Hierarchical State Representation and Action Abstractions in Q-Learning for Agent-Based Herding

Tao Mao and Laura E. Ray

***Abstract*—A primary challenge of agent-based policy learning in complex and uncertain environments is escalating computational complexity with the size of the task space and the number of agents. Nonetheless, there is ample evidence in the natural world that high functioning social mammals learn to solve complex problems with ease. This ability to solve computationally intractable problems stems in part from brain circuits for hierarchical representation of state and action spaces and learned policies arising from these representations. Using such mechanisms for state representation and action abstraction, we constrain state-action choices in reinforcement learning in order to improve learning efficiency and generalization of learned policies within a single-agent herding task. We show that satisficing and generalizable policies emerge, which reduce computational cost, and/or memory resources.**

***Index Terms*—Markov decision process; reinforcement learning; hierarchical state representation; robotic herding.**

I. Introduction

Evidence from the natural world shows that mammals outperform machines in task domains relevant to both mammals and autonomous systems, e.g., hunting, tracking, foraging, and patrolling. These problems can be described mathematically as variants of a Markov decision process (MDP), which are proven to have complexity ranging from P-complete (MDP) to non-deterministic exponential or NEXP (decentralized-MDP) [1], [2]. How is it that mammals solve problems that we model as computationally hard or intractable? A large body of knowledge in neuroscience indicates that brain circuits give rise to computational operations that include hierarchical data structuring through clustering, embedded sequences, hash coding, reinforcement learning, and others [3], [4]. Drawing on themes from [3], [4], this paper explores learning derived from hierarchical and abstract representations of state and action spaces to reduce complexity in a single-agent learning task. These concise representations reduce the number of state-action pairs, and importantly, also reduce memory resources required in learning. In addition to improving learning efficiency, we seek to enhance the ability to generalize learned policies.

The concept of hierarchical representation of a state space for reducing a high dimensional or large state space is not new; methodologies for such representations are varied and applications have generally been limited to benchmark-type tasks. Construction of a state space based on statistical sampling can be performed in many ways, e.g., kernels, basis functions and Gibbs sampling [5], [6], [7]. An issue for this kind of state space construction is that of determining the resolution of the state space in advance, and thus these approaches can be impractical for complicated task domains. Another approach is based on pattern analysis of state features. Starting from a mass of states, a bottom-up scheme examines similarity of neighboring states, groups similar states into one state, and partitions the state space [8]. Mean and standard deviation of a reward was first used to determine the homogeneity of states in [8]. Rather than a bottom-up scheme, [9], [10] construct the state space from one initial super state, which provides a flexible framework for the learning agent to self-construct the state space with minimal algorithmic setup. A statistical test applied to a Q-value or to a change in Q-value for a state is introduced in [9], [10]. These algorithms have been tested in robotic and control tasks such as the Mountain Car [10], pole balancing [10], grid-based navigation [8] and soccer [9].

Methods for abstracting an action space presented in previous work include options, hierarchies of abstract machines (HAM), and MAXQ reinforcement learning [11], [12], [13], [14]. Options are learned in order to group sequences of primitive actions into a more abstract super action [12], while HAM and MAXQ focus on structural decomposition of a task domain [13], [14]. HAM and MAXQ require careful work flowchart design a priori.

This paper applies reinforcement learning of state-action space hierarchies to agent-based herding, a task that is learned, e.g., by sheepdogs, chimpanzees, and dolphins. We use a criterion derived from a clustering algorithm to measure similarity of data points in a state. Additionally, we define abstract actions as intuitive "commands" or options that assume that the agent has basic low-level motor primitives, such as setting a velocity or direction. Besides policy convergence, we assess the algorithm's efficiency both computationally and in memory usage. Moreover, we also consider the ability to generalize the learned policy to new instances of the task that were not explored during the learning phase. Single-agent herding is a precursor and prerequisite to a more complex multi-agent tracking/hunting task, in which state complexity is further increased due to the presence of other agents and heterogeneous roles.

The paper is organized as follows. Section II provides a formal definition of the flock model and herding task. Section III reviews reinforcement learning, analyzes the complexity of the single-agent herding task, and details the methodology for reducing complexity through hierarchical state representation and abstract actions. Simulation results and

Manuscript received April 13, 2012; revised May 27, 2012. This work was supported in part by the Office of Naval Research under a Multi-University Research Initiative (MURI) Grant No. N00014-08-1-0693.

The authors are with Thayer School of Engineering at Dartmouth College, Hanover, NH 03755, USA (e-mail: tao.mao@dartmouth.edu, laura.e.ray@dartmouth.edu).

analyses are given in Section IV.

II. Flock Model and Task Description

In the herding task, an agent must cause a flock of reactive agents to move to a goal location. Individual agents in the flock respond to the herding agent, other herded agents and a boundary, as in a dog herding sheep. Magnitudes and directions of these motion components are governed by a potential function whose parameters can be adjusted to emulate flocking behavior as in [15].

The flock model is based on potential field theory [16], which incorporates attraction and repulsion to describe the kinetics of moving agents. In this model, the motion of each herded agent k in the flock has a potential function with three components: c_1 to keep a reactive agent close to fellow reactive agents while avoiding collision; c_2 to repel from the herding agent; and c_3 to repel the reactive agent from the boundary of the environment:

$$v_k = c_1 - c_2 - c_3 \tag{1}$$

$$c_1 = \sum_{i=1, i\neq k}^{N} \left[\left(\frac{K_1}{\left| d_{A_i} + shift \right|^2} \right) \vec{d}_{A_i} - \left(\frac{K_2}{\left| d_{A_i} \right|^2} \right) \vec{d}_{A_i} \right] \tag{2}$$

$$c_2 = \left(\frac{K_3}{\left| d_H \right|^2} \right) \vec{d}_H , \quad c_3 = \left(\frac{K_4}{\left| d_B \right|^2} \right) \vec{d}_B . \tag{3, 4}$$

d_{Ai} is the distance from the reactive agent to its neighboring agent, *shift* is a compensation term to prevent agents from getting too close, d_H is the distance to the herding agent, and d_B is the distance to the boundary. K_1, K_2, K_3 and K_4 are kinetic parameters. N is the number of agents in the flock. The magnitude and direction of the reactive agent's response is associated with the distance and direction for each component as indicated in eq. (2-4). In addition, the reactive agent has a maximum velocity of v_m.

In the herding task, reactive agents can move in a circle whose center is the origin and whose diameter is D. The position of the flock is defined by its center of mass (COM). The goal is located at P_G. "Success" in reaching the goal is defined as the COM of the flock staying within a specified range d_G of the goal. The state is defined as $\{d_1, d_2, \theta\}$ where d_1 is the distance from herder to goal, d_2 is the distance from COM to goal, and θ is the angle between vectors from herder and COM to goal, as in Fig. 1. The herder updates its state with a frequency f. Based on this information, which is similar to that used in [15], the herder uses its learned policy to generate its velocity vector in order to drive the flock to the goal, or, the task terminates when the time exceeds T_f.

III. Reinforcement Learning and Hierarchical State Representation

A. Reinforcement Learning

A Markov decision process (MDP) is a mathematical framework for solving sequential decision-making problems in stochastic domains. A MDP of a single-agent system is formally defined as a 4-tuple $<S, A, P, R>$. S is a set of states $s \in S$; A is a finite action set available to the learning agent with $a_i \in A$; P is a transitional probability $P(s' \mid s, a_i)$ that represents the probability of transition from state s to state s'

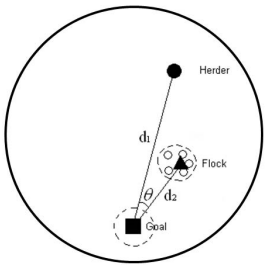


Fig. 1. Agent-based herding task.

with $a_i \in A$; P is a transitional probability $P(s' \mid s, a_i)$ that represents the probability of transition from state s to state s' due to action a_i taken by the agent; and R is the reward function of s and a. Computational complexity of a MDP and its variants is presented in [1] and is reviewed here for a herding task in which the size of the state-action space is a measure of complexity.

Q-learning, a reinforcement learning algorithm based on updating state-action Q values at each time step, is used here to solve the herding task, which is modeled as an MDP [17]:

$$Q_t(s,a) = (1-\alpha_t) Q_{t-1}(s,a) + \alpha_t [r + \gamma \max_{a'} \{ Q_{t-1}(s', a') \}] \tag{5}$$

s and s' are the state at time step t-1 and t, a is the action taken at state s, r is the reward received from the environment at time step t, $\alpha_t \in (0,1]$ is the learning rate, and $\gamma \in [0,1]$ is the discount factor. The best found policy is recorded in π^*:

$$\pi^* = \arg\max_a Q(s,a) \tag{6}$$

The convergence of single-agent Q-learning has been proven in [18] with conditions of action-value pairs visited indefinitely often and α_t satisfying

$$\lim_{T\to\infty} \sum_{t=0}^{T} \alpha_t = \infty \text{ and } \lim_{T\to\infty} \sum_{t=0}^{T} \alpha_t^2 < \infty \tag{7}$$

In addition to asynchronous learning, Q-learning can be interrupted and will return the best policy found thus far. This feature makes Q-learning a rational choice when there is a constraint on computation cost.

B. Complexity

Complexity of single-agent herding arises from discretization of continuous state and action spaces for describing relative location and for setting velocity and direction of the herder, as in numerous path planning and navigation problems [19]. Considering the COM of the flock to represent its aggregate location, $\{d_1, d_2, \theta\}$ can represent the continuous state space describing relative location of the herder, COM, and goal location. A discrete representation of this space has size $\prod_{i=1}^{3} n_i$, where n_i is the number of discrete bins for each element of the state space. The action space also contributes to learning complexity. A traditional approach to describing the action space is to discretize the continuous

space establishing heading angle of the herder as indicated by the blue and red lines in Fig. 2a; hence, for an action space of size a, the number of state-action pairs is of order $a\prod_{i=1}^{3} n_i$.

From options theory [12], the action space can be reduced by abstracting the concrete actions established by discretizing the continuous action space into a reduced set of actions. For the herding problem, we abstract the action space to six actions, which allow the herder to "Go toward" the flock, "Approach from the left or right", "Go around the flock from the left or right" and "Stop" as denoted by the pink arcs and stop sign in Fig. 2b. The command-like abstract action set stems from training of sheepdogs, where a shepherd may call out a finite set of commands.

C. Tree-Based Hierarchical State Representation

A coarse resolution of the continuous state space might not lead to a learned policy due to insufficient state information; conversely, a fine resolution results in a state space that is too large to explore. In this paper, we provide a hierarchical state representation to reduce state space complexity. The hierarchical representation is based on a decision-tree in which visited states are clustered according to similarity, characterized by patterns of action policies as in [9, [10]. States are clustered using a top-down scheme. Initially, there exists a single super-state, which is sequentially partitioned into subspaces as learning proceeds. The continuous state space need not be discretized for clustering.

The decision-tree-based (DTB) state representation is constructed by a sequence of judgments along a tree graph, where branches split the state space into subspaces, and subspaces are further split, when dissimilarity of action policies in a given node exceeds a threshold. This representation reduces the state space from $\prod_{i=1}^{3} n_i$ to $O(2^d)$, where d is the depth of the graph. In essence, what is learned is not a specific policy (sequence of actions) for a given set of initial conditions, but rather a state representation is learned along with a mapping associating subspaces of the state space with sequences of actions from each subspace. For instance, in Fig. 2c, which combines abstract actions with the decision-tree based representation (AA+DTB), in the green region, the dominant policy is "Go around the flock from left," and in the red region, the dominant policy is "Go around from the right."

The algorithm for growing the DTB hierarchical state representation is executed periodically dependent on accumulation of data for changes in Q values $\Delta Q(s,\underline{a}_i)$. When sufficient data exists, the algorithm evaluates the similarity M of two data sets D_1 and D_2 sampled from two data histories, which are divided by a proposed threshold Th in one dimension of the state space:

$$M = \frac{\sum_{d_{1i}\in D_1, d_{2i}\in D_2} \min\left(d_{1i}, d_{2i}\right)}{|D_1|} \tag{8}$$

$d_{1i}\in D_1$, $d_{2i}\in D_2$, and $|D_1|=|D_2|$. When any similarity M is below a threshold M_o, then splitting of a node is performed and the decision tree grows, as in the algorithm shown in Table I.

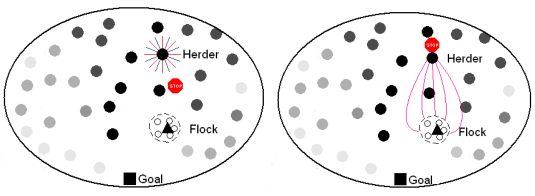
TABLE I: ALGORITHM OF GROWTH OF DTB HIERARCHICAL STATE REPRESENTATION

if #Visits of a state s exceeds N_s*
  Pull the dataset of $\Delta Q(s,\underline{a}_i)$ associated with state s;
  for all proposed thresholds $Th \in$ Range
    Sort data of $\Delta Q(s,\underline{a}_i)$ into two groups according to Th;
    Sampled data set $D1$ and $D2$ drawn from the above two groups;
    Measure the similarity M of $D1$ and $D2$;
    if $M < M_o$**
      Store M and Th as M_o and Th_o;
    end if
  end for
  if Th_o is not empty
    Do split state s into two states according to Th_o;
    // update information for old state s
    Left = new state1;
    Right = new state 2;
    Threshold = Th_o;
    // update for new states
    Initialize new state s' & s"; (Inherit Range from old state and Th_o)
  end if

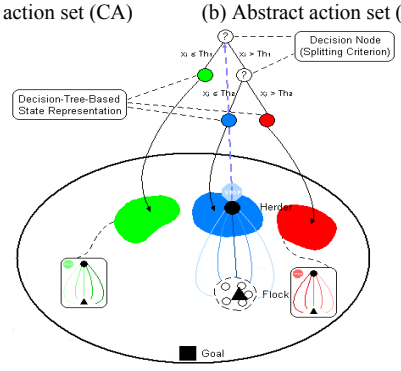
* Ns controls the frequency of inquiry of splitting criterion;
** M_o is defined so as to have a large difference between two data sets.

IV. Results and Analysis

We compare empirical performance of learning with hierarchical state representation for three cases: (1) CA, the traditional discretized representation of the continuous state and concrete action space, with $\prod_{i=1}^{3} n_i = 16\times16\times32 = 8192$ and a = 33 (32 discrete directions and "Stop" action); (2) AA, which assumes the same discretized state space and the abstract action space described above with a = 6; and (3) AA+DTB, which learns a hierarchical state representation and retains the abstract action space with a = 6. Flock model parameters and task environment parameters simulations are specified as follows: $K_1 = K_3 = 1$, $K_2 = 0.8$, $K_4 = 0.02$, $shift$ = 0.05 m, D = 7 m, $P_G = (0,-2.8)$, $d_G = 0.25$ m, f = 20Hz, T_f = 20s. We carry out 15,000 trials to evaluate policy convergence. Each trial starts with random initial locations of the herder and the flock COM, and terminates either when the herder succeeds or when the simulation time exceeds T_f.



(a) Concrete action set (CA) (b) Abstract action set (AA)



(c) Abstract action set and DTB state representation (AA+DTB)

Fig. 2. Abstract action and decision-tree-based state representation in agent-based herding problem.

During the learning phase, the initial position of COM of the flock and herder is constrained to keep the flock between the herder and the goal, but is otherwise random within this constraint. We impose this constraint during learning in order to evaluate the ability to generalize learned policies to initial conditions that violate this constraint.

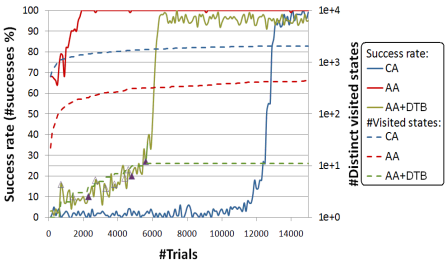


Fig. 3. Convergence of success rate and number of distinct visited states vs. trials for three learning methods with different strategies of state and action space representation.

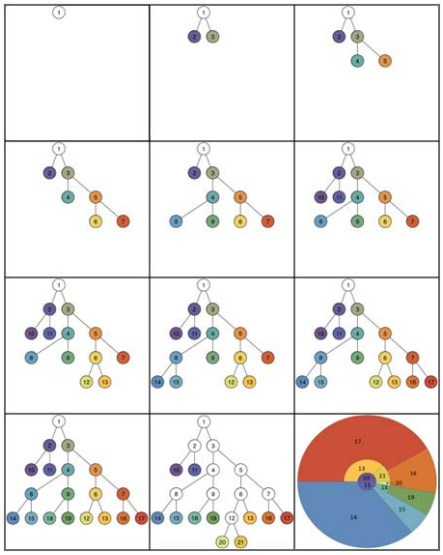


Fig. 4. Progression of learning hierarchical state representation during AA+DTB learning and subspace distribution of states (Colorwheel denotes partitions of each subspace in the final tree).

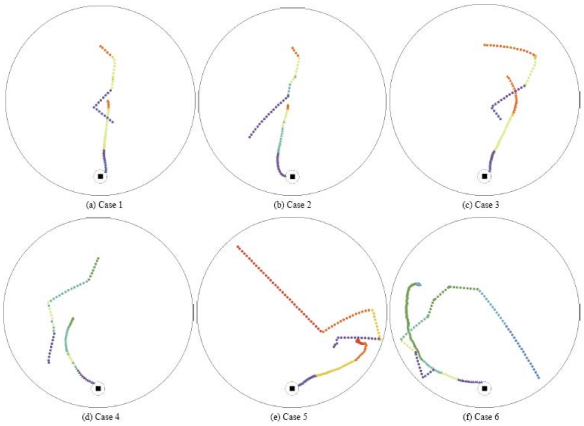


Fig. 5. Trajectories of herding agent and COM of flock in simulation experiments with different initial conditions (square: the goal; triangle: the flock COM; circle: the herder).

A. Learning Phase

Fig. 3 shows the success rate in learning and the number of distinct states visited during learning against number of trials. For the abstract action and decision-tree-based representation (AA+DTB), each triangle marker in Fig. 3 corresponds to a learning trial in which the learned hierarchical state space changes due to splitting of a node in the decision tree. Figure 3 also shows the number of unique states visited during learning for each of the three representations.

Examining convergence properties and the number of distinct states visited in Figure 3 shows that action space abstraction improves learning efficiency substantially because the learning system is less likely to visit useless states (circles shown in light gray in Fig. 2a and 2b). Adding the DTB representation degrades convergence in comparison to action space abstraction, because the hierarchical state representation must be constructed (learned) as new states are visited during learning, and action policies are re-learned when new states are generated in the decision tree. This is indicated by a temporary decrease in success rate after hollow triangle markers in Fig. 3. Nonetheless, the number of distinct states visited during DTB learning is reduced by an order of magnitude over AA and two orders of magnitude over CA. After 14,000 trials, the number of distinct states visited for CA is over 2,000 and ~500 states are visited for AA; however, for the DTB state representation, only 11 distinct states are in use after the hierarchical state representation is learned, and the depth of the DTB representation after 14,000 trials is $d = 5$. Note that 21 states appearing in Fig. 3 for the AA+DTB representation represent all states visited along the learning process, however, some of these states are split and are no longer used to identify a distinct state at the end of learning, which gives rise to the fact that only 11 distinct states are required to represent the state space once learning has converged.

Fig. 4 shows the progression of the developing state representation. Each snapshot shows the state representation when a new state is generated. The colorwheel in Fig. 4 is coded by state and represents the size of the subspace associated with each state. A hierarchical representation that is largely unrelated to d_1 emerges, and the colorwheel therefore shows subspaces in terms of d_2 (distance from the origin of the colorwheel) and θ (angle subtended by each state within the colorwheel) only. These results show that the learned hierarchical representation not only "maps" large portions of the state space to an associated action space, but can also provide fine-grained distinctions between states when needed and can reduce state space dimensionality.

This asymmetric division of the state space reduces complexity substantially; simple state and associated policy representations are achieved by integrating action abstraction with clustering and sequencing through the DTB state representation, just as mammals construct, retrieve, and process information through thalamocortical circuits [3], [4]. Furthermore, the large reduction in the size of the state-action space reduces memory requirements, which is of importance especially when agents learn a number of tasks concurrently.

B. Policy Evaluation

Fig. 5 shows trajectories for policies arising from AA+DTB learning applied to six different sets of initial conditions. In each case, the herding agent uses its learned policy to successfully drive the flock to the goal. The states that the herding agent encounters are denoted by color corresponding to the colorwheel in Fig. 4. Because the initial conditions vary, the herding agent will visit different absolute locations, but similar clustered states. In Fig. 5, a rich set of dynamic behaviors (indicated by the variety of trajectories)

emerge despite the sparse state and action space representations. For example, cases 1-2 have similar initial conditions and similar outcome of the herding agent driving the herd nearly directly to the goal. Nonetheless, the agent's policy execution results in nondeterministic behavior. Case 3 has an initial agent location similar to cases 1-2, with a different flock location; while the state transitions are similar to cases 1 and 2, the agent and herd trajectories are quite dissimilar. Case 4 has initial conditions that are nearly a mirror of Case 3. Thus, the herding agent carries out a series of action policies that almost mirror Case 3's trajectory.

Cases 5 and 6 are extreme cases compared to the first four cases. Case 5 sets up the initial condition of the flock and the herder agent such that they are far apart, while Case 6 purposely violates the constraint on the initial condition defined during the learning phase of keeping the flock between the herder and the goal. In both cases, the herder successfully applies its learned policy. Case 6 indicates of the ability to generalize the learned policy to initial conditions that are not encountered during the learning phase.

Fig. 6 shows the trajectories for the six cases of Fig. 5 plotted in the state space as depicted by the colorwheel in Fig. 4. Though the trajectories of Cases 1-4 show rich dynamic behaviors, the trajectories in the state space are similar. When initial conditions take on extreme values (i.e., Cases 5 and 6), the herder has the ability to drive the trajectory to a state in which it can then apply commonly-used policies to herd the flock to the goal.

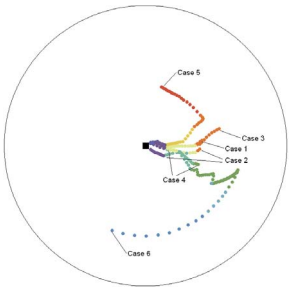


Fig. 6. Trajectory progresses in state space for above six cases

V. Conclusions

Action space abstraction and hierarchical state representation are applied to a herding task in order to reduce learning complexity. A hierarchical state representation is developed according to similarity of action values in different state regions. The final algorithm allows a herding agent to learn policies that generalize to both similar and extreme instances of the task. Reducing complexity in the single-agent herding task is a precursor to multi-agent herding and tracking in which complexity increases due to the presence of multiple learning agents.

References

[1] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819-840, 2002.

[2] Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein, "The complexity of multiagent systems: The price of silence," in *Proc. Second Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (*AAMAS*), Melbourne, Australia, pp. 1102-1103, 2003.

[3] R. Granger, "Engines of the brain: the computational instruction set of human cognition," *AI Magazine*, vol. 27, no. 2, pp. 15-32, 2006.

[4] A. Rodriguez, J. Whitson, and R. Granger, "Derivation and analysis of basic computational operations of thalamocortical circuits," *J. Cognitive* Neuroscience, vol. 16, no. 5, pp. 856–877, 2004.

[5] D. Ormoneit and S. Sen, "Kernel-based reinforcement learning," *IEEE Trans. Automatic Control*, vol. 47, no. 10, pp. 1624-1636, 2002.

[6] A. Yamaguchi, J. Takamatsu, and T. Ogasawara, "Constructing action set from basis functions for reinforcement learning of robot control," in *Proc. 2009 IEEE Int. Conf. on Robotics and Application* (*ICRA*), Kobe, Japan, pp. 4173-4180, 2009.

[7] H. Kimura, "Reinforcement learning in multi-dimensional state-action space using random rectangular coarse coding and Gibbs sampling," in *Proc.* SICE *Annual Conf.*, Takamatsu, Japan, pp. 2754-2761, 2007.

[8] M. Asadi and M. Huber, "State space reduction for hierarchical reinforcement learning," in *Proc.17th Int. FLAIRS Conf.*, Miami, FL, USA, 2004, pp. 509-514.

[9] W. Uther and M. Veloso, "Tree-based discretization for continuous state space reinforcement learning," in *Proc. Sixteenth National Conf. on* Artificial *Intelligent* (*AAAI-08*), Madison, WI, USA, July 1998.

[10] L. Pyeatt and A. Howe, "Decision tree function approximation in reinforcement learning," in *Proc. Third Int. Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*, pp. 70-77, 2001.

[11] A. Barto and S. Mahadevan, "Recent advance in hierarchical reinforcement learning," *Discrete Event Systems Journal*, vol. 13, pp. 41-77, 2003.

[12] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in Neural Information Processing Systems* 22, 2010.

[13] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," *Advances in Neural Information Processing Systems* 10, Cambridge, MA, USA, pp. 1043-1049, MIT Press.

[14] T. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," Technical report, Department of Computer Science, Oregon State University, Corvallis, Oregon, 1997.

[15] R. Vaughan, N. Sumpter, A. Frost *et al.*, "Experiments in automatic flock control," *Robotics and Autonomous Systems*, vol. 31, pp. 109-116, 2000.

[16] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 501–518, April 1992.

[17] C. Watkins and P. Dayan, "Technical Note: Q-Learning. Machine Learning," vol. 8, no. 3-4, pp. 279-292, 1992.

[18] J. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine Learning*. vol. 16, pp. 185-202, 1994.

[19] D. Busquets, R. L. de Màntaras, C. Sierra *et al.*, "Reinforcement Learning for Landmark-based Robot Navigation," in *Proc. Int. Joint Conf. Autonomous Agents and Multiagent Systems* (*AAMAS*), Bologna, Italy, pp. 841-843, 2002

Tao Mao joined Thayer School of Engineering at Dartmouth College in 2008 and is currently pursuing a Ph.D. degree. He received his Bachelor degree in Electrical Engineering with first-class honors from Zhejiang University in China. His research interests include multi-agent intelligent systems, machine learning and reinforcement learning.

Laura E. Ray is a professor of engineering sciences at the Thayer School of Engineering, Dartmouth College. She received the B.S. degree with highest honors and the Ph.D. in Mechanical and Aerospace Engineering from Princeton University and the M.S. degree in Mechanical Engineering from Stanford University. Her current research interests include control of multi-agent systems, robot mobility and vehicle-terrain interaction, and field robotics.