Reinforcement Learning of Hierarchical Skills on the Sony Aibo robot

Vishal Soni and Satinder Singh
Computer Science and Engineering
University of Michigan, Ann Arbor
{*soniv, baveja*}*@umich.edu*

***Abstract*— Humans frequently engage in activities for their own sake rather than as a step towards solving a specific task. During such behavior, which psychologists refer to as being intrinsically motivated, we often develop skills that allow us to exercise mastery over our environment. Reference [7] have recently proposed an algorithm for intrinsically motivated reinforcement learning (IMRL) aimed at constructing hierarchies of skills through self-motivated interaction of an agent with its environment. While they were able to successfully demonstrate the utility of IMRL in simulation, we present the first realization of this approach on a real robot. To this end, we implemented a control architecture for the Sony-AIBO robot that extends the IMRL algorithm to this platform. Through experiments, we examine whether the Aibo is indeed able to learn useful skill hierarchies.**

***Index Terms*— Reinforcement Learning, Self-Motivated Learning, Options**

I. Introduction

Reinforcement learning [8] has made great strides in building agents that are able to learn to solve *single* complex sequential decision-making tasks. Recently, there has been some interest within the reinforcement learning (RL) community to develop learning agents that are able to achieve the sort of broad competence and mastery that most animals have over their environment ([3], [5], [7]). Such a broadly competent agent will possess a variety of skills and will be able to accomplish many tasks. Amongst the challenges in building such agents is the need to rethink how reward functions get defined. In most control problems from engineering, operations research or robotics, reward functions are defined quite naturally by the domain expert. For the agent the reward is then some extrinsic signal it needs to optimize. But how does one define a reward function that leads to broad competence? A significant body of work in psychology shows that humans and animals have a number of intrinsic motivations or reward signals that could form the basis for learning a variety of useful skills (see [1] for a discussion of this work). Building on this work in psychology as well as on some more recent computational models ([2]) of intrinsic reward, [7] have recently provided an initial algorithm for intrinsically motivated reinforcement learning (IMRL). They demonstrated that their algorithm, henceforth the IMRL algorithm, is able to learn a hierarchy of useful skills in a simple simulation environment. The main contribution of this paper is an empirical evaluation of the IMRL algorithm on a physical robot (the Sony-Aibo). We introduce several augmentations to the IMRL algorithm to meet the challenges posed by working in a complex and real-world domain. We show that our augmented IMRL algorithm lets the Aibo learn a hierarchy of useful skills.

The rest of this paper is organized as follows. We first present the IMRL algorithm, then describe our implementation of it on the Aibo robot, and finally we present our results.

II. Intrinsically Motivated Reinforcement Learning

The IMRL algorithm builds on two key concepts: a *reward mechanism* internal to the agent, and the *options framework* for representing temporally abstract skills in RL developed by [9]. Lets consider each concept in turn.

In the IMRL view of an agent, its environment is factored into an external environment and an internal environment. The critic is part of the internal environment and determines the reward. Typically in RL the reward is a function of external stimuli and is specifically tailored to solving the single task at hand. In IMRL, the internal environment also contains the agent's intrinsic motivational system which should be general enough that it does not have to be redesigned for different problems. While there are several possibilities for what an agent might consider intrinsically motivating or rewarding, the current instantiation of the IMRL algorithm designs intrinsic rewards around novelty, i.e., around unpredicted but salient events. We give a more precise description of novelty as used in IMRL when we describe our algorithm.

An option (in RL) can be thought of as a temporally extended action or skill that accomplishes some subgoal. An option is defined by three quantities: a policy that directs the agent's behavior when executing the option, a set of initiation states in which the option can be invoked, and termination conditions that define when the option terminates (Figure 1). Since an option-policy can be comprised not only of primitive actions but also of other options, this framework allows for

Option Definition:

- *Initiation Set* $\mathcal{I} \subseteq \mathcal{S}$ which specifies the states in which the option is available.
- *Option Policy* $\pi : \mathcal{I} \times \mathcal{A} \rightarrow \{0, 1\}$ which specifies the probability of taking action a in state s for all $a \in \mathcal{A}, s \in \mathcal{I}$.
- *Termination condition* $\beta : \mathcal{S} \rightarrow \{0, 1\}$ which specifies the probability of the option terminating in state s for all $s \in \mathcal{S}$.

Fig. 1. Option Definition. See text for details.

the development of hierarchical skills. The following two components of the overall option framework are particularly relevant to understanding IMRL:

- *Option models:* An option model predicts the probabilistic consequences of executing the option. As a function of the state s in which the option o is initiated, the model gives the (discounted) probability, $P^o(s'|s)$, for all s' that the option terminates in state s', and the total expected (discounted) reward $R^o(s)$ expected over the option's execution. Option models can usually be learned (approximately) from experience with the environment as follows: $\forall x \in \mathcal{S}$, current state s_t, next state s_{t+1}

$$\begin{aligned} P^o(x|s_t) &\overset{\alpha}{\leftarrow} [\gamma(1-\beta^o(s_{t+1})P^o(x|s_{t+1}) \\ &\quad +\gamma\beta^o(s_{t+1})\delta_{s_{t+1}x}] \\ R^o(s_t) &\overset{\alpha}{\leftarrow} [r^e_t + \gamma(1-\beta^o(s_{t+1}))R^o(s_{t+1})] \end{aligned}$$

  where and α is the learning rate, β^o is the termination condition for option o, γ is the discount factor, δ is the Kronecker delta, and r^e_t is the extrinsic reward. Note that equations of the form $x \overset{\alpha}{\leftarrow} [y]$ are short for $x \leftarrow (1-\alpha)x + \alpha[y]$.
- Intra-option learning methods: These methods allow for all options consistent with a current primitive action to be updated simultaneously in addition to the option that is being currently executed. This greatly speeds up learning of options. Specifically, if a_t is the action executed at time t in state s_t, the option Q-values for option o are updated according to: $\forall$options $o, s_t \in I^o$

$$\begin{aligned} Q^o(s_t, a_t) &\overset{\alpha}{\leftarrow} [r^e_t + \gamma\ (\beta^o(s_{t+1}) \times \lambda^o) \\ &\quad +\gamma(1-\beta^o(s_{t+1})) \times \max_{a \in A \cup O} Q^o(s_{t+1}, a)] \end{aligned}$$

$$\begin{aligned} &\forall\text{options } o',\ s_t \in I^{o'}, o \neq o' \\ &Q^o(s_t, o') \overset{\alpha}{\leftarrow} R^{o'}(s_t) + \sum_{x \in S} P^{o'}(x|s_t)[\beta^o(x) \times \lambda^o \\ &\quad +((1-\beta^o(x)) \times \max_{a \in A \cup O} Q^o(x, a))] \end{aligned}$$

where λ^o is the terminal value for option o.

Next we describe the IMRL algorithm. The agent continually acts according the ϵ-greedy policy with respect to

Loop forever
Current state s_t, next state s_{t+1}
current primitive action a_t
current option o_t,
extrinsic reward r^e_t, intrinsic reward r^i_t

If s_{t+1} is a salient event *e*
  If option for *e* does not exist
  Create option o_e
    Add s_t to the initiation set I^{o_e} of o_e
    Make s_{t+1} the termination state for o_e
  Determine intrinsic reward
    $r^i_{t+1} = \tau[1 - P^{o_e}(s_{t+1}|s_t)]$

Update option models $\forall$ learned options $o \neq o_e$
  If $s_{t+1} \in I^o$, then add s_t to initiation set I^o
  If a_t is greedy option for o in state s_t
    update the *option model* of o

Update the Behavior Q-value function:
  $\forall$ primitive options, do Q-Learning update
  $\forall$ learned options, do SMDP-planning update

$\forall$ learned options
  Update the option Q-Value function

Choose the next option to execute, a_{t+1}
Determine next extrinsic reward, r^e_{t+1}
Set $s_t \leftarrow s_{t+1}$
$a_t \leftarrow a_{t+1}$;
$r^e_t \leftarrow r^e_{t+1}$; $r^i_t \leftarrow r^i_{t+1}$

Fig. 2. The IMRL algorithm.

a evolving behavior Q-value function. The agent starts with an initial set of primitive actions (some of which may be options). The agent also starts with a hardwired notion of salient or interesting events in its environment. The first time a salient event is experienced the agent initiates in its knowledge base an option for learning to accomplish that salient event. As it learns the option for a salient event it also updates the option-model for that option. Each time a salient event is encountered, the agent gets an internal reward in proportion to the error in the prediction of the salient event from the associated option model. Early encounters with the salient event generate a lot of reward because the associated option model is wrong. This leads the agent's behavior Q-value function to learn to accomplish the salient event which in turn improves both the option for achieving the salient event as well as its option model. As the option model improves the reward for the associated salient event decreases

and the agent's action-value function no longer takes the agent there. The agent, in effect, gets "bored" and moves on to other interesting events. Once an option or equivalently skill has been learned and is in the knowledge base of the agent, it becomes available as an action to the agent. This in turn enables more sophisticated behaviors on the part of the agent and leads to the discovery of more challenges to achieve salient events. Over time this builds up a hierarchy of skills. The details of the IMRL algorithm are presented in Figure 2 in a more structured manner.

One way to view IMRL is as a means of semi-automatically discovering options. There have been other approaches for discovering options, e.g., [4]. The advantage of the IMRL approach is that, unlike previous approaches, it learns options outside the context of any externally specified learning problem and that there is a kind of self-regulation built into it so that once an option is learned the agent automatically focuses its attention on things not yet learned.

As stated in the Introduction, thus far IMRL has been tested on a simple simulated world. Our goal here is to extend IMRL to a complex robot in the real world. To this end, we constructed a simple 4 feet x 4 feet play environment for our Sony-Aibo robot containing 3 objects: a pink ball, an audio speaker that plays pure tones and a person that can also make sounds by clapping or whistling or just talking. The experiments will have the robot learn skills such as acquire the ball, approach the sound, and the putting the two together to fetch the ball to the sound or perhaps to the person, and so on. Before presenting our results, we describe some details of our IMRL implementation on the Aibo robot.

III. IMRL ON AIBO

In implementing IMRL on the Aibo, we built a modular system in which we could experiment with different kinds of perceptual processing and different choices for primitive actions in a relatively plug-and-play manner without introducing significant computational and communication latency overhead. We briefly describe the modules in our system.

a) Perception Module:: This module receives sensory input from the Aibo and parses it into an observation vector for the other modules to use. At present, our implementation filters only simple information from the sensors (such as percentage of certain colors in the Aibo's visual field, intensity of sound, various pure tones, etc.). Work is underway to extract more complex features from audio (eg. pitch) and visual data (eg. shapes). Clearly the simple observation vector we extract forms a very coarsely abstracted state representation and hence the learning problem faced by IMRL on Aibo will be heavily non-Markovian. This constitutes our first challenge in moving to a real robot. Will IMRL and in particular the option-learning algorithms within it be able to cope with the non-Markovianness?

- *Explore*: Look for Pink Ball. Terminate when Ball is in visual field.
- *Approach Ball*: Walk towards Ball. Terminate when Ball is near.
- *Capture Ball*: Slowly try to get between between fore limbs.
- *Check Ball*: Angle neck down to check if ball is present between the fore limbs.
- *Approach Sound*: Walk towards sound of a specific tone. Terminate when the sound source is near.

Fig. 3. A list of primitive options programmed on the Aibo. See text for details.

b) Options Module:: This module stores the primitive options available to the agent. When called upon by the Learning module to execute some option it computes what action the option would take in the current state and calls on the low-level motor control primitives implemented in Carnegie Mellon University's Robo-Soccer code with the parameters needed to implement the action. A list of some key options with a brief description of each is provided in Figure 3. This brings up another challenge to IMRL. Many of the options of Figure 3 contain parameters whose values can have dramatic impact on the performance of those options. For example, the option to Approach Ball has a parameter that determines when the option terminates; it terminates when the ball occupies a large enough fraction of the visual field. The problem is that under different lighting conditions the portion of the ball that looks pink will differ in size. Thus the *nearness-threshold* cannot be set a priori but must be learned for the lighting conditions. Similarly, the Approach Sound option has a nearness-threshold based on intensity of sound that is dependent on details of the environment. One way to deal with these options with parameters is to treat each setting of the parameter as defining a distinct option. Thus, Approach Ball with nearness-threshold 10% is a different option than Approach Ball with nearness-threshold of 20%. But this will explode the number of options available to the agent and thus slow down learning. Besides, treating each parameter setting as a distinct option ignores the shared structure across all Approach Ball options. As one of our augmentations to IMRL we treated the parameterized options as a single option and learned the parameter values using a hill-climbing approach. This use of hill-climbing to adapt the parameters of an option while using RL to learn the option policy and option-model is a novel approach to extending options to a real robot. Indeed, to the best of our knowledge there hasn't been much work in using options on a challenging (not lookuptable) domain.

c) Learning Module:: The Learning module implements the IMRL algorithm defined in the previous section.

- Ball Status = {lost, visible, near, **captured first time**, captured, capture-unsure}
- Destinations = {near no destination, **near sound first time**, near sound, near experimenter, **ball taken to sound**}

Fig. 4. State variables for the play environment. Events in bold are salient. A ball status of 'captured' indicates that the Aibo is aware the ball is between its fore limbs. The status 'captured first time' indicates that ball was previously not captured but is captured now (similarly, the value 'near sound first time' is set when the Aibo transitions from not being near sound to being near it). The status of 'maybe capture' indicates that the Aibo is unsure whether the ball is between its limbs. If the Aibo goes for a specific length of time without checking for the ball, the ball status transitions to being 'capture-unsure'.

IV. Experiments

Recall that for our experiments, we constructed a simple 4'x4' play environment containing 3 objects: a pink ball, an audio speaker that plays pure tones and a person that can also make sounds by clapping or whistling or just talking. The Explore (primitive) option turned the Aibo in place clockwise until the ball became visible or until the Aibo had turned in circle twice. The Approach Ball option had the Aibo walk quickly to the ball and stop when the nearness-threshold was exceeded. The Approach Sound option used the difference in intensity between the two microphones in the two ears to define a turning radius and forward motion to make the Aibo move toward the sound until a nearness-threshold was exceeded. Echoes of the sound from other objects in the room as well as from the walls or the moving experimenter made this a fairly noisy option. The Capture Ball option had the Aibo move very slowly to get the ball between the fore limbs; walking fast would knock the ball and often get it rolling out of sight.

We defined an externally rewarded task in this environment as the task of acquiring the ball, taking it to the sound and then taking it to the experimenter in sequence. Upon successful completion of the sequence, the experimenter would pat the robot on the back (the Aibo has touch sensors there), thereby rewarding it and then pick up the robot and move it (approximately) to an initial location. The ball was also moved to a default initial location at the end of a successful trial. This externally rewarded task is a fairly complex one. For example, just the part of bringing the ball to sound involves periodically checking if the ball is still captured, for it tends to roll away from the robot; if it is lost then the robot has to Explore for ball and Approach Ball and then Capture Ball again, and then repeat. Figure 9 shows this complexity visually in a sequence of images taken from a video of the robot taking the ball to sound. In image 5 the robot has lost the ball and has to explore, approach and capture the ball again (images 6 to 9) before proceeding towards sound again (image 10). We report the results of three experiments.

Experiment 1: We had the Aibo learn with just the external reward available and no intrinsic reward. Consequently, the Aibo did not learn any new options and performed standard Q-learning updates of its (primitive) option Q-value functions. As Figure 5(a) shows, with time the Aibo gets better at achieving its goal and is able to do this with greater frequency. The purpose of this experiment is to serve as a benchmark for comparison. In the next two experiments, we try to determine two things. (a) Does learning options hinder learning to perform the external task when both are performed simultaneously? And (b), How 'good' are the options learned? If the Aibo is bootstrapped with the learned options, how would it perform in comparison with the learning in Experiment 1?

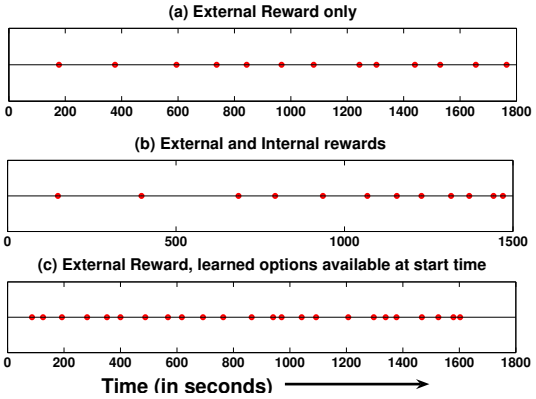

Fig. 5. A Comparison of learning performance. Each panel depicts the times at which the Aibo was successfully able to accomplish the externally rewarded task. See text for details.

Experiment 2: We hardwired the Aibo with three salient events: (a) acquiring the ball, (b) arriving at sound source, and (c) arriving at sound source with the ball. Note that (c) is a more complex event to achieve than (a) or (b), and also that (c) is not the same as doing (a) and (b) in sequence for the skill needed to approach ball without sound is much simpler than approaching sound with ball (because of the previously noted tendency of the ball to roll away as the robot pushes it with its body). The Aibo is also given an external reward if it brings the ball to the sound source and then to the experimenter in sequence. In this experiment the Aibo will learn new options as it performs the externally rewarded task. The robot starts by exploring its environment randomly. Each first encounter with a salient event initiates the learning of an option for that event. For instance, when the Aibo first captures the ball, it sets up the data structures for learning the Acquire Ball option. As the Aibo explores its environment, all the options and their models are updated via intra-option learning.

Figure 6 presents our results from Experiment 2. Each panel in the figure shows the evolution of the intrinsic reward

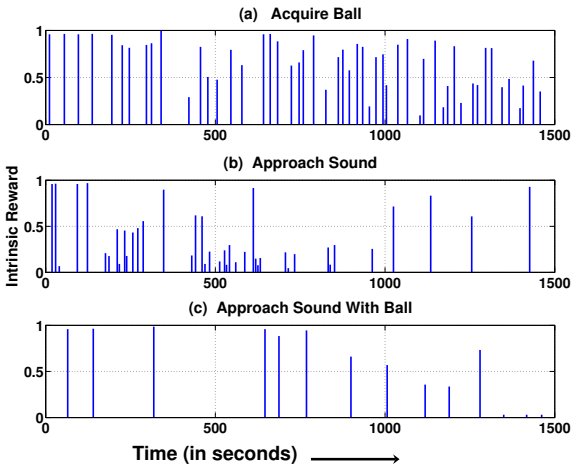

Fig. 6. Evolution of intrinsic rewards as the Aibo learns new options. See text for details. Solid lines depict the occurrence of salient events. The length of these lines depicts the associated intrinsic reward.

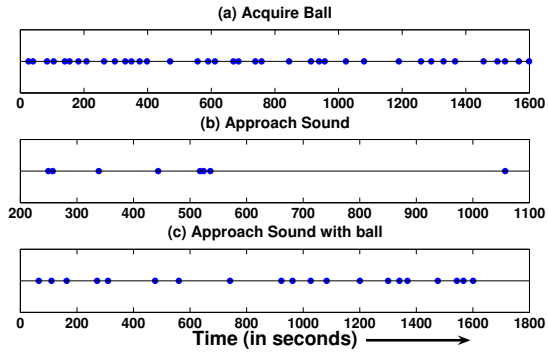

Fig. 7. Performance when the Aibo bootstrapped with the options learned in Experiment 2. The markers indicate the times at which the Aibo was able to achieve each event.

associated with a particular salient event. Each encounter with a salient event is denoted by a vertical bar and the height of the bar denotes the magnitude of the intrinsic reward. We see that in the early stages of the experiment, the event of arriving at sound and the event of acquiring the ball tend to occur more frequently than the more complex event of arriving at sound with the ball. With time however, the intrinsic reward from arriving at sound without the ball starts to diminish. The Aibo begins to get bored with it and moves on to approaching sound with the ball. In this manner, the Aibo learns simpler skills before more complex ones. Note, however, that the acquire ball event continues to occur frequently throughout the learning experience because it is an intermediate step in taking the ball to sound.

Figure 5(b) denotes the times in Experiment 2 at which the Aibo successfully accomplished the externally rewarded task while learning new options using the salient events. Comparing this with the result from Experiment 1 (Figure 5(a)) when the Aibo did not learn any options, we see that the additional onus of learning new options does not significantly impede the agent from learning to perform the external task, indicating that we do not loose much by having the Aibo learn options using IMRL. The question that remains is, do we gain anything? Are the learned options actually effective in achieving the associated salient events?

Experiment 3: We had the Aibo learn, as in the Experiment 1, to accomplish the externally rewarded task without specifying any salient events. However, in addition to the primitive options, the Aibo also had available the learned options from Experiment 2. In Figure 7, we see that the Aibo achieved all three salient events quite early in the learning process. Thus the learned options did in fact accomplish the associated salient events and furthermore these options are indeed accurate enough to be used in performing the external task. Figure 5(c) shows the times at which the external reward was obtained in Experiment 3. We see that the Aibo was quickly able to determine how to achieve the external goal and was fairly consistent thereafter in accomplishing this task. This result is encouraging since it shows that the options it learned enabled the Aibo to bootstrap more effectively.

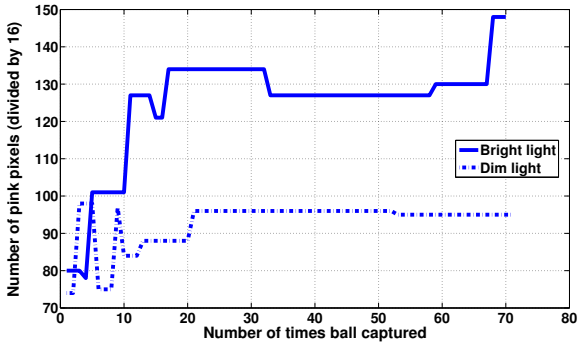

Fig. 8. A comparison of the *nearness threshold* as set by the hill-climbing component in different lighting conditions. The solid line represents the evolution of the threshold under bright lighting, and the dashed line represents the same under dimmer conditions. Here we see that the hill climbing is, at the very least, able to distinguish between the lighting conditions and tries to set a lower threshold for case with dim lighting.

Finally, we examined the effectiveness of the hill-climbing component that determines the setting of the *nearness threshold* for the *Approach Object* option. We ran the second experiment, where the Aibo learns new options, in two different lighting conditions - one with bright lighting and the other with considerably dimmer lighting. Figure 8 presents these results. Recall that this threshold is used to determine when the *Approach Object* option should terminate. For a particular lighting condition, a higher threshold means that *Approach Object* will terminate closer since a greater number of pink pixels are required for the ball to be assumed near

Fig. 9. In these images we see the Aibo executing the learned option to take the ball to sound (the speakers). In the first four images, the Aibo endeavors to acquire the ball. The Aibo has learned to periodically check for the ball when walking with it. So in image 5 when ball rolls away from the Aibo, it realizes that the ball is lost. Aibo tries to find the ball again, eventually finding it and taking it to the sound.

(and thus capturable). Also, for a particular distance from the ball, more pink pixels are detected under bright lighting than under dim lighting. The hill-climbing algorithm was designed to find a threshold so that the option terminates as close to the ball as possible without losing it thereby speeding up the average time to acquire the ball. Consequently, we would expect that the nearness threshold should be set lower for dimmer conditions.

V. Conclusions

In this paper we have provided the first successful application of the IMRL algorithm to a complex robotic task. Our experiments showed that the Aibo learned a two-level hierarchy of skills and in turn used these learned skills in accomplishing a more complex external task. A key factor in our success was our use of parameterized options and the use of the hill-climbing algorithm in parallel with IMRL to tune the options to the environmental conditions. Our results form a first step towards making intrinsically motivated reinforcement learning viable on real robots. Greater eventual success at this would allow progress on the important goal of making broadly competent agents (see [6] for a related effort).

As future work, we are building more functionality on the Aibo in terms of the richness of the primitive options available to it. We are also building an elaborate physical playground for the Aibo in which we can explore the notion of broad competence in a rich but controlled real-world setting. Finally, we will explore the use of the many sources of intrinsic reward in addition to novelty that have been identified in the psychology literature.

Acknowledgements

The authors were funded by NSF grant CCF 0432027 and by a grant from DARPA's IPTO program. 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 NSF or DARPA.

References

[1] A. G. Barto, S. Singh, and N. Chentanez. Intrinsically motivated reinforcement learning of hierarchical collection of skills. In *International Conference on Developmental Learning (ICDL)*, LaJolla, CA, USA, 2004.

[2] S. Kakade and P. Dayan. Dopamine: generalization and bonuses. *Neural Netw.*, 15(4):549–559, 2002.

[3] F. Kaplan and P.-Y. Oudeyer. Maximizing learning progress: An internal reward system for development. In *Embodied Artificial Intelligence*, pages 259–270, 2003.

[4] A. McGovern. *Autonomous Discovery of Temporal Abstractions from Interactions with an Environment*. PhD thesis, University of Massachusetts, 2002.

[5] L. Meeden, J. Marshall, and D. Blank. Self-motivated, task-independent reinforcement learning for robots. In *AAAI Fall Symposium on Real-World Reinforcement Learning*, Washington D.C., 2004.

[6] P.-Y. Oudeyer, F. Kaplan, V. Hafner, and A. Whyte. The playground experiment: Task-independent development of a curious robot. In *AAAI Spring Symposium Workshop on Developmental Robotics. To Appear*, 2005.

[7] S. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17*, pages 1281–1288. MIT Press, Cambridge, MA, 2004.

[8] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[9] R. S. Sutton, D. Precup, and S. P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999.