Learning Options through Human Interaction

Kaushik Subramanian, Charles L. Isbell, Andrea L. Thomaz
Georgia Institute of Technology
College of Computing
801 Atlantic Dr., Atlanta, GA 30332
{ksubrama, isbell, athomaz}@cc.gatech.edu

Abstract

Hierarchical Reinforcement Learning solves problems by decomposing them into a set of sub-tasks or *options*. In this paper we develop a method of soliciting options from humans. We show how humans design actions that naturally decompose the problem, making them compatible with the options framework. We instantiate our approach in the Taxi domain and Pac-Man and show that the human-derived options outperform automated methods of option extraction both in terms of optimality and computation time. Our experiments show that human-options are generalizable across the state space of the problem. Further analysis shows that options given by humans are often required to be interrupted before completion. As such, we discuss the potential use of Modular Reinforcement Learning to approach the problem.

1 Introduction

Our research is in the realm of Interactive Machine Learning, which leverages human input for Machine Learning (ML) performance gains. In particular, we are interested in applying ML algorithms in application domains where humans are end users, and can readily provide input. In this paper we address one kind of input that we think humans will be good at providing: task decompositions.

In psychology literature, it has repeatedly been shown that we interpret actions of other humans based on goals rather than specific activities or motion trajectories [Woodward *et al.*, 2001; Gleissner *et al.*, 2000]. From dynamic action, people can find meaningful boundaries between smaller action chunks. Baldwin et al., research the mechanisms by which humans do this kind of parsing and find that human adults show agreement in their decomposition of a continuous task (e.g., watching someone clean in a kitchen) [Baldwin and Baird, 2001]. Thus, we hypothesize that these kinds of task decompositions will be useful for a Reinforcement Learner.

Such decompositions have been studied in great detail in Hierarchical Reinforcement Learning (HRL). HRL takes advantage of the hierarchy present in real-world domains to simplify otherwise complex problems and has been used successfully in the optimization of manufacturing processes [Wang and Mahadevan, 1999], quadruped locomotion [Kolter *et al.*, 2007], and simulated flying [Ryan and Reid, 2000] among others.

Within HRL, there have been several techniques proposed to take advantage of task hierarchies. In this work, we make use of the *options* framework [Sutton *et al.*, 1999]. An option is a temporally extended action. It provides a layer of action abstraction to the underlying domain and offers a principled way of speeding up learning algorithms [Sutton *et al.*, 1999]. In the options framework, the components that make up each option are either designed beforehand or learned automatically. Most automatic methods depend on the ability to construct options by identifying bottleneck states. In large sparse domains, extracting these states can be computationally intensive, and some methods are prone to generating redundant options [Pickett and Barto, 2002]. Furthermore constructing the options a priori by hand, if even possible, is a time-consuming task.

We introduce an interactive method for understanding the human ability to identify meaningful options for a given problem. Our goal is to understand three key questions:

1. **Is the way humans decompose problems consistent with the options framework?**
   In our first experiment, we analyze the ways in which humans decompose two problem domains and show that humans design actions that subdivide these problems in a way that one might define as options for the domain.
2. **Can humans teach options that lead to performance gains?**
   Second, we develop an interaction technique and a learning mechanism that allows humans to teach an agent the subtasks defined in Experiment 1. We instantiate our framework on two domains and show how human options are general across the underlying problem and provide significant speedup compared to automatically generated options, and to directly learning the entire problem.
3. **Are human options designed to execute to completion?**
   We briefly discuss how human options are designed to be interrupted. We propose the use of Modular Reinforcement Learning as an alternative approach to this problem.

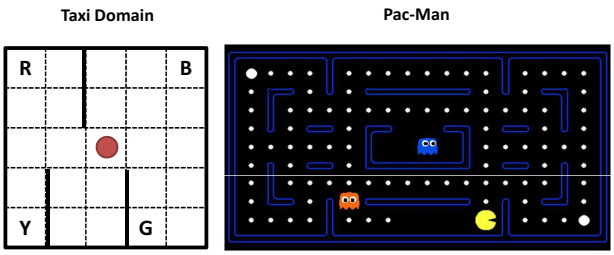


Figure 1: The domains used to illustrate interactive option learning. Taxi (left) and Pac-Man (right).

2 Preliminaries - Reinforcement Learning

A Reinforcement Learning (RL) agent interacts with an environment described by a Markov Decision Process (MDP), a tuple $M = \langle S, A, \mathcal{T}, R, \gamma \rangle$ with states S, actions A, transition function $\mathcal{T} : S \times A \mapsto \Pr[S]$, reward function $R : S \times A \mapsto [R_{min}, R_{max}]$, and discount factor $\gamma \mapsto [0, 1]$. A policy $\pi : S \mapsto A$ is a relation that defines which action should be taken in a particular state. Every state $s \in S$ is represented by n-tuple of features, $s = (f_1, f_2, ..., f_n)$.

Within the framework of Reinforcement Learning, temporally extended actions can be modeled using the options framework [Sutton *et al.*, 1999]. An option consists of three components: $\langle I, \pi, \beta \rangle$ where $I \in S$ is the initiation set, which determines the states from which the option can be started, $\pi : S \times A \mapsto [0, 1]$ is the option's policy, a probabilistic distribution over each s, a pair that the option is defined in and $\beta(s)$ is the termination set which gives the probability of an option terminating in state s. Formally, a set of options defined over an MDP constitutes a Semi-Markov Decision Process (SMDP) [Sutton *et al.*, 1999]. For an SMDP, the Markov policy over options $\mu : S \times O \mapsto [0, 1]$ is the relation that defines which option should be taken in a particular state. Methods of planning and computing the optimal policy with options have been discussed in Sutton *et al.* (1999).

3 Domains

We begin by describing the two domains used in our experiments: Taxi and Pac-Man. Both are shown in Figure 1.

Taxi. In the Taxi domain [Dietterich, 2000], a taxi navigates on a grid with immovable obstacles. The taxi must transport passengers from and to predefined locations. The taxi can move deterministically in one of the four cardinal directions, pick up the passenger when the taxi is in the same cell, and drop off the passenger. The taxi starts in a random cell of a 5x5 grid. Every state is defined by a set of 10 features, including the taxi location in the grid, location of the passenger, location of the destination, whether passenger has been picked up, whether the taxi is at the passenger location, among others. The domain has a total of 500 states. There is a reward of 0 for picking up the passenger, a reward of $+1$ for completing the task and -1 everywhere else.

Pac-Man. In Pac-Man, the agent navigates through a maze and eats the white dots. Pac-Man must avoid the two ghosts who can kill him. Upon eating one of two power pills Pac-Man can temporarily eat the ghosts who change color and direction, and slow down. One can also earn extra points by eating fruit that sometimes appears in the maze. The game ends when three lives have been lost. The features for Pac-Man include agent position, ghost position, nearest food pellet, position of the power pill and a ghost runaway timer. We used a discretized grid of size 9×18.

Taxi Domain	
Button Name	Percentage of participants who gave this button
Go to passenger and pickup	60%
Go to destination and dropoff	60%
Go to passenger	40%
Go to destination	40%
Pickup/Dropoff	40%
Move away from obstacles	20%
Pac-Man	
Go to the closest food	100%
Avoid Ghost	100%
Go to the nearest power pellet	100%
Eat the ghost	40%

Table 1: Human-defined options for the two domains

4 Human Task Decomposition

In the first experiment we focus on how humans decompose the Taxi and Pac-Man domains. Our hypothesis is that when humans can define new actions to solve these tasks, they choose ones that decompose the problem into sub-tasks that represent useful options.

Our study involved 10 volunteers from the campus community. Each participant was assigned one of the two domains. We described the domain to the participant as an interactive game with buttons representing low-level actions. They were allowed to play the game until they were familiar with the controls. Afterwards, each participant was asked to suggest modified buttons that would make winning the game "easier and faster". They were restricted to creating a fixed number of buttons, at most 3 for Taxi and at most 4 for Pac-Man. Table 1 summarizes the responses from this study. Each button name can be thought of as a sequence of temporally extended low-level actions, thereby making them very similar to options. This suggests that human task decomposition is performed in a manner consistent with the options learning framework. We will henceforth refer to these buttons as human-options.

5 Soliciting Human-Options

The options given by people depend on specific features or attributes of the domain. For example, the button "go to passenger" can be considered as a generalized option that depends on the passenger location and is not a fixed sequence of primitive actions. This form of option-specific feature selection is an important component of our interactive framework.

For each option, we ask each person to provide 1) option-feature dependencies and 2) example trajectories, solicited sequentially. Our feature selection interaction requires that

the state features of the domain be represented in a manner that is verbalizable to the human. By allowing the human to select feature dependencies for an option, we not only take advantage of their demonstrations, but also let them explicitly help define how an option should generalize across the state space. Through a series of *Yes* or *No* questions, as shown below, we setup the domain for the demonstration phase of teaching:

> Does the "Option-Name-X" option depend on "Feature-Name-1"? - Yes
>
> Does the "Option-Name-X" option depend on "Feature-Name-2"? - Yes
>
> Does the "Option-Name-X" option depend on "Feature-Name-3"? - No

Then, with only the option-specific features activated, the participant provides an example of the expected policy of the option. Given a sequence of randomly chosen start states, they show sample trajectories by playing the game. These trajectories are state, action and reward pairs, $s_0a_0r_0, s_1a_1r_1, ...s_na_nr_n$ from start to end, sampled from the expected policy of the option. The states are represented as a vector of features $s = (f_1, f_2, ..., f_n)$. The human uses the primitive actions (those available in the domain) when providing trajectories.

6 Learning Human-Option Components

Using the trajectories provided by the volunteers for each human-option, we now need to learn the individual components: a model of the option's policy, π, and a model of the initiation set, *I* and termination set, β. We use the visited states as part of the initiation set, the actions taken in these states to represent the option's policy, the end state as part of the termination set and the reward acquired to calculate the average return for executing the option.

We setup three decision tree learners, one for each of the three components of the human-option. The input to each decision tree learner is a sequence of states (represented by a vector of features) followed by either the action taken or a binary label indicating its membership in the initiation set or to the termination set. Given a set of N such examples for an option, the policy, the initiation set and the termination set model learned from the decision tree are tested to assess their respective optimality. In order to test the computed option policy, we use the learned policy model to generate sample trajectories. The average return from these sample trajectories, R_{O^π} is compared to the average return acquired form the human demonstrations $R^H_{O^\pi}$ and this threshold indicates the need for further examples from the human. The cross validation (ten-fold) error of the decision tree model for the initiation and termination set, E_{O_I} and E_{O_β} respectively, indicate how representative the visited states are of the entire state space. A higher value indicates that more of the state space is required to be explored.

In this manner, we continue to acquire trajectories from different start states from the human until the criteria for the policy, the initiation set and termination set are satisfied. This interactive process is repeated for all the options.

7 Evaluation

To test our mechanism for learning human options through interaction, we collected data from 6 volunteers (from the same set of volunteers used in the task decomposition experiment). Each participant was given the opportunity to teach a set of options for one of the domains. This set was determined as the majority of the buttons given by the other 4 participants. For the Taxi domain, 2 buttons were used - "go to passenger and pickup", "go to destination and dropoff" and for Pac-Man, 3 buttons were used - "go to the closest food", "avoid ghost" and "go to the nearest power pellet".

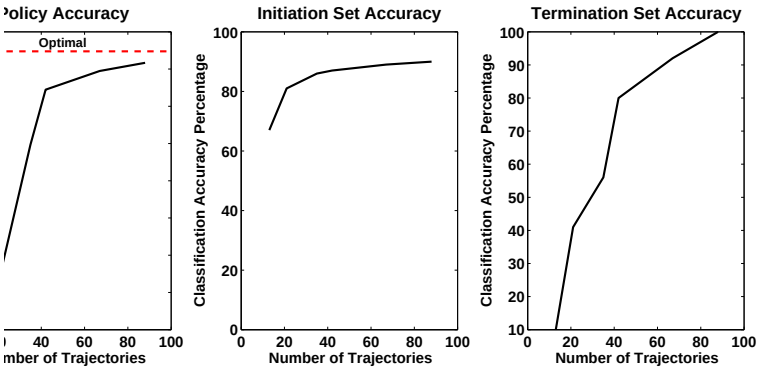
7.1 Optimality of Human-Options

We perform an analysis on the optimality of the human-options for selected options chosen from Table 1. Figures 2(a) and 2(b) show the effect of the number of human trajectories on the policy and classification accuracy for the two domains. The policy accuracy is measured by sampling trajectories from the learned model and computing the average expected return. The classification accuracy is measured in terms of the how many states were correctly classified versus how many were incorrectly classified with respect to the optimal model.

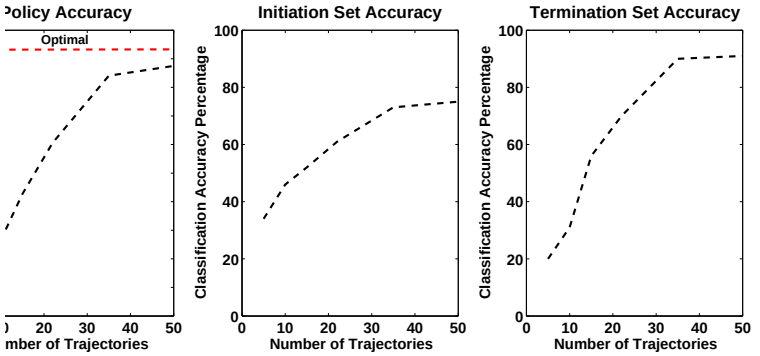
The data shown in Figure 2 is an average over the 6 participants for specific options. We see that the accuracy increases with larger amounts of training data and reaches acceptable levels with over 40 trajectories in both the domains. The policy accuracy indicates the optimality of the learned human-option model. It is worth noting that the classification accuracy in both domains does not reach 100%. This can be explained by several reasons - error propagation from the decision tree, sub-optimal human trajectories and most importantly incomplete options. In the Pac-Man domain, the human-options that we used did not cover the entire state space. This included the parts of the state space where the agent could eat the ghost after eating the power pellet. This notion of incomplete options will be discussed later in detail. To reach the maximum number of trajectories, the participants, on average, took no more than 15 mins in Taxi and 20 mins in Pac-Man; therefore, we were able to achieve sufficient accuracy with reasonable interaction time with the human.

7.2 Human vs. Automatic Decomposition Comparison

Next we compare the task decomposition abilities of human-options with automated methods. We extract automated options using the algorithm described in Stolle and Precup (2002). This approach is the first principled algorithm used to extract options in reinforcement learning domains. The automatic option extraction algorithm works by extracting "bottleneck states" in the domain. Bottleneck states refer to the states that are most often visited when performing a set of relevant tasks in the domain. It uses these states as termination states and builds a policy to reach these states. A sample of human-options and the automatically extracted options for the Taxi domain is shown in Figure 3. The checkered flags show the locations of bottleneck states identified by the algorithm as target states.



(a) Optimality of "Go to destination and dropoff" human-option in the Taxi Domain



(b) Optimality of "Go to nearest food pellet" human-option in the Pac-Man domain

Figure 2: Diagram showing Policy and Classification accuracy of the human-option components with increasing number of trajectories

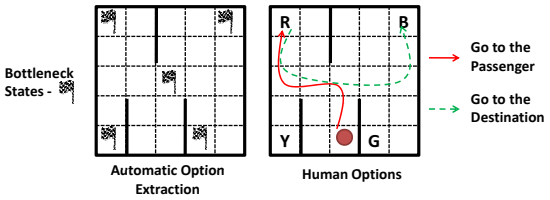


Figure 3: Diagram showing automatic options on the left and human-options on the right, extracted for the Taxi domain

The automated algorithm generated 9 options. Four for going to each of the passenger locations and performing a pickup action, another four to navigate the passenger to one of the four destinations and perform a dropoff and one to reach the cell in the center of the grid. We note that there are three more automated options than the number of primitive actions available. Furthermore, the automated options cover very similar sub-tasks. For example, the four passenger options can be thought of as a single generalized "go to passenger at (x,y) and pickup" option. The human-options, "go to passenger and pickup", "go to destination and dropoff" are shown in Figure 3. The human-options are more general in that the same option is applicable for different passenger and destination locations.

7.3 Human vs. Automatic Performance Comparison

To evaluate the performance of the human-options, we compared it with the automated options as well as using the primitive actions in terms of computation and efficiency. We use Synchronous VI as described in [Precup, 2000] to compute the optimal value function for the options. We sampled the policy acquired and computed the average return. Results of this comparison are seen in Table 2. We observe that the options selected by humans were more efficient than the ones obtained automatically. In terms of computation time, the human-options were able to achieve significant speed-up, particularly in the Pac-Man domain. This can be attributed to the nature of the human-options: more general and fewer in number, making them more desirable. The automated options do not do as well as human-options but they are faster when compared to the primitive actions. We attribute this result to the redundancy in the selection of automated options.

It is clear that there is significant speed-up in convergence when using options; however, the computed value for the options are never as good as when using primitive actions. This can be explained by the use of a sub-optimal model for options during planning [Sutton *et al.*, 1999].

Figure 4 shows the learning rate of the different action sets in terms of the number of steps in an episode of Q-learning for the Taxi domain. The graph shows the speed-up in learning

Model	Computation Time (seconds)	Average Reward of Computed Policy
Taxi Domain		
Primitive actions	17.45	-2.294
Human options	10.45	-4.294
Automated options	25.75	-4.311
Pac-Man		
Primitive actions	600	1890
Human options	60.45	1790
Automated options	120.47	1442

Table 2: Comparative listing of Computation Time and Average Reward using different action sets

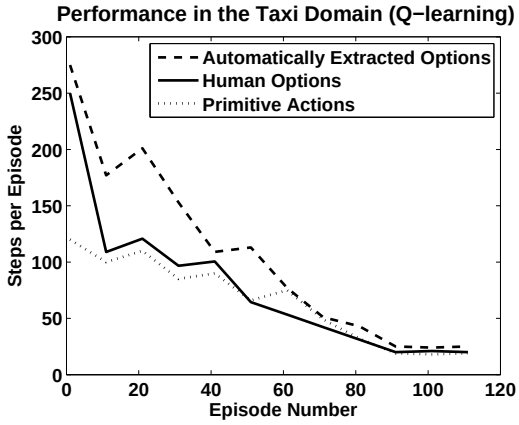


Figure 4: Steps taken per episode with Q-learning on primitive actions, human-options and automatic options for the Taxi domain.

that can be achieved when using human-options when compared to automated options.

8 Discussion

From our experiments there were several insights we gained about learning from human-options. In our first experiment we allowed each human to devise the set of options without receiving any feedback about the options' performance. This leads to a question of whether humans might be inclined to come up with new options or maybe refine their current ones given feedback about their performance. While we did not test this hypothesis for this paper, we consider it to be an important question for future work.

The options given by humans, for both domains, were general, requiring that they be adapted across the state space. Each of these options was dependent on specific features of the domain. The advantage of using such options over automated methods is the reduction in the generation of redundant options. The smaller number of human-options can be attributed to the ability of humans to identify the important "bottleneck" states within the domain. In addition, we observe that the human-options do not necessarily form a complete set in that it is not always possible to learn the optimal policy for the entire domain using only the human-options. This happened in the Pac-Man domain. We can explain this result in part by inferring that the human's reward function is somewhat different than the reward function used by the agent. For example, in Pac-Man, points matter less than survival. The inclusion of an option by the human indicates that it is relevant to the human's reward function.

In our domains, we noticed quite often that options that were learned needed to be terminated before completion. For example, in the Pac-Man domain, the options of "go to the nearest food pellet" and "move away from ghost" (see Table 1) rarely run until completion. At time step t, a decision needs to be made to switch from one option to the other in order to stay alive. This is not unique to our approach. The literature describes multiple methods to deal with this need (see for example, a heuristic based on the value function or on the number of interruptions[Precup, 2000], a parametric form of the termination condition [Comanici and Precup, 2010] and policy iteration [Precup, 2000]). Each method takes advantage of certain structural features of the target domain, so it is not clear whether there is an optimal mechanism for when to interrupt an option for the general case. This is especially true for options that do not have clear termination states such as "move away from ghost".

Given the kinds of options that humans seem to provide, we propose a different way of thinking about interrupting options. Consider such options to be individual modules that are running simultaneously, each vying for control. Learning with such competing modules can be viewed as a Modular Reinforcement Learning problem [Samejima *et al.*, 2003]. We consider this to be research direction worth pursuing for human-generated options.

9 Related Work

Our work is an instance of Interactive Machine Learning. Argall *et al.* (2009) provides a broad overview of the use of humans in the Learning by Demonstration framework. Specifically within the RL domain, Apprenticeship Learning [Abbeel and Ng, 2004], Inverse Reinforcement Learning [Ng and Russell, 2000], TAMER [Knox and Stone, 2009] and Social Learning [Thomaz and Breazeal, 2008] describe ways in which we can use human input to learn an optimal policy for the given domain. These algorithms use primitive actions for planning so can in principal benefit fro the use of temporally extended actions.

Within the scope of Interactive Machine Learning, there has been some recent work on leveraging human information into action abstraction that is close to our approach. Zang *et al.* (2009) describe a method of automatically extracting options from human trajectories. In their work, they assume that the features relevant to actions are already known while our setup extracts these features from human interaction. Comanici and Precup (2010) develop a policy switching algorithm that iteratively decides how to switch from one option policy to another using human examples. They assume that the low-level behavioral policies are given as prior knowl-

edge, while in our work we extract these policies from human examples.

Most of the previous work on learning with options either uses a human to design the options by hand or automatically extracts them. The automatic methods focus on the ability to find subgoals or bottleneck states within the domain. Heuristics for selecting goal states for options have been developed based on frequency of visit on successful trajectories [McGovern and Barto, 2001], relative novelty [Simsek and Barto, 2004], clustering algorithms [Mannor *et al.*, 2004] and others. Work in Konidaris and Barto (2009) uses the concept of options to learn task hierarchies to extract task-relevant skills.

In all these methods, the heuristics may be of limited applicability. Worse, a human expert must understand the domain well enough to design and encode these heuristics. By contrast, automated methods generate numerous options, making them susceptible to option redundancy. Jong *et al.* (2008) highlight some of the disadvantages of planning with such options and their work motivates the need for more generalized options applicable to different parts of the state space. Our approach is aimed at leveraging human information to generate general options that serve to mitigate some of these problems.

10 Conclusions

In this work, we have shown that humans can decompose problems in a way consistent with the options framwork. Further, we can extract these options in such a way that we learn their components. In particular, we were able to efficiently generalize options across the state space and obtain significant speed-up in planning time. We discussed several characteristics of human-options, including introducing the need to allow for continual interruptions.

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[Argall *et al.*, 2009] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[Baldwin and Baird, 2001] D.A. Baldwin and J.A. Baird. Discerning intentions in dynamic human action. *Trends in Cognitive Sciences*, 5(4):171–178, 2001.

[Comanici and Precup, 2010] Gheorghe Comanici and Doina Precup. Optimal policy switching algorithms for reinforcement learning. In *AAMAS*, pages 709–714, 2010.

[Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res. (JAIR)*, 13:227–303, 2000.

[Gleissner *et al.*, 2000] B. Gleissner, A. N. Meltzoff, and H. Bekkering. Children's coding of human action: cognitive factors influencing imitation in 3-year-olds. *Developmental Science*, 3(4):405–414, 2000.

[Jong *et al.*, 2008] Nicholas K. Jong, Todd Hester, and Peter Stone. The utility of temporal abstraction in reinforcement learning. In *AAMAS (1)*, pages 299–306, 2008.

[Knox and Stone, 2009] W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: the tamer framework. In *K-CAP*, pages 9–16, 2009.

[Kolter *et al.*, 2007] J. Zico Kolter, Pieter Abbeel, and Andrew Y. Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In *NIPS*, 2007.

[Konidaris and Barto, 2009] George Konidaris and Andrew G. Barto. Efficient skill learning using abstraction selection. In *IJCAI*, pages 1107–1112, 2009.

[Mannor *et al.*, 2004] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *ICML*, 2004.

[McGovern and Barto, 2001] Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML*, pages 361–368, 2001.

[Ng and Russell, 2000] A Y Ng and S Russell. Algorithms for inverse reinforcement learning. In *In Proc. ICML*, pages 663–670, 2000.

[Pickett and Barto, 2002] Marc Pickett and Andrew G. Barto. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML*, pages 506–513, 2002.

[Precup, 2000] Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, UMass Amherst, 2000.

[Ryan and Reid, 2000] Malcolm R. K. Ryan and Mark D. Reid. Learning to fly: An application of hierarchical reinforcement learning. In *ICML*, pages 807–814, 2000.

[Samejima *et al.*, 2003] Kazuyuki Samejima, Kenji Doya, and Mitsuo Kawato. Inter-module credit assignment in modular reinforcement learning. *Neural Networks*, 16(7):985–994, 2003.

[Simsek and Barto, 2004] Özgür Simsek and Andrew G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *ICML*, 2004.

[Stolle and Precup, 2002] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *Lecture Notes in Computer Science*, pages 212–223, 2002.

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

[Thomaz and Breazeal, 2008] Andrea Lockerd Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artif. Intell.*, 172(6-7):716–737, 2008.

[Wang and Mahadevan, 1999] Gang Wang and Sridhar Mahadevan. Hierarchical optimization of policy-coupled semi-markov decision processes. In *ICML*, pages 464–473, 1999.

[Woodward *et al.*, 2001] A. L. Woodward, J. A. Sommerville, and J. J. Guajardo. How infants make sense of intentional actions. In B. Malle, L. Moses, and D. Baldwin, editors, *Intentions and Intentionality: Foundations of Social Cognition*, chapter 7, pages 149–169. MIT Press, Cambridge, MA, 2001.

[Zang *et al.*, 2009] Peng Zang, Peng Zhou, David Minnen, and Charles Lee Isbell Jr. Discovering options from example trajectories. In *ICML*, page 153, 2009.