle S, A, P, R\rangle$ comprises a finite set of states S, a finite set of actions A, a transition function $P : S \times A \times S \rightarrow [0,1]$, and a reward function $R : S \times A \rightarrow \mathbb{R}$. Executing an action a in a state s yields an expected immediate reward $R(s,a)$ and causes a transition to state s' with probability $P(s,a,s')$. A policy $\pi : S \rightarrow A$ specifies an action $\pi(s)$ for every state s and induces a value function $V^\pi : S \rightarrow \mathbb{R}$ that satisfies the Bellman equations $V^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} P(s,\pi(s),s')V^\pi(s')$, where $\gamma \in [0,1]$ is a discount factor for future reward that may be necessary to make the equations satisfiable. For every MDP at least one optimal policy π^* exists that maximizes the value function at every state simultaneously. We denote the unique optimal value function V^*. Many learning algorithms converge to optimal policies by estimating the optimal state-action value function $Q^* : S \times A \rightarrow \mathbb{R}$, with $Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s,\pi(s),s')V^*(s')$.

Without loss of generality, assume that the state space is the cartesian product of (the domains of) n state variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ and m state variables $\mathcal{Y} = \{Y_1, \ldots, Y_m\}$, so $S = X_1 \times \cdots \times X_n \times Y_1 \times \cdots \times Y_m$. We write $[s]_\mathcal{X}$ to denote the projection of s onto $\mathcal{X}$ and $s' \models [s]_\mathcal{X}$ to denote that s' agrees with s on every state variable in $\mathcal{X}$. Our goal is to determine when we can safely abstract away $\mathcal{Y}$. In this work we introduce a novel approach to state abstraction called *policy irrelevance*. Intuitively, if an agent can behave optimally while ignoring a state variable, then we should abstract that state variable away. More formally, we say that $\mathcal{Y}$ is policy irrelevant at s if some optimal policy specifies the same action for every s' such that $s' \models [s]_\mathcal{X}$:

$$\exists_a \forall_{s' \models [s]_\mathcal{X}} \forall_{a'} \; Q^*(s',a) \geq Q^*(s',a'). \quad (1)$$

If $\mathcal{Y}$ is policy irrelevant for every s, then $\mathcal{Y}$ is policy irrelevant for the entire domain.

Consider the illustrative toy domain shown in Figure 1. It has just four nonterminal states described by two state variables, X and Y. It has two deterministic actions, represented by the solid and dashed arrows respectively. When $X = 1$, both actions terminate the episode but determine the final reward, as indicated in the figure. This domain has two optimal policies, one of which we can express without Y: take the solid arrow when $X = 0$ and the dashed arrow when $X = 1$. We thus say that Y is policy irrelevant across the entire domain.

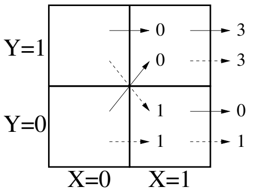

Figure 1: A domain with four nonterminal states and two actions. When $X = 1$ both actions transition to an absorbing state, not shown.

Note however that we cannot simply aggregate the four states into two states. As McCallum pointed out, the state distinctions sufficient to represent the optimal policy are not necessarily sufficient to learn the optimal policy [McCallum, 1995]. In this example, observe that if we treat $X = 1$ as a single abstract state, then in $X = 0$ we will learn to take the dashed arrow, since it transitions to the same abstract state as the solid arrow but earns a greater immediate reward. We demonstrate how to circumvent this problem while still benefitting from the abstraction in Section 3.2.

2.2 Testing irrelevance

If we have access to the transition and reward functions, we can evaluate the policy irrelevance of a candidate set of state variables $\mathcal{Y}$ by solving the MDP using a method, such as policy iteration, that can yield the set of optimal actions $\pi^*(s) \subseteq A$ at each state s. Then $\mathcal{Y}$ is policy irrelevant at s if some action is in each of these sets for each assignment to $\mathcal{Y}$: $\bigcap_{s' \models [s]_\mathcal{X}} \pi^*(s') \neq \emptyset$.

However, testing policy irrelevance in an RL context is trickier if the domain has more than one optimal policy, which is often the case for domains that contain structure or symmetry. Most current RL algorithms focus on finding a single optimal action at each state, not all the optimal actions. For example, Figure 2 shows the Q values learned from a run of Q-learning,[1] a standard algorithm that employs stochastic approximation to learn Q^* [Watkins, 1989]. Even though the state variable Y is actually policy irrelevant, from this data we would conclude that an agent must know the value of Y to behave optimally when $X = 1$. In this trial we allowed the learning algorithm enough exploration to find an optimal policy but not enough to converge to accurate Q values for every state-action pair. We argue that this phenomenon is quite common in practical applications, but even with sufficient exploration the inherent stochasticity of the domain may disguise state variable irrelevance. We the propose two methods for detecting policy irrelevance in a manner robust to this variability.

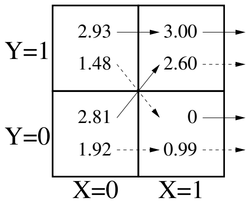

Figure 2: The domain of Figure 1 with some learned Q values.

Statistical hypothesis testing

Hypothesis testing is a method for drawing inferences about the true distributions underlying sample data. In this section, we describe how to apply this method to the problem of inferring policy irrelevance. To this end, we interpret an RL algorithm's learned value $Q(s,a)$ as a random variable, whose distribution depends on both the learning algorithm and the domain. Ideally, we could then directly test that hypothesis (1) holds, but we lack an appropriate test statistic. Instead, we assume that for a reasonable RL algorithm, the means of these distributions share the same relationships as the corresponding true Q values: $\overline{Q}(s,a) \geq \overline{Q}(s,a') \equiv Q^*(s,a) \geq Q^*(s,a')$. We then test propositions of the form

$$\overline{Q}(s,a) \geq \overline{Q}(s,a'), \quad (2)$$

using a standard procedure such as a one-sided paired t-test or Wilcoxon signed ranks test [Degroot, 1986]. These tests output for each hypothesis (2) a significance level $p_{s,a,a'}$. If $\overline{Q}(s,a) = \overline{Q}(s,a')$ then this value is a uniformly random number from the interval $(0,1)$. Otherwise, $p_{s,a,a'}$ will tend towards 1 if hypothesis (2) is true and towards 0 if it is false. We combine these values in a straightforward way to obtain a confidence measure for hypothesis (1):

$$p = \max_a \min_{s' \models [s]_\mathcal{X}} \min_{a' \neq a} p_{s',a,a'}. \quad (3)$$

[1] No discounting, learning rate 0.25, Boltzmann exploration with starting temperature 50, cooling rate 0.95, for 50 episodes

Figure 3 shows these p values for our toy domain. To obtain the data necessary to run the test, we ran 25 independent trials of Q-learning. We used the Wilcoxon signed-ranks test, which unlike the t-test does not assume that $Q(s, a)$ is Gaussian. In Figure 3a we see "random" looking values, so we accept that Y is policy irrelevant for both values of X. In Figure 3b we see values very close to 0, so we must reject our hypothesis that X is policy irrelevant for either value of Y. In our work, we use 0.05 as a threshold for rejecting hypothesis (1). If p exceeds 0.05 for every s, then $\mathcal{Y}$ is irrelevant across the entire domain. In practice this number seems quite conservative, since in those cases when the hypothesis is false we consistently see p values orders of magnitude smaller.

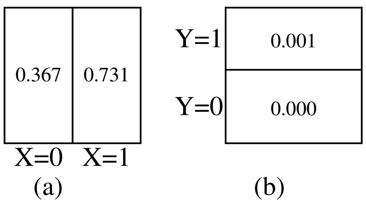

Figure 3: The value of p for each of the two abstract states when testing the policy irrelevance of (a) Y and (b) X.

Monte Carlo simulation

The hypothesis testing approach is computationally efficient, but it requires a large amount of data. We explored an alternative approach designed to conserve experience data when interaction with the domain is expensive. We draw upon work in Bayesian MDP models [Dearden *et al.*, 1999] to reason more directly about the distribution of each $Q(s, a)$. This technique regards the successor state for a given state-action pair as a random variable with an unknown multinomial distribution. For each multinomial distribution, we perform Bayesian estimation, which maintains a probability distribution over multinomial parameters. After conditioning on state transition data from a run of an arbitrary RL algorithm, the joint distribution over the parameters of these multinomials gives us a distribution over transition functions. The variance of this distribution goes to 0 and its mean converges on the true transition function as the amount of data increases.[2]

Once we have a Bayesian model of the domain, we can apply Monte Carlo simulation to make probabilistic statements about the Q values. We sample MDPs from the model and solve them[3] to obtain a sample for each Q value. Then we can estimate the probability that $Q^*(s, a) \geq Q^*(s, a')$ holds as the fraction of the sample for which it holds. We use this probability estimate in the same way that we used the significance levels in the hypothesis testing approach to obtain a confidence measure for the policy irrelevance of $\mathcal{Y}$ at some s:

$$p = \max_a \min_{s' \models [s]_\mathcal{X}} \min_{a' \neq a} \Pr(Q^*(s', a) \geq Q^*(s', a')). \quad (4)$$

This method seems to yield qualitatively similar results to the hypothesis testing method. We almost always obtain a value of $p = 0$ for cases in which $\mathcal{Y}$ actually is relevant; we obtain a value near 1 when only one action is optimal; we obtain a uniformly random number in $(0, 1)$ when more than one action is optimal. Although it achieves similar results using less data, this method incurs a higher computational cost due to the need to solve multiple MDPs.[4]

3 Abstraction discovery

3.1 Discovering irrelevance

The techniques described in Section 2.2 both involve two stages of computation. In the first stage, they acquire samples of state-action values, either by solving the task repeatedly or by solving sampled MDPs repeatedly. In the second stage, they use this data to test the relevance of arbitrary sets of state variables at arbitrary states. Any one of these tests in the second stage is very cheap relative to the cost of the first stage, but the number of possible tests is astronomical. We must limit both the sets of state variables that we test and the states at which we test them.

First consider the sets of state variables. It is straightforward to prove that if $\mathcal{Y}$ is policy irrelevant at s, then every subset of $\mathcal{Y}$ is also policy irrelevant at s.[5] A corollary is that we only need to test the policy irrelevance of $\{Y_1, \ldots, Y_k\}$ at s if both $\{Y_1, \ldots, Y_{k-1}\}$ and $\{Y_k\}$ are policy irrelevant at s. This observation suggests an inductive procedure that first tests each individual state variable for policy irrelevance and then tests increasingly larger sets only as necessary. This inductive process will continue only so long as we find increasingly powerful abstractions.

We can afford to test each state variable at a given state, since the number of variables is relatively small. In contrast, the total number of states is quite large: exponential in the number of variables. We hence adopt an heuristic approach, which tests for policy irrelevance only at those states visited on some small number of trajectories through the task. For these states, we then determine what sets of state variables are policy irrelevant, as described above. For each set of state variables we can then construct a binary classification problem with a training set comprising the visited states. An appropriate classification algorithm then allows us to generalize the region over which each set of state variables is policy irrelevant. Note that in Section 3.2 we take steps to ensure that the classifiers' generalization errors do not lead to the application of unsafe abstractions.

3.2 Exploiting irrelevance

Section 3.1 describes how to represent as a learned classifier the region of the state space where a given set of state variables is policy irrelevant. A straightforward approach to state abstraction would simply aggregate together all those

[2] It is also possible to build a Bayesian model of the reward function, but all the domains that we have studied use deterministic rewards.

[3] We use standard value iteration.

[4] We ameliorate this cost somewhat by initializing each MDP's value function with the value function for the maximum likelihood MDP, as in [Strens, 2000].

[5] The converse is not necessarily true. Suppose we duplicate an otherwise always relevant state variable. Then each copy of the state variable is always policy irrelevant given the remainder of the state representation, but the pair of them is not.

states in this region that differ only on the irrelevant variables. However, this approach may prevent an RL algorithm from learning the correct value function and therefore the optimal policy. In Section 2.1 we gave a simple example of such an abstraction failure, even with perfect knowledge of policy irrelevance. Generalizing the learned classifier from visited states in one domain to unvisited states in a similar domain introduces another source of error. A solution to all of these problems is to encapsulate each learned state abstraction inside a temporal abstraction. In particular, we apply each state space aggregation only inside an option [Sutton *et al.*, 1999], which is an abstract action that may persist for multiple time steps in the original MDP.

Formally, for a set of state variables $\mathcal{Y}$ that is policy irrelevant over some $S' \subseteq S$, we construct an option $o = \langle \pi, \mathcal{I}, \beta \rangle$, comprising an option policy $\pi : [S']_{\mathcal{X}} \to A$, an initiation set $\mathcal{I} \subset S$, and a termination condition $\beta : S \to [0, 1]$. Once an agent executes an option o from a state in $\mathcal{I}$, it always executes primitive action $\pi(s)$ at each state s, until terminating with probability $\beta(s)$. We set $\mathcal{I} = S'$ and $\beta(s) = 0.01$ for $s \in \mathcal{I}$ and $\beta(s) = 1$ otherwise.[6] Since $\mathcal{Y}$ is policy irrelevant over S', we may choose an option policy π equal to the projection onto $[S']_{\mathcal{X}}$ of an optimal policy for the original MDP. An agent augmented with such options can behave optimally in the original MDP by executing one of these options whenever possible.

Although we believe that the discovery of this structure is interesting in its own right, its utility becomes most apparent when we consider transferring the discovered options to novel domains, for which we do not yet have access to an optimal policy. To transfer an option to a new domain, we simply copy the initiation set and termination condition. This straightforward approach suffices for domains that share precisely the same state space as the original domain. Even when the state space changes, our representation of $\mathcal{I}$ and β as a learned classifier gives us hope for reasonable generalization. We can also copy the option policy π, if we expect the optimal behavior from the original domain to remain optimal in the new domain.

In this paper we assume only that the policy irrelevance remains the same. We thus relearn the option policy concurrently with the learning of the high-level policy, which chooses among the original primitive actions and the discovered options. For each option, we establish an RL subproblem with state space $[\mathcal{I}]_{\mathcal{X}}$ and the same action space A. Whenever an option terminates in a state s, we augment the reward from the environment with a pseudoreward equal to the current estimate of the optimal high-level value function evaluated at s. We therefore think of the option not as learning to achieve a subgoal but learning to behave while ignoring certain state variables. In other words, the option adopts the goals of the high-level agent, but learns in a reduced state space.

Since each option is just another action for the high-level agent to select, RL algorithms will learn to disregard options as suboptimal in those states where the corresponding abstractions are unsafe. The options that correspond to safe state abstractions join the set of optimal actions at each appropriate state. The smaller state representation should allow the option policies to converge quickly, so RL algorithms will learn to exploit these optimal policy fragments instead of uncovering the whole optimal policy the hard way. We illustrate this process in the next section.

4 Results

We use Dietterich's Taxi domain [Dietterich, 2000], illustrated in Figure 4, as the setting for our work. This domain has four state variables. The first two correspond to the taxi's current position in the grid world. The third indicates the passenger's current location, at one of the four labeled positions (Red, Green, Blue, and Yellow) or inside the taxi. The fourth indicates the labeled position where the passenger would like to go. The domain therefore has $5 \times 5 \times 5 \times 4 = 500$ possible states. At each time step, the taxi may move north, move south, move east, move west, attempt to pick up the passenger, or attempt to put down the passenger. Actions that would move the taxi through a wall or off the grid have no effect. Every action has a reward of -1, except illegal attempts to pick up or put down the passenger, which have reward -10. The agent receives a reward of +20 for achieving a goal state, in which the passenger is at the destination (and not inside the taxi). In this paper, we consider the stochastic version of the domain. Whenever the taxi attempts to move, the resulting motion occurs in a random perpendicular direction with probability 0.2. Furthermore, once the taxi picks up the passenger and begins to move, the destination changes with probability 0.3.

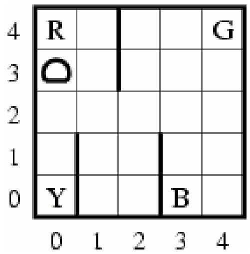

Figure 4: The Taxi domain.

This domain's representation requires all four of its state variables in general, but it still affords opportunity for abstraction. In particular, note that the passenger's destination is only relevant once the agent has picked up the passenger. We applied the methodology described in Sections 2 and 3 to the task of discovering this abstraction, as follows. First, we ran 25 independent trials of Q-learning to obtain samples of Q^*. For each trial, we set the learning rate $\alpha = 0.25$ and used ϵ-greedy exploration with ϵ = 0.1. Learning to convergence required about 75000 time steps for each trial. This data allows us to compute the policy irrelevance of any state variable at any state. For example, consider again the passenger's destination. To demonstrate the typical behavior of the testing procedure, we show in Figure 5a the output for every location in the domain, when the passenger is waiting at the upper left corner (the Red landmark), using the Wilcoxon signed-ranks test. The nonzero p values at every state imply that the passenger's destination is policy irrelevant in this case. Note that the values are extremely close to 1 whenever the agent has only one optimal action to get to the upper left corner, which the procedure can then identify confidently. The squares with intermediate values are precisely the states in which more than one optimal action exists. Now consider

[6]The nonzero termination probability for $s \in \mathcal{I}$ serves as a probabilistic timeout to escape from bad abstractions.

Figure 5b, which shows the output of the same test when the passenger is inside the taxi. The p values are extremely close to 0 in every state except for the four at the bottom middle, where due to the layout of the domain the agent can always behave optimally by moving north.

0.9999	0.9999	0.9999	0.9919	0.3784
0.9999	0.3188	0.9999	0.4731	0.5799
0.9999	0.1766	0.9999	0.9999	0.9999
0.9999	0.9999	0.5799	0.9999	0.4095
0.9999	0.9994	0.7940	0.9999	0.7703

(a)

0.0000	0.0000	0.0001	0.0002	0.0000
0.0000	0.0000	0.0004	0.0003	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.4412	0.4946	0.0000	0.0003
0.0000	0.9171	0.6518	0.0000	0.0000

(b)

Figure 5: The results of the Wilcoxon signed-ranks test for determining the policy irrelevance of the passenger's destination in the Taxi domain. We show the result of the test for each possible taxi location for (a) a case when the passenger is not yet in the taxi and (b) the case when the passenger is inside the taxi.

Rather than compute the outcome of the test for every subset of state variables at every state, we followed the approach described in Section 3.1 and sampled 20 trajectories from the domain using one of the learned policies. We tested each individual state variable at each state visited, again using the hypothesis testing approach. We created a binary classification problem for each variable, using the visited states as the training set. For the positive examples, we took each state at which the hypothesis test returns a p value above the conservative threshold of 0.05. Finally, we applied a simple rule-learning classifier to each problem: the Incremental Reduced Error Pruning (IREP) algorithm, as described in [Cohen, 1995]. A typical set of induced rules follows:

1. Taxi's x-coordinate:
   (a) $y = 1 \wedge$ passenger in taxi $\wedge$ destination Red $\Rightarrow$ *policy irrelevant*
   (b) otherwise, *policy relevant*
2. Taxi's y-coordinate:
   (a) $x = 4 \wedge$ passenger in taxi $\Rightarrow$ *policy irrelevant*
   (b) otherwise, *policy relevant*
3. Passenger's destination:
   (a) passenger in taxi $\Rightarrow$ *policy relevant*
   (b) otherwise, *policy irrelevant*
4. Passenger's location *and* destination
   (a) $(x = 1 \wedge y = 2) \vee (x = 1 \wedge y = 1)$ $\Rightarrow$ *policy irrelevant*
   (b) otherwise, *policy relevant*

The sets of state variables not mentioned either had no positive training examples or induced an empty rule set, which classifies the state variables as relevant at every state. Rule set 3 captures the abstraction that motivated our analysis of this domain, specifying that the passenger's destination is policy relevant only when the passenger is in the taxi. The other three rules classify state variables as usually relevant, except in narrow cases. For example, rule 1a holds because the Red destination is in the upper half of the map, $y = 1$ specifies that the taxi is in the lower half, and all the obstacles in this particular map are vertical. Rule 2a is an example of an overgeneralization. When holding the passenger on the rightmost column, it is usually optimal just to go left, unless the passenger wants to go the Green landmark in the upper-right corner.

We tested the generalization performance of these learned abstractions on 10×10 instances of the Taxi domain with randomly generated obstacles, running both horizontally and vertically. We placed one landmark near each corner and otherwise gave these domains the same dynamics as the original. Each abstraction was implemented as an option, as discussed in Section 3.2. Since the locations of the landmarks moved, we could not have simply transferred option policies from the original Taxi domain. In all our experiments, we used Q-learning with ϵ-greedy exploration[7] to learn both the option policies and the high-level policy that chose when to apply each option and thus each state abstraction.[8] To improve learning efficiency, we added off-policy training [Sutton *et al.*, 1999] as follows. Whenever a primitive action a was executed from a state s, we updated $Q(s, a)$ for the high-level agent as well as for every option that includes s in its initiation set. Whenever an option o terminated, we updated $Q(s, o)$ for every state s visited during the execution of o. Each state-action estimate in the system therefore received exactly one update for each timestep the action executed in the state. Figure 6 compares the learning performance of this system to a Q-learner without abstraction. The abstractions allowed the experimental Q-learner to converge much faster to an optimal policy, despite estimating a strict superset of the parameters of the baseline Q-learner.

5 Related work

Our approach to state abstraction discovery bears a strong resemblance to aspects of McCallum's U-tree algorithm [McCallum, 1995], which uses statistical hypothesis testing to determine what features to include in its state representation. U-tree is an online instance-based algorithm that adds a state variable to its representation if different values of the variable predict different distributions of expected future reward. The algorithm computes these distributions of values in part from the current representation, resulting in a circularity that prevents it from guaranteeing convergence on an optimal state abstraction. In contrast, we seek explicitly to preserve optimality.

Our encapsulation of partial state abstractions into options is inspired by Ravindran and Barto's work on MDP homomorphisms [Ravindran and Barto, 2003] and in particular their discussion of partial homomorphisms and relativized options. However, their work focuses on developing a more

[7] $\epsilon = 0.1$ and $\alpha = 0.25$

[8] In general, SMDP Q-learning is necessary to learn the high-level policy, since the actions may last for more than one time step. However, this algorithm reduces to standard Q-learning in the absence of discounting, which the Taxi domain does not require.

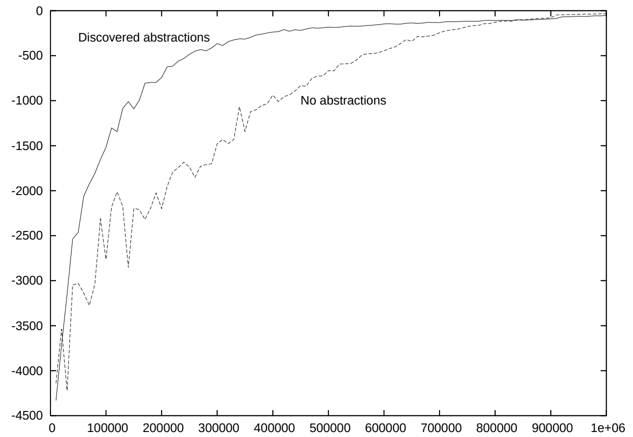

Figure 6: The average reward per episode earned by agents with learned abstractions encapsulated as options and only primitive actions, respectively, on a 10×10 version of the Taxi domain. The reward is averaged over 10000-step intervals. The results are the average of 25 independent trials.

agile framework for MDP minimization, not on the discovery of abstractions in an RL context.

Our work is also related to recent research into the automatic discovery of temporal abstractions [Özgür Şimşek and Barto, 2004; Mannor *et al.*, 2004], usually in the options framework. These techniques all seek to identify individual subgoal states that serve as chokepoints between well-connected clusters of states or that otherwise facilitate better exploration of environments. Our usage of options suggests an alternative purpose for temporal abstractions: to enable the safe application of state abstractions. Note that we can construe our definition of policy irrelevance as a statement about when a single reusable subtask *could* have contributed to several parts of an optimal policy.

The connection to hierarchical RL suggests the recursive application of our abstraction discovery technique to create hierarchies of temporal abstractions that explicitly facilitate state abstractions, as in MAXQ task decompositions [Dietterich, 2000]. This possibility highlights the need for robust testing of optimal actions, since each application of our method adds new potentially optimal actions to the agent. However, we leave the development of these ideas to future work.

6 Conclusion

This paper addressed the problem of discovering state abstractions automatically, given only prior experience in a similar domain. We defined a condition for abstraction in terms of the relevance of state variables for expressing an optimal policy. We described two statistical methods for testing this condition for a given state and set of variables. One method applies efficient statistical hypothesis tests to Q-values obtained from independent runs of an RL algorithm. The other method applies Monte Carlo simulation to a learned Bayesian model to conserve experience data. Then we exhibited an efficient algorithm to use one of these methods to discover what sets of state variables are irrelevent over what regions of the state space. Finally, we showed that encapsulating these learned state abstractions inside temporal abstractions allows an RL algorithm to benefit from the abstractions while preserving convergence to an optimal policy.

Acknowledgments

We would like to thank Greg Kuhlmann for helpful comments and suggestions. This research was supported in part by NSF CAREER award IIS-0237699 and DARPA grant HR0011-04-1-0035.

References

[Cohen, 1995] William W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[Dean and Givan, 1997] Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

[Dearden *et al.*, 1999] Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.

[Degroot, 1986] Morris H. Degroot. *Probability and Statistics*. Addison-Wesley Pub Co, 2nd edition, 1986.

[Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Mannor *et al.*, 2004] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 560–567, 2004.

[McCallum, 1995] Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1995.

[Özgür Şimşek and Barto, 2004] Özgür Şimşek and Andrew G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 751–758, 2004.

[Ravindran and Barto, 2003] Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[Strens, 2000] Malcolm Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 943–950, 2000.

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

[Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

[Thrun and Schwartz, 1995] Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*, 1995.

[Watkins, 1989] Watkins. *Learning From Delayed Rewards*. PhD thesis, University of Cambridge, England, 1989.