Relativized Options: Choosing the Right Transformation

Balaraman Ravindran RAVI@CS.UMASS.EDU
Andrew G. Barto BARTO@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01002 USA

Abstract

Relativized options combine model minimization methods and a hierarchical reinforcement learning framework to derive compact reduced representations of a related family of tasks. Relativized options are defined without an absolute frame of reference, and an option's policy is transformed suitably based on the circumstances under which the option is invoked. In earlier work we addressed the issue of learning the option policy online. In this article we develop an algorithm for choosing, from among a set of candidate transformations, the right transformation for each member of the family of tasks.

1. Introduction

Techniques for scaling decision theoretic planning and learning methods to complex environments with large state spaces have attracted much attention lately, e. g. (Givan et al., 2003; Dietterich, 2000; Sutton et al., 1999). Learning approaches such as the MaxQ algorithm (Dietterich, 2000), Hierarchies of Abstract Machines (Parr & Russell, 1997), and the options framework (Sutton et al., 1999) decompose a complex problem into simpler sub-tasks and employ the solutions of these sub-tasks in a hierarchical fashion to solve the original task. The sub-problems chosen are not only simpler than the original task but often are sub-tasks whose solutions can be repeatedly reused.

Model minimization methods (Givan et al., 2003; Hartmanis & Stearns, 1966; Ravindran & Barto, 2001) also address the issue of planning in large state spaces. These methods attempt to abstract away redundancy in the problem definition to derive an equivalent smaller model of the problem that can be used to derive a solution to the original problem. While reducing entire problems by applying minimization methods is often not feasible, we can apply these ideas to various sub-problems and obtain useful reductions in problem size. In ref. (Ravindran & Barto, 2002) we showed that by applying model minimization we can reduce a family of related sub-tasks to a single sub-task, the solution of which can be suitably transformed to recover the solutions for the entire family. We extended the options framework (Sutton et al., 1999) to accommodate minimization ideas and introduced the notion of a *relativized option*: an option defined without an absolute frame of reference. A relativized option can be viewed as a compact representation of a related family of options. We borrow the terminology from Iba (1989) who developed a similar representation for related families of macro operators.

In this article we explore the case in which one is given the reduced representation of a sub-task and is required to choose the right transformations to recover the solutions to the members of the related family. Such a scenario would often arise in cases where an agent is trained in a small prototypical environment and is required to later act in a complex world where skills learned earlier are useful. For example, an agent may be trained to perform certain tasks in a particular room and then be asked to repeat them in different rooms in the building. In many office and university buildings, rooms tend to be very similar to one another. Thus the problem of navigating in each of these rooms can be considered simply that of suitably transforming the policy learned in the first room. An appropriate set of candidate transformations in this case are reflections and rotations of the room. We build on the options framework and propose a transformation selection mechanism based on Bayesian parameter estimation. We empirically illustrate that the method converges to the correct transformations in a gridworld "building" environment with multiple similar rooms. We also consider a game like environment inspired by the Pengi (Agre, 1988) video game. Here there are many candidate transformations and the con-

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

ditions for minimization are only satisfied very approximately. We introduce a heuristic modification to our transformation selection method and again empirically demonstrate that this method performs adequately in the game environment.

We employ Markov Decision Processes (MDPs) as our basic modeling paradigm. First we present some notation regarding MDPs and partitions (Section 2), followed by a brief summary of MDP homomorphisms and our MDP minimization framework (Section 3). We the introduce relativized options (Section 4) and present our Bayesian parameter estimation based method to choose the correct transformation to apply to a sub-task (Section 5). We then briefly describe approximate equivalence and present a heuristic modification of our likelihood measure (Section 6). In Section 7 we conclude by discussing relation to existing work and some future directions of research.

2. Notation

A *Markov Decision Process* is a tuple $\langle S, A, \Psi, P, R \rangle$, where S is a finite set of states, A is a finite set of actions, $\Psi \subseteq S \times A$ is the set of admissible state-action pairs, $P : \Psi \times S \to [0, 1]$ is the transition probability function with $P(s, a, s')$ being the probability of transition from state s to state s' under action a, and $R : \Psi \to \mathbb{R}$ is the expected reward function, with $R(s, a)$ being the expected reward for performing action a in state s. Let $A_s = \{a | (s, a) \in \Psi\} \subseteq A$ denote the set of actions admissible in state s. We assume that for all $s \in S$, A_s is non-empty.

A *stochastic policy*, π, is a mapping from Ψ to the real interval $[0, 1]$ with $\sum_{a \in A_s} \pi(s, a) = 1$ for all $s \in S$. For any $(s, a) \in \Psi$, $\pi(s, a)$ gives the probability of executing action a in state s. The *value* of a state-action pair (s, a) under policy π is the expected value of the sum of discounted future rewards starting from state s, taking action a, and following π thereafter. The *action-value function*, Q^π, corresponding to a policy π is the mapping from state-action pairs to their values. The solution of an MDP is an *optimal policy*, $\pi^\star$, that uniformly dominates all other possible policies for that MDP.

Let B be a partition of a set X. For any $x \in X$, $[x]_B$ denotes the block of B to which x belongs. Any function f from a set X to a set Y induces an equivalence relation on X, with $[x]_f = [x']_f$ if and only if $f(x) = f(x')$.

An option (or a temporally extended action) (Sutton et al., 1999) in an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ is defined by the tuple $O = \langle \mathcal{I}, \pi, \beta \rangle$, where the initiation set $\mathcal{I} \subseteq S$ is the set of states in which the option can be invoked, π is the option policy, and the termination function $\beta : S \to [0, 1]$ gives the probability of the option terminating in any given state. The option policy can in general be a mapping from arbitrary sequences of state-action pairs (or histories) to action probabilities. We restrict attention to Markov options in which the policies are functions of the current state alone. The states over which the option policy is defined is known as the *domain* of the option.

3. Model Minimization

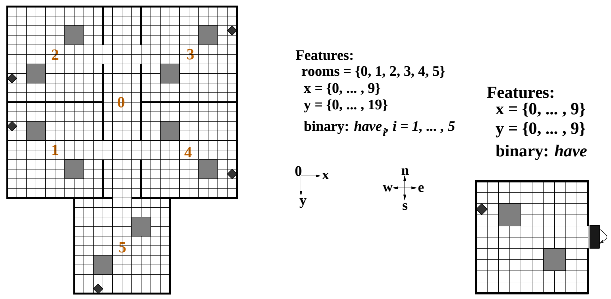
Minimization methods exploit redundancy in the definition of an MDP $\mathcal{M}$ to form a reduced model $\mathcal{M}'$, whose solution yields a solution to the original MDP. One way to achieve this is to derive $\mathcal{M}'$ such that there exists a transformation from $\mathcal{M}$ to $\mathcal{M}'$ that maps equivalent states in $\mathcal{M}$ to the same state in $\mathcal{M}'$, and equivalent actions in $\mathcal{M}$ to the same action in $\mathcal{M}'$. An MDP homomorphism from $\mathcal{M}$ to $\mathcal{M}'$ is such a transformation. Formally, we define it as:

Definition: An *MDP homomorphism* h from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ to an MDP $\mathcal{M}' = \langle S', A', \Psi', P', R' \rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s | s \in S\} \rangle$, with $h((s, a)) = (f(s), g_s(a))$, where $f : S \to S'$ and $g_s : A_s \to A'_{f(s)}$ for $s \in S$, such that $\forall s, s' \in S, a \in A_s$:

$$P'(f(s), g_s(a), f(s')) = \sum_{s'' \in [s']_f} P(s, a, s''), \quad (1)$$

$$R'(f(s), g_s(a)) = R(s, a). \quad (2)$$

We call $\mathcal{M}'$ the *homomorphic image* of $\mathcal{M}$ under h. We use the shorthand $h(s, a)$ to denote $h((s, a))$. The surjection f maps states of $\mathcal{M}$ to states of $\mathcal{M}'$, and since it is generally many-to-one, it generally induces nontrivial equivalence classes of states s of M: $[s]_f$. Each surjection g_s recodes the actions admissible in state s of $\mathcal{M}$ to actions admissible in state $f(s)$ of $\mathcal{M}'$. This *state-dependent* recoding of actions is a key innovation of our definition, which we discuss in more detail below. Condition (1) says that the transition probabilities in the simpler MDP $\mathcal{M}'$ are expressible as sums of the transition probabilities of the states of $\mathcal{M}$ that f maps to that same state in $\mathcal{M}'$. This is the stochastic version of the standard condition for homomorphisms of deterministic systems that requires that the homomorphism commutes with the system dynamics (Hartmanis & Stearns, 1966). Condition (2) says that state-action pairs that have the same image under h have the same expected reward. A policy in $\mathcal{M}'$ *induces* a policy in $\mathcal{M}$ and the following describes how to derive such an induced policy.



(a) (b)

Figure 1. (a) A simple rooms domain with stochastic actions. The task is to collect all 5 objects, the black diamonds, in the environment. The state is described by the number of the room the agent is in, the agent's x and y co-ordinates within the room or corridor with respect to the reference direction indicated in the figure, and boolean variables $have_i$, $i = 1, \ldots, 5$, indicating possession of object in room i.(b) The option MDP corresponding to the sub-task *get-object-and-leave-room*.

Definition: Let $\mathcal{M}'$ be the image of $\mathcal{M}$ under homomorphism $h = \langle f, \{g_s | s \in S\}\rangle$. For any $s \in S$, $g_s^{-1}(a')$ denotes the set of actions that have the same image $a' \in A'_{f(s)}$ under g_s. Let π be a stochastic policy in $\mathcal{M}'$. Then π *lifted to* $\mathcal{M}$ is the policy $\pi_\mathcal{M}$ such that for any $a \in g_s^{-1}(a')$, $\pi_\mathcal{M}(s, a) = \pi(f(s), a') \Big/ \left|g_s^{-1}(a')\right|$.[1]

An optimal policy in $\mathcal{M}'$ when lifted to $\mathcal{M}$ yields an optimal policy in $\mathcal{M}$ (Ravindran & Barto, 2001). Thus one may derive a reduced model of $\mathcal{M}$ by finding suitable homomorphic image. Our minimization framework is an extension of the approach proposed by Dean and Givan (Givan et al., 2003). In ref. (Ravindran & Barto, 2003) we explore application of minimization ideas in an hierarchical setting and show that the homomorphism conditions are a generalization of the *safe* state abstraction conditions introduced by Dietterich (2000).

4. Relativized Options

Consider the problem of navigating in the gridworld environment shown in Figure 1(a). The goal is to reach the central corridor after collecting all the objects in the gridworld. There exists no non-trivial homomorphic image of the entire problem. But there are many similar components in the problem, namely, the five sub-tasks of getting the object and exiting $room_i$. Since the rooms are similarly shaped, we can map one sub-task onto the other by applying simple transformations such as reflections and rotations. We can model this similarity among the sub-tasks by a "partial" homomorphic image—where the homomorphism conditions are applicable only to states in the rooms. One such partial image is shown in Figure 1(b).

A relativized option (Ravindran & Barto, 2002) combines partial reductions with the options framework to represent compactly a family of related options. Here the policy for achieving the option's sub-goal is defined in a partial image MDP (option MDP). When the option is invoked, the current state is projected onto the option MDP and the policy action of the option MDP is lifted to the original MDP based on the states in which the option is invoked. For example, action E in the option MDP will get lifted as action W when invoked in room 3 and as action N when invoked in room 5. Formally we define a relativized option as follows:

Definition: A *relativized option* of an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ is the tuple $O = \langle h, \mathcal{M}_O, \mathcal{I}, \beta\rangle$, where $\mathcal{I} \subseteq S$ is the initiation set, $\beta : S' \to [0, 1]$ is the termination function and $h = \langle f, \{g_s | s \in S\}\rangle$ is a partial homomorphism from the MDP $\langle S, A, \Psi, P, R_O\rangle$ to the option MDP $\mathcal{M}_O = \langle S', A', \Psi', P', R'\rangle$ with R_O chosen based on the sub-task.

In other words, the option MDP $\mathcal{M}_O$ is a partial homomorphic image of an MDP with the same states, actions and transition dynamics as $\mathcal{M}$ but with a reward function chosen based on the option's sub-task. The homomorphism conditions (1) and (2) hold only for states in the domain of the option O. The option policy $\pi : \Psi' \to [0, 1]$ is obtained by solving $\mathcal{M}_O$ by treating it as an episodic task. Note that the initiation set is defined over the state space of $\mathcal{M}$ and not that of $\mathcal{M}_O$. Since the initiation set is typically used by the higher level when invoking the option, we decided to define it over S. When lifted to $\mathcal{M}$, π is transformed into policy fragments over Ψ, with the transformation depending on the state of $\mathcal{M}$ the system is currently in.

Going back to our example in Figure 1(a), we can now define a single *get-object-and-leave-room* relativized option using the option MDP of Figure 1(b). The policy learned in this option MDP can then be lifted to $\mathcal{M}$ to provide different policy fragments in the different rooms.

5. Choosing Transformations

In (Ravindran & Barto, 2002) we explored the issue of learning the relativized option policy while learning to

[1] It is sufficient that $\sum_{a \in g_s^{-1}(a')} \pi_\mathcal{M}(s, a) = \pi(f(s), a')$, but we use the above definition to make the lifted policy unique.

solve the original task. We established that relativized options significantly speed up initial learning and enable more efficient knowledge transfer. We assumed that the option MDP and the required transformations were specified beforehand. In a wide variety of problem settings we can specify a set of possible transformations to employ with a relativized option but lack sufficient information to specify which transformation to employ under which circumstances. For example, we can train a two-arm ambidextrous robot to accomplish certain sub-tasks like grasping and moving objects using one arm and a small set of object orientations. If the robot is then supplied a set of rotations and reflections, it could learn the suitable transformations required when it uses the other arm and when it encounters objects in different orientations.

Given a set of candidate transformations $\mathcal{H}$ and the option MDP $\mathcal{M}_O = \langle S', A', \Psi', P', R' \rangle$, how do we choose the right transformation to employ at each invocation of the option? Let $\overline{s}$ be a function of the current state s that captures the features of the states necessary to distinguish the particular sub-problem under consideration.[2] We formulate the problem of choosing the right transformation as a family of Bayesian parameter estimation problems, one for each possible value of $\overline{s}$.

We have a parameter, θ, that can take a finite number of values from $\mathcal{H}$. Let $p(h, \overline{s})$ denote the prior probability that $\theta = h$, i.e., the prior probability that h is the correct transformation to apply in the sub-problem represented by $\overline{s}$. The set of samples used for computing the posterior distribution is the sequence of transitions, $\langle s_1, a_1, s_2, a_2, \cdots \rangle$, observed when the option is executing. Note that the probability of observing a transition from s_i to s_{i+1} under a_i for all i, is independent of the other transitions in the sequence. We employ recursive Bayes learning to update the posterior probabilities incrementally.

Let p_n be the posterior probability n time steps after the option was invoked. We start by setting $p_0(h, \overline{s}) = p(h, \overline{s})$ for all h and $\overline{s}$. Let $E_n = \langle s_n, a_n, s_{n+1} \rangle$ be the transition experienced after n time steps of option execution. We update the posteriors for all $h = \langle f, \{g_s | s \in S\} \rangle$ as follows:

$$p_n(h, \overline{s}) = \frac{Pr(E_n | h, \overline{s}) p_{n-1}(h, \overline{s})}{\mathcal{N}} \quad (3)$$

where $Pr(E_n | h, \overline{s}) = P'(f(s_n), g_{s_n}(a_n), f(s_{n+1}))$ is the probability of observing the h-projection of transition E_n in the option MDP and $\mathcal{N} = \sum_{h' \in \mathcal{H}} P'((f'(s_n), g'_{s_n}(a), f'(s_{n+1})) p_{n-1}(h', \overline{s})$ is a normalizing factor. When an option is executing, at time step n we use $\widehat{h} = \arg\max_h p_n(h, \overline{s})$ to project the state to the option MDP and lift the action to the original MDP. After experiencing a transition, we update the posteriors of all the transformations in $\mathcal{H}$ using (3).

[2] In the example in Figure 1, the room number is sufficient, while in an object-based environment some property of the target object, say color, might suffice. Frequently $\overline{s}$ is a simple function of s like a projection onto a subset of features, as in the rooms example.

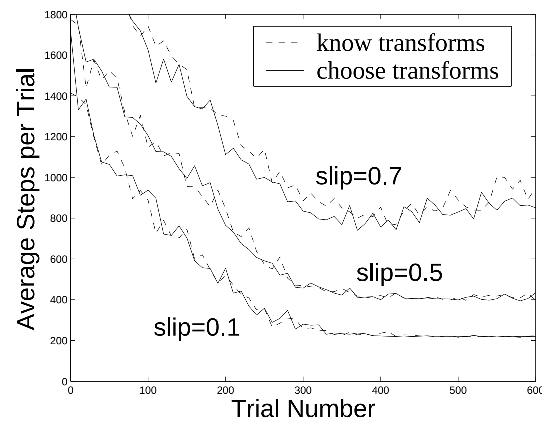


Figure 2. Comparison of initial performance of agents with and without knowledge of the appropriate partial homomorphisms on the task shown in Figure 1 with various levels of stochasticity.

5.1. Results

We tested the algorithm on the gridworld in Figure 1. The agent has one *get-object-and-exit-room* relativized option defined in the option MDP in Figure 1(b). Considering all possible combinations of reflections about the x and y axes and rotations through integral multiples of 90 degrees, we have eight distinct transformations in $\mathcal{H}$. For each of the rooms in the world there is one transformation in $\mathcal{H}$ that is the desired one. In some ways this is a contrived example chosen so that we can illustrate the correctness of our algorithm. Reduction in problem size is possible in this domain by more informed representation schemes. We will discuss, briefly, the relation of such schemes to relativized options in the Section 7. The agent employed hierarchical SMDP Q-learning with ϵ-greedy exploration, with $\epsilon = 0.1$. The learning rate was set at 0.01 and γ at 0.9. We initialized the priors in each of the rooms to a uniform distribution with $p_0(h, \overline{s}) = 0.125$ for all $h \in \mathcal{H}$ and $\overline{s}$. The trials were terminated either on completion of the task or after 3000 steps. The results shown in Figure 2 are averaged over 100 independent runs. We repeated the experiment with different levels

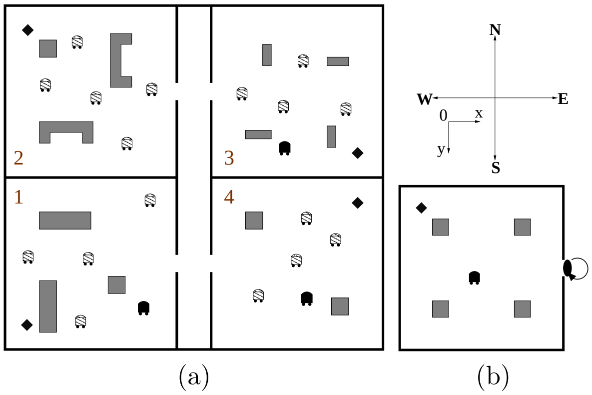


Figure 3. (a) A game domain with interacting robots and stochastic actions. The task is to collect all 4 objects, the black diamonds, in the environment. The state is described as before by the number of the room the agent is in, the agent's x and y co-ordinates within the room or corridor, boolean variables $have_i$, $i = 1, \ldots, 4$, and in addition, the room number, and x and y co-ordinates of every robot in the domain. The robots are of two types—benign (shaded) and delayers (black). See text for explanation of robot types. (b) The option MDP corresponding to the sub-task *get-object-and-leave-room*. There is just one delayer robot in this image MDP.

of stochasticity, or slip, for the actions.

The probability with which an action "fails", i.e., results in movement in a direction other than the desired one, is called the *slip* in the environment. The greater the slip, the harder the problem. As shown in Figure 2 the agent rapidly learned to apply the right transformation in each room under different levels of stochasticity. We compared the performance to an agent learning with primitive actions alone. The primitive action agent didn't start improving its performance until after 30,000 iterations and hence we did not employ that agent in our experiments. We also compared the performance to an agent that knew the right transformations to begin with. As can be seen in Figure 2 the difference in performance is not significant.[3] In this particular task our transformation-choosing algorithm manages to identify the correct transformations without much loss in performance since there is nothing catastrophic in the environment and the agent is able to recover quickly from wrong initial choices. Typically the agent identifies the right transformation in a few updates. For example in room 5 the agent quickly discards all pure reflections and the posterior for transform 5, a rotation through 90 degrees, converges to 1.0 by the tenth update.

[3]All the significance tests were *two sample t-tests*, with a p-value of 0.01, on the learning time distributions.

6. Approximate Equivalence

The task in Figure 1 exhibits perfect symmetric equivalence. Such a scenario does not often arise in practice. In this section we consider a more complex game example with imperfect equivalences inspired by the *Pengi* environment used by Agre (1988) to demonstrate the effectiveness of deictic representations. The layout of the game is shown in Figure 3(a). As in the previous example, the objective of the game is to gather all the objects in the environment and the environment has standard stochastic gridworld dynamics.

Each room also has several robots, one of which might be a *delayer*. If the agent happens to occupy the same square as the delayer then it is prevented from moving for a randomly determined number of time steps, given by a geometric distribution with parameter *hold*. When not occupying the same square, the delayer pursues the agent with some probability, *chase*. The other robots are benign and act as mobile obstacles, executing random walks. The agent does not have prior knowledge of the hold and chase parameters, nor does the agent recognize the delayer on entering the room. But the agent is aware of the room numbers, and x and y co-ordinates of all the robots in the environment, all the *have* features, its own room number, and x and y co-ordinates.

The option MDP (Figure 3(b)) we employ is a symmetrical room with just one robot—a delayer with fixed chase and hold parameters. The features describing the state space of the option MDP consists of the x and y co-ordinates of the agent and the robot and a boolean variable indicating possession of the object. Unlike in the previous example, no room matches the option MDP exactly and no robot has the same chase and hold parameters as the delayer in the image MDP. In fact room 2 does not have a delayer robot. While employing the relativized option, the agent has to not only figure out the right orientation of the room but also which robot is the delayer, i. e. , which robot's location it needs to project onto the option MDP's delayer's features. Thus there are 40 candidate transformations, comprising of different reflections, rotations and projections.[4]

While this game domain is also a contrived example, projection transformations arise often in practice. In an environment with objects, selecting a subset of objects (say blocks) with which to perform a sub-task can be modeled as projecting the features of the chosen ob-

[4]Please refer to http://www-all.cs.umass.edu/~ravi/ICML03-experiments.pdf for a more detailed description of the experimental setup.

jects onto an option MDP. Such projections can also be used to model indexical representations (Agre, 1988). In this game example, finding the projection that correctly identifies the delayer is semantically equivalent to implementing a *the-robot-chasing-me* pointer.

In ref. (Ravindran & Barto, 2002) we extended our minimization framework to accommodate approximate equivalences and symmetries. Since we are ignoring differences between the various rooms there will be a loss in asymptotic performance. We discussed bounding this loss using Bounded-parameter MDPs (Givan et al., 2000) and approximate homomorphisms.

We cannot employ the method we developed in the previous section for choosing the correct transformations with approximate homomorphisms. In some cases even the correct transformation causes a state transition the agent just experienced to project to an impossible transition in the image MDP, i.e., one with a P' value of 0. Thus the posterior of the correct transformation might be set to zero, and once the posterior reaches 0, it stays there regardless of the positive evidence that might accumulate later.

To address this problem we employed a heuristic—lower bound the P' values used in the update equation (3). We compute a "weight" for the transformations using:

$$w_n(h, \overline{s}) = \frac{\overline{P'}((f(s), g_s(a), f(s'))w_{n-1}(h, \overline{s})}{\mathcal{N}} \qquad (4)$$

where $\overline{P'}(s, a, s') = \max(\nu, P'(s, a, s'))$, and $\mathcal{N} = \sum_{h' \in \mathcal{H}} \overline{P'}((f'(s), g'_s(a), f'(s'))w_{n-1}(h', \overline{s})$ is the normalizing factor. Thus even if the projected transition has a probability of 0 in the option MDP, we use a value of ν in update. The weight $w(h, \overline{s})$ is a measure of the likelihood of h being the right transformation in $\overline{s}$ and we use this weight instead of the posterior probability.

We measured the performance of this heuristic in the game environment. Since none of the rooms match the option MDP closely, keeping the option policy fixed leads to very poor performance. So we allowed the agent to continually modify the option policy while learning the correct transformations. As with the earlier experiments, the agent simultaneously learned the policies of the lower level and higher level tasks, using a learning rate of 0.05 for the higher level MDP and 0.1 for the relativized option. The discount factor γ was set to 0.9, ϵ to 0.1 and ν to 0.01. We initialized the initial weights in each of the rooms to a uniform distribution with $w_0(h, \overline{s}) = 0.025$ for all $h \in \mathcal{H}$ and $\overline{s}$. The trials were terminated either on completion of the task or after 6000 steps.

We compared the performance of this heuristic against that of an agent that had 4 different regular Markov options. The agent learned the option policies simultaneously with the higher level task. The results shown in Figure 4 were averaged over 10 independent runs. As shown in Figure 4 the agent using the heuristic shows rapid improvement in performance initially. This vindicates our belief that it is easier to learn the "correct" transformations than to learn the policies from scratch. As expected the asymptotic performance of the regular agent is better than the relativized agent. We couldn't compare the heuristic against an agent that knew the correct transformations beforehand since there are no correct transformation in some of the cases.

Figure 5 shows the evolution of weights in room 4 during a typical run. The weights have not converged to their final values after 600 updates, but transform 12, the correct transformation in this case, has the biggest weight and is picked consistently. After about thousand updates the weight for transform 12 reached nearly 1 and stayed there. Figure 6 shows the evolution of weights in room 2 during a typical run. The weights oscillate a lot during the runs, since none of the transforms are entirely correct in this room. In this particular run, the agent converged to transform 5 after about 1000 updates, but that is not always the case. But the agent can solve the sub-task in room 2 as long as it correctly identifies the orientation and employs any of transforms 1 through 5.

While none of the transformations are completely "correct" in this environment, especially in room 2, the agent is able to complete each sub-task successfully after a bit of retraining. One can easily construct environments where this is not possible and we need to consider some fail-safe mechanism. One alternative is to learn the policy for the failure states at the primitive action level and use the option only in states where there is some chance of success. Another alternative is to spawn a copy of the relativized option which is available only in the failure states and allow the agent to learn the option policy from scratch thus preventing harmful generalization. We have not explored this issue in detail in this paper, since we were looking to develop a method that allows us to select the right transformation under the assumption that we have access to such a transformation.

7. Discussion and Future Work

In this article we presented results on choosing from among a set of candidate partial homomorphisms, assuming that the option MDP and the policy are com-

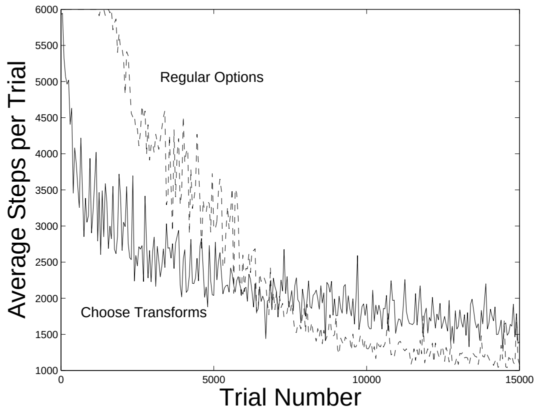


Figure 4. Comparison of the performance of an agent with 4 regular options and an agent using a relativized option and no knowledge of the correct transformation on the task shown in Figure 3(a). The option MDP employed by the relativized agent is shown in Figure 3(b).

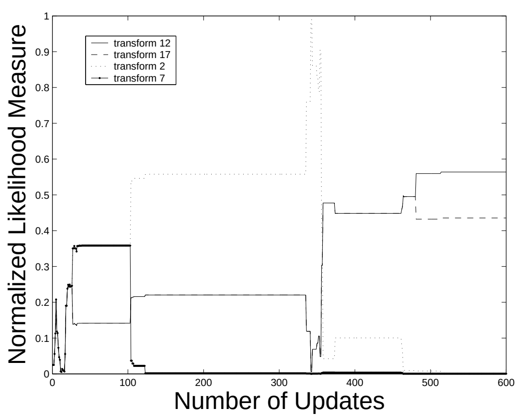


Figure 5. Typical evolution of weights for a subset of transforms in Room 4 in Figure 3(a), with a slip of 0.1.

pletely specified. We are currently working on relaxing these assumptions. Specifically we are looking at learning the option policy online and working with partial specification of the option MDP. We are also applying this approach to more complex problems. In particular we are applying this to solving a family of related tasks on the UMass torso (Wheeler et al., 2002). The torso is initially trained on a single task, and is later required to learn the transformations to apply to derive solutions to the other tasks in the family.

The experiments reported in this paper employ a two-level hierarchy. Our ideas generalize naturally to multi-level hierarchies. The semi-Markov decision process (SMDP) framework is an extension of the MDP framework which is widely employed in modeling hierarchical systems (Dietterich, 2000; Sutton et al., 1999).

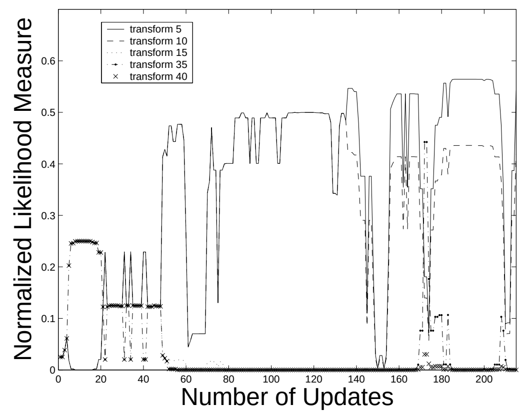


Figure 6. Typical evolution of weights for a subset of transforms in Room 2 in Figure 3, with a slip of 0.1.

We recently developed the notion of SMDP homomorphism and used it in defining hierarchical relativized options (Ravindran & Barto, 2003). The option is now defined using an option SMDP and hence can have other options as part of its action set. We are currently applying the methods developed in this paper to domains that require hierarchical options.

We employed the options framework in this work, but our ideas are applicable to other hierarchical frameworks such as MAXQ (Dietterich, 2000) and Hierarchies of Abstract Machines (Parr & Russell, 1997). Sub-tasks in these frameworks can also be relativized and we could learn homomorphic transformations between related sub-tasks. Relativization can also be combined with automatic hierarchy extraction algorithms (McGovern & Barto, 2001; Hengst, 2002) and algorithms for building abstractions specific to different levels of the hierarchy (Dietterich, 2000; Jonsson & Barto, 2001).

In some environments it is possible to choose representation schemes to implicitly perform the required transformation depending on the sub-task. Examples of such schemes include ego-centric and deictic representations (Agre, 1988), in which an agent senses the world via a set of pointers and actions are specified with respect to these pointers. In the game environment we employed projection transformations to identify the delayer robot. As mentioned earlier, this is a form of indexical representation with the available projections specifying the possible pointer configurations. Thus choosing the right transformation can be viewed as learning *the-robot-chasing-me* pointer. While employing such representations largely simplifies the solution of a problem, they are frequently very difficult to design. Our work is a first step toward systematizing the transformations needed to map similar subtasks onto each other in the absence of versatile sensory mechanisms. The concepts developed here will also serve as stepping stones to designing sophisticated representation schemes.

In our approach we employ different transformations to get different interpretations of the same MDP model. Researchers have investigated the problem of employing multiple models of an environment and combining the predictions suitably using the EM algorithm (Haruno et al., 2001) and mixtures of experts models (Doya et al., 2002). We are investigating specializations of such approaches to our setting where the multiple models can be obtained by simple transformations of one another. This close relationship between the various models might yield significant simplification of these architectures.

Acknowledgments

We wish to thank Mohammad Ghavamzadeh, Dan Bernstein, Amy McGovern and Mike Rosenstein for many hours of useful discussion. This material is based upon work supported by the National Science Foundation under Grant No. ECS-0218125 to Andrew G. Barto and Sridhar Mahadevan. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Agre, P. E. (1988). *The dynamic structure of everyday life* (Technical Report AITR-1085). Massachusetts Institute of Technology.

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

Doya, K., Samejima, K., Katagiri, K., & Kawato, M. (2002). Multiple model-based reinforcement learning. To appear in Neural Computation.

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in Markov decision processes. To appear in Artificial Intelligence.

Givan, R., Leach, S., & Dean, T. (2000). Bounded-parameter Markov decision processes. *Artificial Intelligence*, *122*, 71–109.

Hartmanis, J., & Stearns, R. E. (1966). *Algebraic structure theory of sequential machines*. Englewood Cliffs, NJ: Prentice-Hall.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Computation*, *13*, 2201–2220.

Hengst, B. (2002). Discovering hierarchy in reinforcement learning with HEXQ. *Proceedings of the 19th International Conference on Machine Learning* (pp. 243–250).

Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, *3*, 285–317.

Jonsson, A., & Barto, A. G. (2001). Automated state abstraction for options using the u-tree algorithm. *Proceedings of Advances in Neural Information Processing Systems 13* (pp. 1054–1060). Cambridge, MA: MIT Press.

McGovern, A., & Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. *Proceedings of the 18th International Conference on Machine Learning ICML 2001* (pp. 361–368).

Parr, R., & Russell, S. (1997). Reinforcement learning with hierarchies of machines. *Proceedings of Advances in Neural Information Processing Systems 10* (pp. 1043–1049). MIT Press.

Ravindran, B., & Barto, A. G. (2001). *Symmetries and model minimization of Markov decision processes* (Technical Report 01-43). University of Massachusetts, Amherst.

Ravindran, B., & Barto, A. G. (2002). Model minimization in hierarchical reinforcement learning. *Proceedings of the Fifth Symposium on Abstraction, Reformulation and Approximation (SARA 2002)* (pp. 196–211). New York, NY: Springer-Verlag.

Ravindran, B., & Barto, A. G. (2003). SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes. To appear in the *Proceedings of the Eighteenth Internatinal Joint Conference on Artificial Intelligence (IJCAI 2003).*

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

Wheeler, D. S., Fagg, A. H., & Grupen, R. A. (2002). Learning prospective pick and place behavior. *Proceedings of the International Conference on Development and Learning (ICDL'02).*