Automatic construction of temporally extended actions for MDPs using bisimulation metrics

Pablo Samuel Castro and Doina Precup

School of Computer Science
McGill University
{pcastr,dprecup}@cs.mcgill.ca

Abstract. Temporally extended actions are usually effective in speeding up reinforcement learning. In this paper we present a mechanism for automatically constructing such actions, expressed as options [Sutton et al., 1999], in a finite Markov Decision Process (MDP). To do this, we compute a bisimulation metric [Ferns et al., 2004] between the states in a small MDP and the states in a large MDP, which we want to solve. The *shape* of this metric is then used to completely define a set of options for the large MDP. We demonstrate empirically that our approach is able to improve the speed of reinforcement learning, and is generally not sensitive to parameter tuning.

1 Introduction

Temporally extended actions are a well-studied approach for speeding up stochastic planning and reinforcement learning [Sutton et al., 1999; Parr and Russell, 1998; Dietterich, 2000; Barto and Mahadevan, 2003]. Much attention has been devoted to the problem of learning such courses of action (e.g. [McGovern and Barto, 2001; Mannor et al., 2004; Šimšek et al., 2005; Zang et al., 2009; Konidaris et al., 2010]). This is still a challenging problem, as it goes back to the fundamental problem of intelligence: how can one learn useful *representations* of an environment from data?

In this paper we propose a new approach to the problem of learning extended actions, which is based on similar intuitions to learning by analogy, an approach used in cognitive architectures such as ACT-R [Anderson, 1996] and SOAR [Laird et al., 2011]. The idea is to find similarities between a new problem and older, solved problems, and adapt the previous solution to the new situation. Here, we use small example systems, compute the similarity of states from such systems to states in larger problems, then use this similarity measurement in order to produce new temporally extended actions for the large system. It is important to note that, unlike in other research, such as transfer learning (see [Taylor and Stone, 2009] for a survey), we do not attempt to provide an entire solution to the new system. Instead, "pieces" of knowledge are constructed, and they will be used further in learning and planning for the new problem. This is a less daunting task, as we will illustrate in the paper; as a result, the solved problem and the new problem could actually be quite different, and the approach still succeeds in providing useful representations.

We develop this idea in the context of the *options* framework for modeling temporally extended actions in reinforcement learning [Precup, 2000; Sutton et al., 1999]. We use bisimulation metrics [Ferns et al., 2004] to relate states in the two different problems, from the point of view of behavior similarity. Previous work [Castro and Precup, 2010] explored the use of bisimulation metrics for transferring the optimal policy from a small system to a large system. Here, we instead identify subsets of states in which an option should be defined, based on the "shape" of the bisimulation metric. We use the metric to define all aspects of the option: the initiation set, the termination probabilities, and its internal policy. These options can then be used for reinforcement learning. In the experiments, we use these options instead of primitive actions, although they could also be used in addition to the primitives. We use a sampling-based approximant of the bisimulation metric [Ferns et al., 2006] to compute the distances, removing the requirement that a model of the system be known beforehand.

This paper is organized as follows. In section 2 we present the necessary background and introduce the notation. In section 3 we describe the procedure for constructing options. We evaluate empirically the quality of the options obtained in section 4. We present concluding remarks and avenues of future work in section 5.

2 Background and Notation

2.1 MDPs and Q-learning

A Markov Decision Process (MDP) is defined as a 4-tuple $\langle S, A, P, R\rangle$ where S is a finite set of states, A is a finite set of actions available from each state, $P : S \times A \rightarrow Dist(S)$[1] specifies the probabilistic transition function and $R : S \times A \rightarrow \mathbb{R}$ is the reward function. A policy $\pi : S \rightarrow Dist(A)$ indicates the action choice at each state. The value of a state $s \in S$ under a policy π is defined as $V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{\tau=1}^{\infty} \gamma^{\tau-1} r_{\tau} | s_0 = s\right]$, where r_{τ} is a random variable denoting the reward received at time step τ and $\gamma \in (0,1)$ is the discount factor. The optimal state-action value function $Q^* : S \times A \rightarrow \mathbb{R}$ gives the maximal expected return for each state-action pair, given that the optimal policy is followed afterwards. It obeys the following set of Bellman optimality equations: $Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s,a)(s') \max_{a' \in A} Q^*(s',a')$.

Q-learning is a popular reinforcement learning algorithm which maintains an estimate of Q^* based on samples. After performing action a from state s, landing in state s' and receiving reward r, the estimate for $Q(s,a)$ is updated as follows: $Q(s,a) = Q(s,a) + \alpha\left[r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)\right]$, where $\alpha \in (0,1)$ is a learning rate parameter. Watkins and Dayan [1992] proved that the above method converges to Q^* as long as all actions are repeatedly taken from all states (and some conditions on the learning rate α). To guarantee this, a simple strategy

[1] $Dist(X)$ is defined as the set of distributions over X

is choosing an action randomly with probability ϵ and choosing the action that appears best (according to the current estimates) the rest of the time; this is called ϵ-greedy exploration. For a more detailed treatment of these and other reinforcement learning algorithms, please see [Sutton and Barto, 1998].

2.2 Options

The options framework [Sutton et al., 1999; Precup, 2000] allows the definition of temporally extended actions in an MDP. Formally, an option o is a triple $\langle \mathcal{I}_o, \pi_o, \beta_o \rangle$, where $\mathcal{I}_o \subseteq S$ is the set of states where the option is available, $\pi_o : S \to Dist(A)$ is the option's policy and $\beta_o : S \to [0,1]$ is the probability of the option terminating at each state. When an option o is started in state $s \in \mathcal{I}_o$, the policy π_o is followed until the option is terminated, as dictated by β_o. The *model* of an option consists of the *discounted* transition probabilities $Pr(s'|s,o), \forall s, s' \in S$ and the expected reward received while executing the option: $\mathfrak{R}(s,o)$.

More formally, cf. [Sutton et al., 1999; Precup, 2000], let $Pr(s',k|s,o)$ be the probability that option o terminates after k time steps in state s', after being started in s. Then, $Pr(s'|s,o)$ is defined as $Pr(s'|s,o) = \sum_{k=1}^{\infty} Pr(s',k|s,o)\gamma^k$. Note that because the transition model of an option incorporates the discount factor, it actually consists of *sub-probabilities* (*i.e.* $\sum_{s'} Pr(s'|s,o) < 1$).

Similarly, the reward model of an option is defined as:
$\mathfrak{R}(s,o) = \mathbb{E}^o \left[\sum_{\tau=1}^{k} \gamma^{\tau-1} r_\tau | s_0 = s \right]$, where k is the time at which the option terminates.

With these model definitions, planning and reinforcement learning algorithms can be defined for an MDP with given options in a manner very similar to algorithms involving just primitive actions. Options have also been shown to speed up learning and planning in application domains, especially in robotics (e.g. [Stone et al., 2005]).

A significant amount of research has been devoted to methods for learning options (and temporal abstractions in general). Much of the existing work is based on the notion of *subgoals*, i.e. important states that may be beneficial to reach. Subgoals can be identified based on a learned model of the environment, using graph theoretic techniques [Mannor et al., 2004; Šimšek et al., 2005], or based on trajectories through the system, without learning a model [McGovern and Barto, 2001; Stolle and Precup, 2002]. Some existing work targets specifically the problem of learning subgoals from trajectories when the state space is factored and state abstractions need to be acquired at the same time as options [Jonsson and Barto, 2006; Mehta et al., 2008; Zang et al., 2009; Mugan and Kuipers, 2009]. Other work is focused on learning options from demonstrations [Konidaris et al., 2010]. The notion of subgoals is conceptually attractive, as it creates abstractions that are easy to interpret, but it is also somewhat brittle: options defined by subgoals may not be useful, if the subgoals are not well identified. Comanici and Precup [2010] define an option construction algorithm where the general probabilities of termination are learned instead. This is also the approach we take here, but the learning algorithm is not gradient-based, as in their case.

The class of option construction methods most related to our approach is based on MDP homomorphisms [Ravindran and Barto, 2003; Wolfe and Barto, 2006; Soni and Singh, 2006; Sorg and Singh, 2009]. In this case, states from one MDP are mapped into states from a different MDP, based on the transition dynamics and reward functions. The existing work attempts to learn such mappings from data; Taylor et al. [2009] relate the notion of approximate MDP homomorphisms to a notion of MDP similarity called lax bisimulation, which allows for the construction of approximate homomorphisms with provable guarantees. Our approach builds on their metrics (as detailed below). Note also that, in contrast with existing work, we will construct *all* parts of the option (initiation set, policy and termination conditions).

2.3 Bisimulation metrics

Bisimulation captures behavioral equivalence between states in an MDP [Givan et al., 2003]. Roughly speaking, it relates two states when they have equal immediate rewards and equivalent dynamics.

Definition 1. *An equivalence relation E is a bisimulation relation if for any $s, t \in S$, sEt implies that for all $a \in A$ (i): $R(s,a) = R(t,a)$; and (ii): for all $C \in S/_E$, $\sum_{s' \in C} P(s,a)(s') = \sum_{s' \in C} P(t,a)(s')$, where $S/_E$ is the set of all equivalence classes in S w.r.t. E. Two states $s, t \in S$ are called bisimilar, denoted $s \sim t$, if there exists a bisimulation relation E such that sEt.*

Ferns et al. [2004] defined bisimulation metrics, a quantitative analogue of bisimulation relations. A metric d is said to be a bisimulation metric if for any $s, t \in S$, $d(s,t) = 0 \Leftrightarrow s \sim t$.

The bisimulation metric is based on the Kantorovich probability metric $T_K(d)(P,Q)$, where d is a pseudometric[2] on S and P, Q are two state distributions. The Kantorovich probability metric is defined as a linear program, which intuitively computes the cost of "converting" P into Q under metric d. The dual formulation of the problem is an instance of the minimum cost flow (MCF) problem, for which there are many efficient algorithms. Ferns et al. [2004] proved that the functional $F(d)(s,t) = \max_{a \in A} (|R(s,a) - R(t,a)| + \gamma T_K(d)(P(s,a), P(t,a))$ has a greatest fixed point, $d_\sim$, and $d_\sim$ is a bisimulation metric. It can be computed to a desired degree of accuracy δ by iteratively applying F for $\left\lceil \frac{\ln \delta}{\ln \gamma} \right\rceil$ steps. However, each step involves the computation of the Kantorovich metric for all state-state-action triples, which requires a model and has worst case running time of $O(|S|^3 \log |S|)$. We will refer to this bisimulation metric as the *exact metric*. Ferns et al. [2006] introduced a more efficient way of computing the metric by replacing the MCF problem with a weighted matching problem. The state probability distributions P and Q are estimated using statistical samples. More precisely, let $X_1, X_2, \ldots, X_N$ and $Y_1, Y_2, \ldots, Y_N$ be N points independently sampled from P and Q, respectively. The *empirical distribution* P_N is defined as:

[2] A pseudometric is similar to a metric but $d(x,y) = 0 \not\Rightarrow x = y$

$P_N(s) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{X_i=s}, \forall s$, where $\mathbb{1}$ is the indicator function. Then, for any metric d, $T_K(d)(P_N, Q_N) = \min_\sigma \frac{1}{N}\sum_{i=1}^{N} d(X_i, Y_{\sigma(i)})$, where the minimum is taken over all permutations σ on N elements. This is an instance of the weighted assignment problem, which can be solved in $O(N^3)$ time. In this approach, one can control the computation cost by varying the number of samples taken. Ferns et al. [2006] proved that $\{T_K(d)(P_N, Q_N)\}$ converges to $T_K(d)(P, Q)$ almost surely, and the metric $d_{\sim,N}(s,t)$ computed using $T_K(d)(P_N, Q_N)$ converges to $d_\sim$. We refer to this approximant as the *sampled metric.*

A shortcoming of bisimulation relations and metrics as defined above is that the action sets of two states under comparison must be the same. Taylor et al. [2009] overcome this problem by introducing lax-bisimulation relations.

Definition 2. *An equivalence relation E is a lax-bisimulation relation if for any $s, t \in S$, sEt implies that $\forall a \in A, \exists b \in A$ such that (i): $R(s,a) = R(t,b)$ and vice-versa; and (ii): for all $C \in S/_E$, $\sum_{s' \in C} P(s,a)(s') = \sum_{s' \in C} P(t,b)(s')$ Two states s, t are called lax-bisimilar, denoted $s \sim_L t$, if there exists a lax-bisimulation relation E such that sEt.*

Taylor et al. [2009] also extend lax-bisimulation relations to lax-bisimulation metrics by first defining the metric $J(d)$ between state-action pairs, given any metric d, as follows:

$$J(d)((s,a),(t,b)) = |R(s,a) - R(t,b)| + \gamma T_K(d)(P(s,a), P(t,a)) \tag{1}$$

They show that the functional

$$F_L(d)(s,t) = \max \begin{pmatrix} \max_{a\in A} \min_{b \in A} J(d)((s,a),(t,b)), \\ \max_{b\in A} \min_{a \in A} J(d)((s,a),(t,b)) \end{pmatrix} \tag{2}$$

has a greatest fixed point d_L, and d_L is a lax-bisimulation metric. As for the original bisimulation metric, we can approximate T_K using statistical sampling.

The lax bisimulation metric (or its approximants) can be used to establish homomorphisms between MDPs. Castro and Precup [2010] used lax-bisimulation metrics to transfer an optimal policy from a small source MDP to a large target MDP. Their approach was to pair every state in the target MDP with its "nearest" source state. The optimal action is used in the source state, and its "best matching" action counterpart is used in the target.

3 Option construction

We are now ready to present our approach to automatically construct options using a small source MDP for which an optimal policy has been found. The intuition behind the approach is that we will try to define options in a new, large task based on identifying "patches" of states that are similar to some particular state from the small problem. Each state from the small problem, with its corresponding optimal action, will be translated into a full option in the larger task. Only states that are similar enough to the original one should be

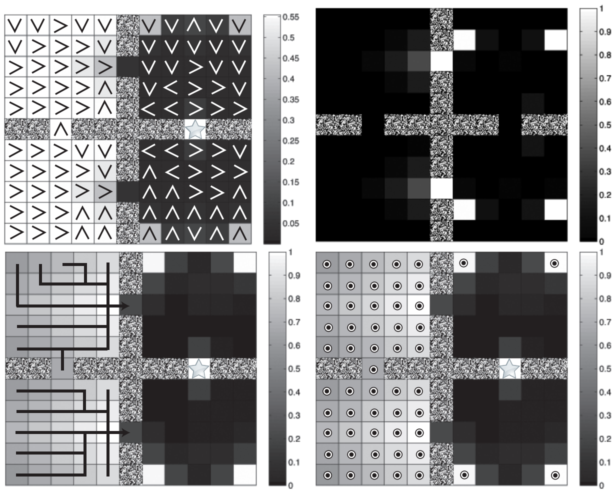

Fig. 1. Top left: Distances to leftmost source state and π_o; Top right: Computed values of β_o; Bottom left: Sum of discounted probabilities with chains used to find $\diamond$; Bottom right: Sum of discounted probabilities, dotted states are in $\mathcal{I}_o$.

included as part of the initiation set for the option. Large changes in similarity indicate large changes in either rewards or dynamics, and we use this as a cue that the option should be discontinued.

To assist in the description of the algorithms, we will use a running example in which a 7-state chain with a unique goal state in the rightmost position acts as a source system M_1, and a domain with four 5x5 rooms and 4 hallways is the target system M_2 (similar to [Sutton et al., 1999]); one of the hallways is the goal state, and the only positive reward is obtained when entering this goal; transitions succeed with 90% probability. We will focus the example on the option arising from the leftmost source state for simplicity. Note that the algorithm will construct one option corresponding to every state in the small source task, but in general one could decide to only construct options from a subset of states in the source. For each source state s we denote the resulting option as o_s. We first construct the policy π_{o_s}, then the termination function β_{o_s}, and finally the initialization set $\mathcal{I}_{o_s}$.

3.1 Constructing π_{o_s}

The policy for o_s is obtained initially by the method used in [Castro and Precup, 2010] for knowledge transfer. More precisely, given MDPs M_1 and M_2, the following metric is computed, where π^* is the optimal policy for MDP M_1:

$$J(d)(s,(t,b)) = |R_1(s,\pi^*(s)) - R_2(t,b)| \quad + \gamma T_K(d)(P_1(s,\pi^*(s)), P_2(t,a)) \tag{3}$$

Note that this is similar to equation (1), but the source state s is restricted to an optimal action given by $\pi^*(s)$. This was suggested in [Castro and Precup, 2010] to speed up the computation of the metric. Now, for any $t \in S_2$: $\pi_{o_s}(t) = \arg\min_{b \in A_2} J(d_L)(s, (t, b))$, where d_L is the lax-bisimulation metric. The top left image in Figure 1 displays the d_L distance between the states in the target system and the leftmost state in the source system, as well as the resulting policy. Note that the policy is defined, at the moment, for all states; it is optimal in the rooms on the left but sub-optimal in the rooms on the right. This is expected as we are comparing against the leftmost state in the source system. The policy generated by the source states that are closer to their goal is better for the rooms on the right than for those on the left (not shown for lack of space).

3.2 Constructing $\boldsymbol{\beta_{o_s}}$

From the left panel of Fig. 1, it is clear that the bisimulation distance has areas of smooth variation, among states that are at the same relative position to the goal (i.e. the source of reward) and areas of sharp variation around the hallways. Intuitively, if one considers the temporal neighborhood of states, areas in which the bisimulation distance varies drastically indicate a big behavioral change, which suggests a *boundary*. This is a similar idea to the change point detection of Konidaris et al. [2010], but here there is no human demonstration involved; the changes arise automatically from the similarities. These *areas of interest* will be used to define β_{o_s}.

To formalize this idea, for all state-action pairs (t, a) from M_2, we consider the most probable next state: $\bigcirc(t, a) = \arg\max_{t'} P_2(t, a)(t')$. For any state t' and option o_s, let $\zeta(t', o_s)$ be the set of states that have t' as their most probable next state:

$$\zeta(t', o_s) = \{t \in S_2 | \bigcirc (t, \pi_{o_s}(t)) = t'\}$$

The change in distance at each target state t' can be computed as follows:

$$\Delta(t', o_s) = \left| d_L(s, t') - \frac{1}{|\zeta(t', o_s)|} \sum_{t \in \zeta(t', o_s)} d_L(s, t) \right|$$

We now convert Δ into a termination probability:

$$\beta_{o_s}(t') = \frac{\Delta(t', o_s)}{\max_{t'' \in S_2} \Delta(t'', o_s)}$$

This definition ensures that options have a greater probability of terminating at states with the greatest change in the distance function. The top right panel of Fig. 1 displays the termination probabilities at each state.

3.3 Constructing the initiation set $\boldsymbol{\mathcal{I}_{o_s}}$

The question of learning good initiation sets is perhaps the least studied in the option construction literature. Intuitively, one desirable property of options is that they terminate within a reasonable amount of time, in order to allow a reconsideration of the course of action. Note that the model of an option provides this information readily, as its probabilities are discounted (*i.e.* $\sum_{t'\in S_2} Pr(t'|t,o) < 1$). The bottom panel of Fig. 1 displays the sum of discounted probabilities under the transferred option. Note that low values result from two different types of behavior: either states in which the policy of the option causes the agent to enter a cycle (i.e. by running into a wall), or states where the policy is good, but the option takes longer to terminate. We would like to distinguish between these two situations, and exclude from the initiation set mainly the states where the policy can lead to high-probability cycles. To achieve this, we define for all $t \in S_2$ and option o the most likely *ending state*, $\Diamond(t,o)$, in a recursive manner:

$$\Diamond(t,o) = \begin{cases} t & \text{If } \bigcirc(t,o) = t \\ \Diamond(\bigcirc(t,o),o) & \text{If } \sum_{t'\in S_2} Pr(t'|t,o) \le \sum_{t'\in S_2} Pr(t'|\bigcirc(t,o),o) \\ t & \text{Otherwise} \end{cases}$$

Thus, in order to determine $\Diamond(t,o)$ we follow the chain of most likely states (using the $\bigcirc$ operator) as long as the sum of discounted probabilities continues to increase (the second condition above). In the bottom left panel of figure 1 we display the chain of states followed to find the most likely states. States without a “path” are those states for which the same state is the most likely state (either by condition 1 or 3 above). We determine whether t should be in $\mathcal{I}_{o_s}$ based on the sum of discounted probabilities of the most likely ending state. Let τ be a threshold parameter. Formally, we define the initialization set as follows:

$$\mathcal{I}_{o_s} = \left\{ t \in S_2 \,\middle|\, \sum_{t'\in S_2} Pr(t'|\Diamond(t,o_s),o_s) \ge \tau \right\}$$

In the bottom right panel of figure 1 the dotted cells are the states in $\mathcal{I}$.

It is possible that some states will not be included in the initialization set for any option. If such a state is encountered, we add it to the $\mathcal{I}_{\hat{o}}$ for the option $\hat{o}$ that has the highest probability of termination when started from t. Note that we could also simply allow such states to use only primitive actions. We did not encounter any such states in our experiments.

4 Empirical evaluation

In this section we evaluate how effective the constructed options are in speeding up learning. Given an MDP, we use a small source MDP to construct options in the larger MDP, using a threshold parameter of $\tau = 0.7$[3] between for all environments. Then we perform standard SMDP Q-learning using *only* the options, and compare its results to Q-learning with just primitive actions. The Q-value functions were initialized to 0 in all domains. We use the sampled version of the

[3] The size of the initiation set is inversely proportional to τ. There was not much difference in performance when $\tau \in [0.6, 0.8]$.

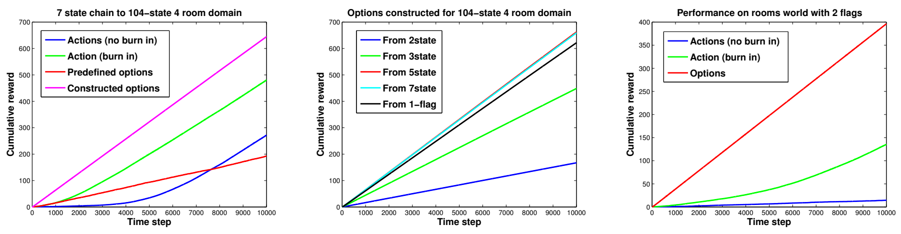

Fig. 2. Rooms world. Top left: No flags with 7-state chain as source. Top right: Various source domains with no flag target domain. Bottom: 2 flags added to target domain.

metric, as we do not assume that the environment model is known; all needed quantities for the construction algorithm are computed based on these samples. The sampling is performed once at the beginning of the algorithm, resulting in $N * |S| * |A|$ total samples. To allow a fair comparison with Q-learning, we grant Q-learning a "burn-in" phase, where it is allowed to take $N * |S| * |A|$ steps and update its Q-value function estimate. For all our experiments we set $N = 10$. As a baseline, we also compared with the performance when using the exact metric for option construction. The exact metric was not able to handle the larger domains, and in the domains where the exact metric was computed, the difference with the performance of the sampled metric was negligible. Thus, for clarity we have excluded the plots with the exact metric.

We allow the learning algorithms to run for 10,000 time steps and plot the cumulative reward received, averaged over 30 runs. Additionally, for the computation of the sampled metric we take 30 different sets of samples and average over the performance resulting from the respective metrics. The default settings are ϵ-greedy exploration policy with $\epsilon = 0.1$, and a learning rate of $\alpha = 0.1$ for all algorithms; in some experiments we vary these parameters to study their effect.

4.1 Rooms world

We use the 7-state chain and the rooms world domain described above as source and target, respectively. The starting state is fixed throughout the experiments. In the top left panel of figure 2 we display the performance of the different algorithms. We also ran learning using the four predefined options defined in [Castro and Precup, 2010], which are the options one would intuitively define for this domain. Despite this, the options constructed by our method have a better performance. This is most likely because our method automatically disables "bad" options for certain states. In the top right panel of figure 2, we compare the performance of using chains of varying sizes as the source domains, as well as a tiny 3-state domain with one flag (resulting in 6 overall states). As we can see, the performance is not affected much by changing source domains.

This behaviour was observed in all of the experiments presented below.

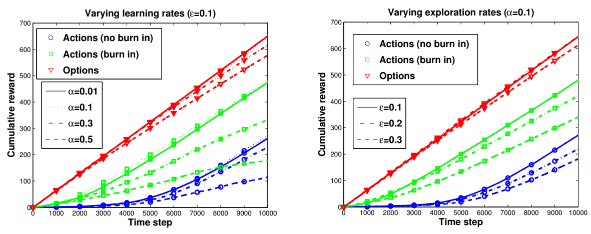

Fig. 3. Effect of varying parameters on rooms world.

To increase the complexity of the problem, we added two flags, which increases the state space to 416 states. Upon entering the goal state, the agent receives a reward of +1 for each flag picked up. The source system used for this problem is the one-flag system mentioned above. In the bottom panel of figure 2 we display the performance on the flagged domain. We note that the flagged domain was too large for the exact metric.

In Fig. 3 we evaluate the effect of varying parameters on the rooms world domain (without flags). Varying the learning and exploration rates has little effect on our method, which suggests that the constructed options are nearly optimal. This behavior was observed in all the problems discussed below.

So far we have been using the same reward "structure" for the source and target domain (*i.e.* a reward of 1 is received upon entering the goal). A natural question is whether our method is sensitive to differences in reward. In figure 4 it can be seen that this is not the case. We modified the target task so a reward of 5 is received upon entering the goal, and varied the goal reward in the source domain. In the right panel, we added a penalty of 0.1 in the target domain for each step that did not end in a goal state. Learning using the constructed options yields much better results than learning with primitive actions. It is interesting to note that the performance is improved when we add a penalty to the target domain. This is due to the fact that bisimulation distinguishes first based on differences in reward, so adding a non-zero reward at each state provides the bisimulation metric with more information.

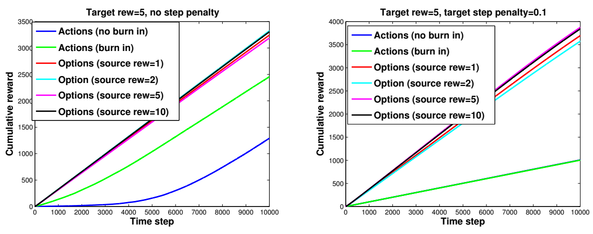

Fig. 4. Varying reward structures. Left: Reward at target fixed at 5. Right: Target domain has a step penalty of 0.1

4.2 Maze domain

To test our method on a domain with a different topology and that does not have clear "subgoals", we used a maze domain with 36 states. The results are displayed in figure 5. This is a difficult domain and standard Q-learning performs quite poorly.

Using the constructed options has a far better performance, even when 4 flags are added to the maze (right image in figure 5). For the maze without flags we only used a 3-state chain; despite this small source domain, the resulting options yield a far superior performance. For the maze with flags, we used the small one flag system mentioned above.

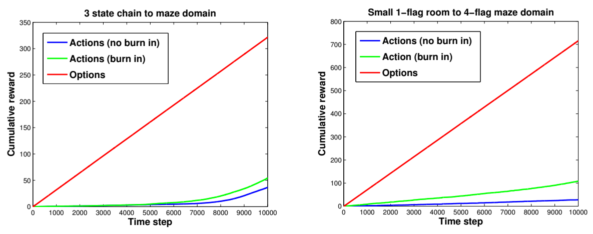

Fig. 5. Left: Performance without flags. Right: Performance with 4 flags (576 states).

5 Conclusion and future work

We presented a novel approach for option construction using bisimulation metrics. The most interesting aspect of our results is the minimal amount of prior knowledge required, which comes in the form of very simple, small prior tasks. We note that the target domains we use bear very little resemblance to the source systems; they do not have similar topology, for example. Yet the bisimuation metric is powerful enough to define good options by comparison to these cases. We have conducted preliminary experiments in larger domains, and the results are generally consistent with those presented here. When using the 4-room domain with very large rooms and no intermediate rewards, however, there is a decrease in performance if the source chain is too small. This is probably due to the fact that bisimulation uses either immediate rewards or differences in dynamics to distinguish states. In very large rooms there is a lot of "open space" with no reward, so most of the states are clustered together, resulting in sub-optimal options. By using chains that are approximately the length of the longest "open space" in the target domain, the performance is as above. Furthermore, as mentioned above, adding a penalty at each step leads to improved performance, as it gives the metric more information.

One objection against using bisimulation is its computational cost. We avoid it here by using statistical sampling, which has a two-fold advantage: an exact model of the system is not required and we can control the computational cost by choosing number of samples.

We believe that our method can also be used to perform a hierarchical decomposition of an MDP by clustering "similar" states. Specifically, two states $t_1, t_2 \in S_2$ are clustered together if for all options o, $t_1 \in \mathcal{I}_o \Leftrightarrow t_2 \in \mathcal{I}_o$ and $\Diamond(t_1, o) == \Diamond(t_2, o)$. Initial results on the rooms world domain are promising, and return a clustering generally consistent with the room topology. A different way of approaching this problem would be to estimate the reward model of the option explicitly and incorporate it in the option construction. This line of work requires further investigation.

Acknowledgements

The authors would like to thank NSERC and ONR for providing financial support.

References

Anderson, J. R. (1996). Act: A simple theory of complex cognition. *American Psychologist 51*, 355–365.

Barto, A. G. and S. Mahadevan (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems 13*(4), 341–379.

Boutilier, C., R. Dearden, and M. Goldszmidt (1995). Exploiting structure in policy construction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*.

Castro, P. S. and D. Precup (2010). Using bisimulation for policy transfer in MDPs. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, pp. 1065–1070.

Comanici, G. and D. Precup (2010). Optimal policy switching algorithms in reinforcement learning. In *Proceedings of AAMAS*.

Dietterich, T. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research 13*, 227–303.

Ferns, N., P. S. Castro, D. Precup, and P. Panangaden (2006). Methods for computing state similarity in Markov Decision Processes. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pp. 174–181.

Ferns, N., P. Panangaden, and D. Precup (2004). Metrics for finite Markov decision processes. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pp. 162–169.

Givan, R., T. Dean, and M. Greig (2003). Equivalence Notions and Model Minimization in Markov Decision Processes. *Artificial Intelligence 147*(1–2), 163–223.

Jonsson, A. and A. G. Barto (2006). Causal graph based decomposition of factored MDPs. *Journal of Machine Learning Research 7*, 2259–2301.

Konidaris, G., S. Kuindersma, A. G. Barto, and R. A. Grupen (2010). Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems 23*, pp. 1162–1170.

Laird, J., M. K. Bloch, and the Soar Group (2011). Soar home page.

Mannor, S., I. Menache, A. Hoze, and U. Klein (2004). Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the 21st International Conference on Machine Learning (ICML-04)*.

McGovern, A. and A. G. Barto (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*.

Mehta, N., S. Ray, P. Tapadalli, and T. Dietterich (2008). Automatic discovery and transfer of maxq hierarchies. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*.

Mugan, J. and B. Kuipers (2009). Autonomously learning an action hierarchy using a learned qualitative state representation. In *Proceedings of the 21st International Joint Conference on Articial Intelligence*.

Parr, R. and S. Russell (1998). Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems (NIPS-98)*.

Precup, D. (2000). *Temporal Abstraction in Reinforcement Learning*. Ph. D. thesis, University of Massachusetts, Amherst.

Ravindran, B. and A. G. Barto (2003). Relativized options: Choosing the right transformation. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*.

Soni, V. and S. Singh (2006). Using Homomorphism to Transfer Options across Reinforcement Learning Domains. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI-06)*.

Sorg, J. and S. Singh (2009). Transfer via Soft Homomorphisms. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2009)*.

Stolle, M. and D. Precup (2002). Learning options in reinforcement learning. In *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*.

Stone, P., R. S. Sutton, and G. Kuhlmann (2005). Reinforcement learning for robocup-soccer keepaway. *Adaptive Behavior 13*(3), 165–188.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press.

Sutton, R. S., D. Precup, and S. Singh (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence 112*, 181–211.

Taylor, J., D. Precup, and P. Panangaden (2009). Bounding performance loss in approximate MDP homomorphisms. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS-09)*.

Taylor, M. E. and P. Stone (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research 10*, 1633–1685.

Şimşek, O., A. P. Wolfe, and A. G. Barto (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML-05)*.

Watkins, C. J. and P. Dayan (1992). Q-learning. *Machine Learning 8*, 279–292.

Wolfe, A. P. and A. G. Barto (2006). Defining object types and options using MDP homomorphisms. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*.

Zang, P., P. Zhou, D. Minnen, and C. Isbell (2009). Discovering options from example trajectories. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*.