Effective Control Knowledge Transfer Through Learning Skill and Representation Hierarchies [*]

Mehran Asadi
Department of Computer Science
West Chester University of Pennsylvania
West Chester, PA 19383
masadi@wcupa.edu

Manfred Huber
Department of Computer Science and Engineering
The University of Texas at Arlington
Arlington, TX 76019
huber@cse.uta.edu

Abstract

Learning capabilities of computer systems still lag far behind biological systems. One of the reasons can be seen in the inefficient re-use of control knowledge acquired over the lifetime of the artificial learning system. To address this deficiency, this paper presents a learning architecture which transfers control knowledge in the form of behavioral skills and corresponding representation concepts from one task to subsequent learning tasks. The presented system uses this knowledge to construct a more compact state space representation for learning while assuring bounded optimality of the learned task policy by utilizing a representation hierarchy. Experimental results show that the presented method can significantly outperform learning on a flat state space representation and the MAXQ method for hierarchical reinforcement learning.

1 Introduction

Learning capabilities in biological systems far exceed the ones of artificial agents, partially because of the efficiency with which they can transfer and re-use control knowledge acquired over the course of their lives.

To address this, knowledge transfer across learning tasks has recently received increasing attention [Ando and Zhang, 2004; Taylor and Stone, 2005; Marthi *et al.*, 2005; Marx *et al.*, 2005]. The type of knowledge considered for transfer includes re-usable behavioral macros, important state features, information about expected reward conditions, and background knowledge. Knowledge transfer is aimed at improving learning performance by either reducing the learning problem's complexity or by guiding the learning process.

Recent work in Hierarchical Reinforcement Learning (HRL) has led to approaches for learning with temporally extended actions using the framework of Semi-Markov Decision Processes (SMDPs) [Sutton *et al.*, 1999], for learning subgoals and hierarchical action spaces, and for learning abstract representations [Givan *et al.*, 1997]. However, most of these techniques only address one of the aspects of transfer and do frequently not directly address the construction of action and representation hierarchies in life-long learning.

The work presented here focuses on the construction and transfer of control knowledge in the form of behavioral skill hierarchies and associated representational hierarchies in the context of a reinforcement learning agent. In particular, it facilitates the acquisition of increasingly complex behavioral skills and the construction of appropriate, increasingly abstract and compact state representations which accelerate learning performance while ensuring bounded optimality. Moreover, it forms a state hierarchy that encodes the functional properties of the skill hierarchy, providing a compact basis for learning that ensures bounded optimality.

2 Reinforcement Learning

In the RL framework, a learning agent interacts with an environment over a series of time steps $t = 0, 1, 2, 3, \ldots$. At each time t, the agent observes the state of the environment, s_t , and chooses an action, a_t , which causes the environment to transition to state s_{t+1} and to emit a reward, r_{t+1}. In a Markovian system, the next state and reward depend only on the preceding state and action, but they may depend on these in a stochastic manner. The objective of the agent is to learn to maximize the expected value of reward received over time. It does this by learning a (possibly stochastic) mapping from states to actions called a policy. More precisely, the objective is to choose each action a_t so as to maximize the expected return, $E\{\sum_{i=0}^{\infty} \gamma^i r_{t+i}\}$, where $\gamma \in [0, 1]$ is a discount-rate parameter. Other return formulations are also possible. A common solution strategy is to approximate the optimal action-value function, or Q-function, which maps each state and action to the maximum expected return starting from the given state and action and thereafter always taking the best actions.
To permit the construction of a hierarchical learning system, we model our learning problem as a Semi-Markov Decision Problem (SMDP) and use the options framework [Sutton *et al.*, 1999] to define subgoals. An option is a temporally extended action which, when selected by the agent, executes

[*]This work was supported in part by DARPA FA8750-05-2-0283 and NSF ITR-0121297. The U. S. Government may reproduce and distribute reprints for Governmental purposes notwithstanding any copyrights. The authors' views and conclusions should not be interpreted as representing official policies or endorsements, expressed or implied, of DARPA, NSF, or the Government.

until a termination condition is satisfied. While an option is executing, actions are chosen according to the option's own policy. An option is like a traditional macro except that instead of generating a fixed sequence of actions, it follows a closed-loop policy so that it can react to the environment. By augmenting the agent's set of primitive actions with a set of options, the agent's performance can be enhanced. More specifically, an option is a triple $o_i = (I_i, \pi_i, \beta_i)$, where I_i is the option's input set, i.e., the set of states in which the option can be initiated; π_i is the option's policy defined over all states in which the option can execute; and β_i is the termination condition, i.e., the option terminates with probability $\beta_i(s)$ for each state s. Each option that we use in this paper bases its policy on its own internal value function, which can be modified over time in response to the environment. The value of a state s under an SMDP policy π^o is defined as [Boutilier *et al.*, 1999; Sutton *et al.*, 1999]:

$$V^\pi(s) = E\left[R(s, o_i) + \sum_{s'} F(s'|s, o_i)V^\pi(s')\right]$$

where

$$F(s'|s, o_i) = \sum_{k=1}^{\infty} P(s_t = s'|s_t = s, o_i)\gamma^k \tag{1}$$

and

$$R(s', o_i) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots |\epsilon(\pi_i^o, s, t)] \tag{2}$$

where r_t denotes the reward at time t and $\epsilon(\pi^o, s, t)$ denotes the event of an action under policy π^o being initiated at time t and in state s [Sutton *et al.*, 1999].

3 Hierarchical Knowledge Transfer

In the approach presented here, skills are learned within the framework of Semi-Markov Decision Processes (SMDP) where new task policies can take advantage of previously learned skills, leading from an initial set of basic actions to the formation of a skill hierarchy. At the same time, abstract representation concepts are derived which capture each skill's goal objective as well as the conditions under which use of the skill would predict achievement of the objective. Figure 1 shows the approach.

The state representations are formed here within the framework of Bounded Parameter Markov Decision Processes (BPMDPs) [Givan *et al.*, 1997] and include a decision-level model and a more complex evaluation-level model. Learning of the new task is then performed on the decision-level model using Q-learning, while a second value function is maintained on the evaluation-level model. When an inconsistency is discovered between the two value functions, a refinement augments the decision-level model by including the concepts of the action that led to the inconsistency.

Once a policy for the new task is learned, subgoals are extracted from the system model and corresponding subgoal skills are learned off-line. Then goal and probabilistic *affordance* concepts are learned for the new subgoal skills and both, the new skills and concepts are included into the skill and representation hierarchies in the agent's memory, making them available for subsequent learning tasks.

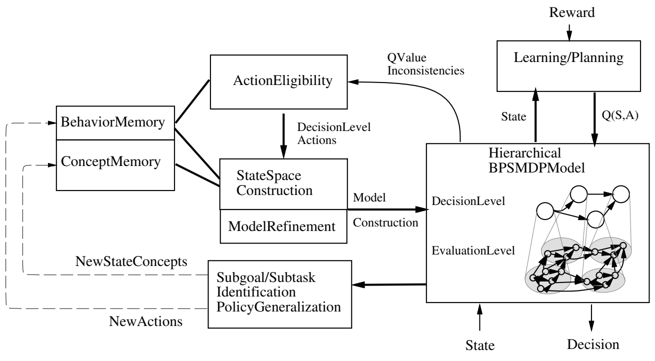

Figure 1: System Overview of the Approach for Hierarchical Behavior and State Concept Transfer.

3.1 Learning Transferable Skills

To learn skills for transfer, the approach presented here tries to identify subgoals. Subgoals of interest here are states that have properties that could be useful for subsequent learning tasks. Because the new tasks' requirements, and thus their reward functions, are generally unknown, the subgoal criterion used here does not focus on reward but rather on local properties of the state space in the context of the current task domain. In particular, the criterion used attempts to identify states which locally form a significantly stronger "attractor" for state space trajectories as measured by the relative increase in visitation likelihood.

To find such states, the subgoal discovery method first generates N random sample trajectories from the learned policy and for each state, s, on these trajectories and determines the expected visitation likelihood, $C_H^*(s)$, where, $C_H^*(s)$ is the sum over all states in sample trajectories $h_i \in H$ weighed by their accumulated likelihood to pass through s. The change of visitation likelihoods along a sample trajectory, h_i, is then determined as $\Delta_H(s_t) = C_H^*(s_t) - C_H^*(s_{t-1})$, where s_t is the t^{th} state along the path. The ratio of this change along the path is then computed as

$$\frac{\Delta_H(s_t)}{\max(1, \Delta_H(s_{t+1}))}$$

for every state in which $\Delta_H(s_t) > 0$. Finally, a state s_t is considered a potential subgoal if its average change ratio is significantly greater than expected from the distribution of the ratios for all states [1]. For all subgoals found, corresponding policies are learned off-line as SMDP options, o_i, and added to the skill hierarchy.

Definition 1 *A state s' is a direct predecessor of state s, if under a learned policy the action in state s' can lead to s i.e., $P(s|s', a) > 0$.*

Definition 2 *The count metric for state s under a learned policy, π, is the sum over all possible state space trajectories weighed by their accumulated likelihood to pass through state s.*

[1]The threshold is computed automatically using a t-test based criterion and a significance threshold of 2.5%.

Let $C^*_\pi(s)$ be the count for state s, then:

$$C^1_\pi(s) = \sum_{s' \neq s} P(s|s', \pi(s')) \tag{3}$$

and

$$C^t_\pi(s) = \sum_{s' \neq s} P(s|s', \pi(s')) C^{t-1}_\pi(s') \tag{4}$$

$$C^*_\pi(s) = \sum_{i=1}^{n} C^i_\pi(s) \tag{5}$$

where n is such that $C^n_\pi(s) = C^{n+1}_\pi(s)$ or $n = |S|$. The condition $s' \neq s$ prevents the counting of self loops and $P(s|s', \pi(s'))$ is the probability of reaching state s from state s' by executing action $\pi(s')$. The slope of $C^*_\pi(s_t)$ along a path, ρ, under policy π is:

$$\Delta_\pi(s_t) = C^*_\pi(s_t) - C^*_\pi(s_{t-1}) \tag{6}$$

where s_t is the t^{th} state along the path. In order to identify subgoals, the gradient ratio $\Delta_\pi(s_t)/\max(1, \Delta_\pi(s_{t+1}))$ is computed for states where $\Delta_\pi(s_t) > 0$. A state s_t is considered a potential subgoal candidate if the gradient ratio is greater than a specified threshold $\mu > 1$. Appropriate values for this user-defined threshold depend largely on the characteristics of the state space and result in a number of subgoal candidates that is inversely related to the value of μ. This approach is an extension of the criterion in [Goel and Huber, 2003] with $\max(1, \Delta_\pi(s_{t+1}))$ addressing the effects of potentially obtaining negative gradients due to nondeterministic transitions.
In order to reduce the computational complexity of the above method in large state spaces, the gradient ratio is here computed using Monte Carlo sampling.

Definition 3 *Let $H = \{h_1, ..., h_N\}$ be N sample trajectories induced by policy π, then the sampled count metric, $C^*_H(s)$, for each state s that is on the path of at least one path h_i can be calculated as the average of the accumulated likelihoods of each path, h_i, $1 \leq i \leq N$, rescaled by the total number of possible paths in the environment.*
We can show that for trajectories h_i and sample size N such that

$$N \geq \frac{\max_t C^*_H(s_t)}{\epsilon_N^2} 2(1+\epsilon_N) log(\frac{2}{1-p}) \tag{7}$$

the following statement is true with probability p:

$$|C^*_H(s_t) - C^*_\pi(s_t)| \leq \epsilon_N$$

Theorem 1 *Let $H = \{h_1, ..., h_N\}$ be N sample trajectories induced by policy π with N selected according to Equation 7. If $\frac{\Delta_H(s_t)}{\max(1, \Delta_H(s_{t+1}))} > \mu + \frac{2\epsilon_N(\mu+1)}{\max(1, \Delta_H(s_{t+1}))}$, then $\frac{\Delta_\pi(s_t)}{\max(1, \Delta_\pi(s_{t+1}))} > \mu$ with probability $\geq p$.*
Theorem 1 implies that for a sufficiently large sample size the exhaustive and the sampling method predict the same subgoals with high probability.

3.2 Learning Functional Descriptions of State

The power of complex actions to improve learning performance has two main sources; (i) their use reduces the number of decision points necessary to learn a policy, and (ii) they usually permit learning to occur on a more compact state representation. To harness the latter, it is necessary to automatically derive abstract state representations that capture the functional characteristics of the actions. To do so, the presented approach builds a hierarchical state representation within the basic framework of BPMDPs extended to SMDPs, forming a hierarchical Bounded Parameter SDMP (BPSMDP). Model construction occurs in a multi-stage, action-dependent fashion, allowing the model to adapt rapidly to action set changes.

The BPSMDP state space is a partition of the original state space where the following inequalities hold for all blocks (BPSMDP states) B_i and actions o_j [Asadi and Huber, 2005]:

$$\left| \sum_{s'' \in B_j} F(s''|s, o_i) - \sum_{s'' \in B_j} F(s''|s', o_i) \right| \leq \delta \tag{8}$$

$$|R(s, o_i) - R(s', o_i)| \leq \epsilon \tag{9}$$

where $R(s, o)$ is the expected reward for executing option o in state s, and $F(s'|s, o)$ is the discounted transition probability for option o initiated in state s to terminate in state s'. These properties of the BPSMDP model ensure that the value of the policy learned on this model is within a fixed bound the optimal policy value on the initial model, where the bound is a function of ϵ and δ [Givan *et al.*, 1997].

To make the construction of the BPSMDP more efficient, the state model is constructed in multiple steps. First functional concepts for each option, o, are learned as termination concepts $C_{t,o}$, indicating the option's goal condition, and probabilistic prediction concepts ("affordances"), $C_{p,o,x}$, indicating the context under which the option will terminate successfully with probability $x \pm \epsilon$. These conditions guarantee that any state space utilizing these concepts in its state factorization fulfills the conditions of Equation 8 for any single action.

To construct an appropriate BPMDP for a specific action set $O_t = \{o_i\}$, an initial model is constructed by concatenating all concepts associated with the options in O_t. Additional conditions are then derived to achieve the condition of Equation 8 and, once reward information is available, the reward condition of Equation 9. This construction facilitates efficient adaptation to changing action repertoires.

To further utilize the power of abstract actions, a hierarchy of BPSMDP models is constructed here where the decision-level model utilizes the set of options considered necessary while the evaluation-level uses all actions not considered redundant. In the current system, a simple heuristic is used where the decision-level set consists only of the learned subgoal options while the evaluation-level set includes all actions.

Let $P = \{B_1, \ldots, B_n\}$ be a partition for state space S derived by the action-dependent partitioning method, using subgoals $\{s_1, \ldots, s_k\}$ and options to these subgoals $\{o_1, \ldots, o_k\}$. If the goal state G belongs to the set of subgoals $\{s_1, \ldots, s_k\}$, then G is achievable by options $\{o_1, \ldots, o_k\}$ and the task is learnable according to Theorem 2.

Theorem 2 *For any policy π for which the goal G can be represented as a conjunction of terminal sets (subgoals) of the available actions in the original MDP M, there is a policy π_P in the reduced MDP, M_P , that achieves G as long as for each state s_t in M for which there exists a path to G , there exists a path such that $F(G|s_t, \pi_P(s_t)) > \delta$.*

If $G \notin \{s_1, \ldots, s_k\}$ then the task may not be solvable using only the options that terminate at subgoals. The proposed approach solves this problem by maintaining a separate value function for the original state space while learning a new task on the partition space derived from only the subgoal options. During learning, the agent has access to the original actions as well as all options, but makes decisions only based on the abstract partition space information.
While the agent tries to solve the task on the abstract partition space, it computes the difference in Q-values between the best actions in the current state in the abstract state space and in the original state space. If the difference is larger than a constant value (given by Theorem 2), then there is a significant difference between different states underlying the particular block that was not captured by the subgoal options. Theorem 3 [Kim and Dean, 2003] shows that if blocks are stable with respect to all actions the difference between the Q-values in the partition space and in the original state space is bounded by a constant value.

Theorem 3 *Given an MDP $M = (S, A, T, R)$ and a partition P of the state space M_P , the optimal value function of M given as V^* and the optimal value function of M_P given as V_P^* satisfy the bound on the distance*

$$\| V^* - V_P^* \|_\infty \leq 2\left(1 + \frac{\gamma}{1-\gamma}\epsilon_p\right)$$

where $\epsilon_p = \min_{V_p} \| V^* - V_P^* \|_\infty$ and

$$LV(s) = \max_a [R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V(s)]$$

When the difference between the Q-values for states in block B_i are greater than $2(1 + \frac{\gamma}{1-\gamma}\epsilon_p)$, then the primitive action that achieves the highest Q-value on the original state in the MDP will be added to the action space of those states that are in block B_i and block B_i is refined until it is stable for the new action set. Once no such significant difference exists, the goal will be achievable in the resulting state space according to Theorem 2.

3.3 Learning on a Hierarchical State Space

To learn new tasks, Q-learning is used here at the decision-level of the BPSMDP hierarchy. Because the compact decision-level state model encodes only the aspects of the environment relevant to a subset of the actions, it only ensures the learning of a policy within the pre-determined optimality bounds if the policy only utilizes the actions in the decision-level action set. Since, however, the action set has to be selected without knowledge of the new task, it is generally not possible to guarantee that it contains all required actions.

To address this, the approach maintains a second value function on top of the evaluation-level system model. While decisions are made strictly based on the decision-level states, the evaluation-level value function is used to discover value inconsistencies, indicating that a significant aspect of the state space is not represented in the evaluation-level state model. The determination of inconsistencies here relies on the fact that the optimal value function in a BPMDP, V_P^*, is within a fixed bound of the optimal value function, V^*, on the underlying MDP [Givan *et al.*, 1997].

Inconsistencies are discovered when the evaluation-level value for a state significantly exceeds the value of the corresponding state at the decision level. In this case, the action producing the higher value is included for the corresponding block at the decision level and the block is refined with this action to fulfill Equations 8 and 9 as illustrated in Figure 2.

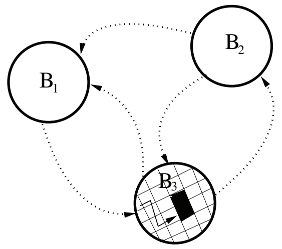

Figure 2: Decision-level model with 3 initial blocks (B_1, B_2, B_3) where block B_3 has been further refined.

4 Experiments

To evaluate the approach, it has been implemented on the Urban Combat Testbed (UCT,), a computer game (http://gameairesearch.uta.edu). For the experiments presented here, the agent is given the abilities to move through the environment shown in Figure 3 and to retrieve and deposit objects.

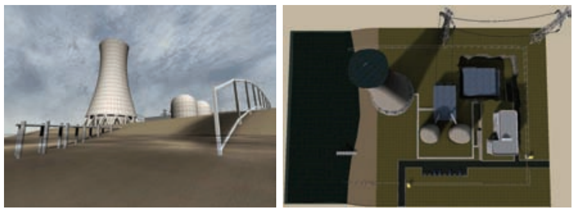

Figure 3: Urban Combat Testbed (UCT) domain.

The state is here characterized by the agent's pose as well as by a set of local object precepts, resulting in an effective state space with $20,000$ states.

The agent is first presented with a reward function to learn to move to a specific location. Once this task is learned, subgoals are extracted by generating random sample trajectories as shown in Figure 4.

As the number of samples increases, the system identifies an increasing number of subgoals until, after $2,000$ samples,

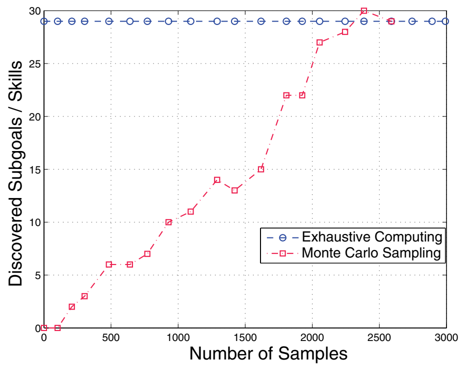

Figure 4: Number of subgoals discovered using sampling.

all 29 subgoals that could be found using exhaustive calculation have been captured.

Once subgoals are extracted, subgoal options, o_i, are learned and termination concepts, C_{t,o_i} and probabilistic outcome predictors, $C_{p,o_i,x}$ are generated. These subgoal options and the termination and prediction concepts are then transferred to the next learning tasks.

The system then builds a hierarchical BPSMDP system model where the decision-level only utilizes the learned subgoal actions while the evaluation-level model is built for all available actions. On this model, a second task is learned where the agent is rewarded for retrieving a flag (from a different location than the previous goal) and return it to the home base. During learning, the system augments its decision-level state representation to allow learning of a policy that is within a bound of optimal as shown in Figure 6.

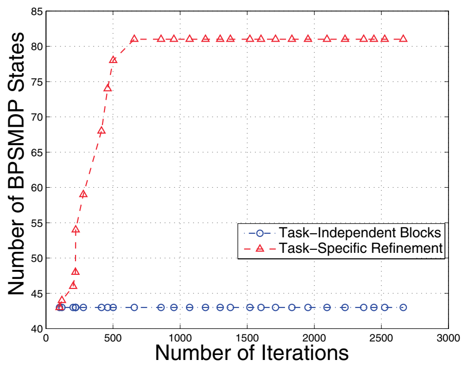

Figure 5: Size of the decision-level state representation.

Figure 5 shows that the system starts with an initial state representation containing 43 states. During learning, as value function inconsistencies are found, new actions and state splits are introduced, eventually increasing the decision-level state space to 81 states. On this state space, a bounded optimal policy is learned as indicated in Figure 6. This graph compares the learning performance of the system against a learner that only transfers the discovered subgoal options and a learner without any transfer mechanism. These graphs show a transfer ratio[2] of ≈ 2.5 when only subgoal options are transferred, illustrating the utility of the presented subgoal criterion. Including the representation transfer and hierarchical BPSMDP learning approach results in significant further improvement with a transfer ratio of ≈ 5.

[2]The transfer ratio is the ratio of the area over the learning curve between the no-transfer and the transfer learner.

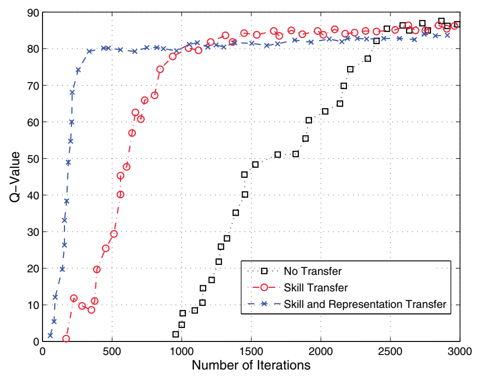

Figure 6: Learning performance with and without skill and representation/concept transfer.

5 Comparison with MAXQ

Dietterich [Dietterich, 2000] developed an approach to hierarchical RL called the MAXQ Value Function Decomposition, which is also called the MAXQ method. Like options and HAMs, this approach relies on the theory of SMDPs. Unlike options and HAMs, however, the MAXQ approach does not rely directly on reducing the entire problem to a single SMDP. Instead, a hierarchy of SMDPs is created whose solution can be learned simultaneously. The MAXQ approach starts with a decomposition of a core MDP M into a set of subtasks $\{M_0, \ldots, M_n\}$. The subtasks form a hierarchy with M_0 being the root subtask, which means that solving M_0 solves M. Actions taken in solving M_0 consist of either executing primitive actions or policies that solve other subtasks, which can in turn invoke primitive actions or policies of other subtasks, etc.

Each subtask, M_i, consists of three components. First, it has a subtask policy, p_i, that can select other subtasks from the set of M_i 's children. Here, as with options, primitive actions are special cases of subtasks. We also assume the subtask policies are deterministic. Second, each subtask has a termination predicate that partitions the state set, s, of the core MDP into s_i, the set of active states in which M_i's policy can execute, and t_i, the set of termination states which, when entered, causes the policy to terminate. Third, each subtask m_i has a pseudo-reward function that assigns reward values to the states in t_i. The pseudo-reward function is only used during learning.

Figure 7 shows the comparison between the MAXQ decomposition and the learning of an SMDP with the sampling-base subgoal discovery but without action-dependent partitioning. This experiment illustrates that MAXQ will outperform an SMDP with options to the subgoals that are discovered by sampling-based subgoal discovery. The reason for this is that while subgoals are hand designed in the MAXQ decomposition, the sampling-based method is fully autonomous and does not rely on human decision. As a result, subgoal dis-

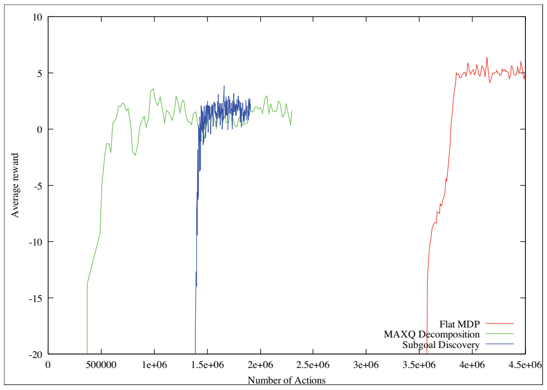

Figure 7: Comparison of policies derived with the MAXQ method and with a SMDP with sampling-based subgoal discovery.

covery generates additional subgoal policies that are not required for the task at hand and might not find the optimal option set. Figure 8 illustrates the comparison between learning time in MAXQ and the BPSMDP constructed by the action-dependent partitioning method. This experiment shows that action-dependent partitioning can significantly outperform the MAXQ decomposition since it constructs state and temporal abstractions resulting in a more abstract state space. In this form, it can transfer the information contained in previously learned policies for solving subsequent tasks.

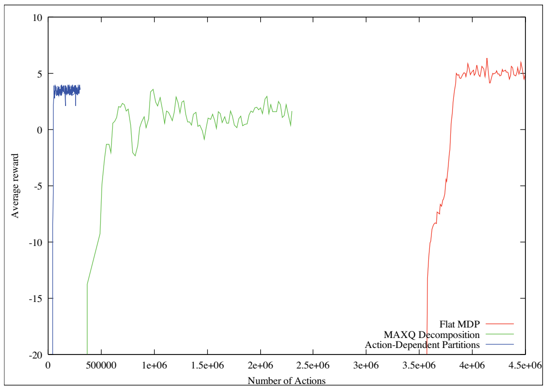

Figure 8: Comparison of policies derived with the MAXQ method and action-dependent partitioning with autonomous subgoal discovery.

6 Conclusion

Most artificial learning agents suffer from the inefficient reuse of acquired control knowledge in artificial. To address this deficiency, the learning approach presented here provides a mechanism which extracts and transfers control knowledge in the form of potentially useful skill and corresponding representation concepts to improve the learning performance on subsequent tasks. The transferred knowledge is used to construct a compact state space hierarchy that captures the important aspects of the environment in the context of the agent's capabilities and thus results in significant improvements in learning performance.

References

[Ando and Zhang, 2004] Rie K. Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. Technical Report RC23462, IBM T.J. Watson Research Center, 2004.

[Asadi and Huber, 2005] M. Asadi and M. Huber. Accelerating Action Dependent Hierarchical Reinforcement Learning Through Autonomous Subgoal Discovery. In *Proceedings of the ICML 2005 Workshop on Rich Representations for Reinforcement Learning*, 2005.

[Boutilier *et al.*, 1999] C. Boutilier, T. Dean, and S. Hanks. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

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

[Givan *et al.*, 1997] R. Givan, T. Dean, , and S. Leach. Model Reduction Techniques for Computing Approximately Optimal Solutions for Markov Decision Processes. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 124–131, San Francisco, CA, 1997. Morgan Kaufmann Publishers.

[Goel and Huber, 2003] S. Goel and M. Huber. Subgoal Discovery for Hierarchical Reinforcement Learning Using Learned Policies. In *Proceedings of the 16th International FLAIRS Conference*, pages 346–350. AAAI, 2003.

[Kim and Dean, 2003] K. Kim and T. Dean. Solving Factored MDPs using Non-Homogeneous Partitions. *Artificial Intelligence*, 147:225–251, 2003.

[Marthi *et al.*, 2005] Bhaskara Marthi, Stuart Russell, David Latham, and Carlos Guestrin. Concurrent hierarchical reinforcement learning. In *International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[Marx *et al.*, 2005] Zvika Marx, Michael T. Rosenstein, and Leslie Pack Kaelbling. Transfer leraning with an ensemble of background tasks. In *NIPS 2005 Workshop on Transfer Learning*, Whistler, Canada, 2005.

[Sutton *et al.*, 1999] R.S. Sutton, D. Precup, and S. Singh. Between MDPs and Semi-MDPs: Learning, Planning, and Representing Knowledge at Multiple Temporal Scales. *Artificial Intelligence*, 112:181–211, 1999.

[Taylor and Stone, 2005] Matthew E. Taylor and Peter Stone. Behavior transfer for value-function-based reinforcement learning. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53–59, New York, NY, July 2005. ACM Press.