Dynamical Categories and Control Policy Selection *

Jefferson A. Coelho Jr., Elizeth G. Araujo, Manfred Huber, Roderic A. Grupen

Laboratory for Perceptual Robotics† - Department of Computer Science
University of Massachusetts, Amherst, MA, 01003 USA

ABSTRACT

Every autonomous agent operating in realistic settings must deal with incomplete state information. Perceptual limitations often compromise system observability, and may prevent the acquisition of optimal control policies for a given task. This paper addresses the observability problem within the control composition framework, where the agent selects which policy to adopt next from a set of pre-defined control policies. The idea is to treat the agent in its environment as a dynamical system, and augment the perceived state space (situation space) with contextual cues extracted empirically as the agent exercises each of the existing control policies. Contextual cues are provided by the correlation between dynamic features of the agent-environment interaction and agent performance. Initial experiments involving an agent with impoverished sensing capabilities in a simulated, dynamic environment demonstrate that relevant contextual information can be extracted and used to enhance the agent's performance.

KEYWORDS: *Learning, embodied agents, attention, sensorimotor development, reinforcement learning.*

1. INTRODUCTION

For an autonomous agent acting in the real world it is important to learn behavioral strategies without depending on outside supervision. To achieve such flexibility, a number of systems employing reinforcement learning techniques have been implemented [3, 9, 11]. In these approaches the agent acquires a policy by interacting with the environment, while observing a simple reinforcement signal. Typically, the complexity of exploration-based learning techniques renders them impracticable for many on-line learning tasks. To address this complexity issue some interest has focused recently on hybrid or modular learning approaches. Here the objective is to learn the best composition for a set of *controllers* [1, 12, 15] rather than the best activation sequence for a set of low-level control actions. Using controllers or even complete policies as actions in the learning scheme leads to smaller state and action spaces, allowing for the solution of more complex tasks [5, 8]. Moreover, controllers can also provide the system with a basic level of performance by providing solutions to important subproblems.

A practical problem arises in real world tasks due to unobservable or incomplete state information, resulting in a non-Markovian system for which learning techniques such as Q-learning [17] will generally not derive optimal policies. To address the problem of hidden state information several approaches have been proposed which either explicitly model the system as a partially observable Markov decision problem (POMDP) [10] or which attempt to augment the state information on-line by explicitly incorporating a sufficient amount of past information [14]. Both approaches attempt to resolve non-Markovian character throughout the learning process and to learn an optimal policy for the developing Markovian system. This is a computationally hard task and may easily become intractable even for moderately complex problems.

This paper addresses learning in non-Markovian environments within the control composition framework. Every policy in the composition set elicits a characteristic dynamic response that varies as the control context varies; in fact, we propose to use the dynamic response itself as an identifier for the control context. We show that control context can be used to augment the observed state variables (or situation variables), and that categories of dynamic responses can, in principle, be used to resolve hidden states. The approach proposed is called Contextual Policy Selection (or CPS for short). The complete approach is described in the next section. Section 3 describes the task used to test our approach, including the original state representation, the baseline control policies and their derivation. Task-specific details for the implementation of our approach are discussed in Section 4. Section 5 presents experiments performed using this approach on the foraging task, and Section 6 concludes this paper.

2. CONTEXTUAL POLICY SELECTION

This work presumes the existence of a set of control policies that promote agent behavior in response to sensory observations. For simplicity, it is assumed that only one control policy is active at every moment, although in general more than one policy may be eligible for execution; in addition, the control policies are independent of each other (i.e. two eligible policies

*This work was supported in part by NSF grants IRI-9503687 and IRI-9704530.

†http://piglet.cs.umass.edu:4321

should promote distinct behavior at least in same region of the state space).

The framework described is attractive for a number of reasons. First, policy synthesis or optimization for non-Markovian systems is a hard computational problem, and the use of policies as the synthesis elements minimize the complexity of the learning task. Control policies or controllers allow the system to abstract from dealing with raw percepts, and focus on relevant aspects of the task. Also, pre-defined control policies can be used to address important aspects of the problem domain early in the learning process. Second, alternative control policies are often available; in fact, standard reinforcement learning techniques can be used to derive alternative control policies for partially observable environments. The derived policies will be suboptimal in general, but distinct (suboptimal) policies may have complementary strengths across the range of control contexts. Third, the lack of state information can be mitigated by employing a number of distinct control policies in appropriate contexts. Consider the case in which the agent's task is to forage for food in an environment subject to changes in seasons. If the agent does not have a sensor for "winter" or "summer", the best policy it will develop is an average policy that guarantees its survival across both seasons. As it adopts the average policy, the agent may disregard very good foraging policies that are season specific.

The goal of this work is to derive a policy selection strategy that optimizes the agent's performance in the environment. The basic idea is to bring context to bear on the policy selection process; here context is defined by the very dynamic response of the agent, and therefore it is a function of the current control policy. The method consists of three steps, namely (1) online acquisition of a dynamic models database for each control policy, (2) construction of a Bayesian model of system performance, and (3) learning control composition on the augmented situation space. This framework can be used to study learning in sensorimotor systems, the role of native control elements in overall performance, and mechanisms for resolving relevant forms of hidden state in policy formation. The next sections discuss the steps of the CPS approach.

2.1. *Dynamic Models Database*

To construct models of characteristic dynamic responses, each "action" (either a closed-loop controller or a policy at the same level) in the agent's repertoire is executed in a number of trials. Each trial produces a trajectory in the observable state space, and each trajectory is stored in memory. Many distinct paths will be generated, as well as a number of similar paths, due to the structure introduced by the underlying control policy π_i. Each path in state space can be labeled with an attribute vector, evaluated at the end of each trial. The elements of the attribute vector are performance measures relevant to some task. The resulting database records how well the agent expects to fare in the current context or state, given its control repertoire and intrinsic sensory and motor limitations.

We will constrain the control policies to the subset of policies with absorbing states; that is, only those policies with finite duration will be considered. Control policies can be designed to be convergent, or to terminate as the task objective is achieved.

Many dynamical system techniques can be used to determine the number of trials required for the construction of a representative dynamical models database. One can evaluate how much novelty each trial is bringing to the database already in memory; for instance, novelty can be measured as how many of the agent's actions can be predicted with the current dynamic models.

Memory requirements will depend greatly on the complexity of the agent-environment dynamics, and not much can be said in general. However, as other researchers have noted earlier [2, 13] the cost of memory is typically much smaller than the cost of running a robot; it makes sense to cache data corresponding to a number of trials, and use it to advantage later.

2.2. *Augmentation of Observed State Space*

The classic approach for dealing with non-Markovian systems is to augment the notion of state to be the sequence of the last n observations. The underlying assumption is that the non-Markovian system can be modeled by a n-order Markovian system. The determination of n is generally done empirically; in some cases no matter how large n is, this reduction from non-Markovian to Markovian system cannot be accomplished.

Instead of attempting to find a good value for n, we will take a more pragmatic approach, and augment the space of observed states using a scalar measure of the agent performance. This performance measure can be evaluated by matching current observations to the agent's experience captured in the dynamic models database. The added complexity of the resulting augmented state corresponds to a extra scalar dimension, compared to a $n-1$ dimensional vector of previous state observations.

2.3. *Bayesian Models of System Performance*

Once the dynamic models are available, the agent can compute the expected performance $r(\pi)$ under policy π, conditioned to the probability that the sequence of k state observations $x = \{x_1, \ldots, x_k\}$ comes from any model in the database:

$$\begin{aligned} r(\pi) &= E[R|X=x,\pi] \\ &= \sum_{i=1}^{n} r_i p(R=r_i|X=x,\pi), \end{aligned} \tag{1}$$

where π is the current control policy, and n is the number of models in the model database for policy π. As more observations are available (i.e., as k grows), the estimation of r becomes closer to its true value (assuming the universe of models is representative of the agent-environment dynamics), and fewer terms are significant in the summation above. The explicit dependence on π has important consequences: one cannot carry a sequence of observations across changes in the control policy. As soon as the agent decides to change its control policy, the vector of observations x must be reinitialized. So at every moment the agent has the choice to keep executing the current policy and refine the estimation of its performance, or change policies and re-start the estimation process.

Notice that the problem above belongs to the class of Bayesian classification problems [7], in which the objective is to use the evidence available (state observations) to compute the degree of belief such evidence comes from one of several classes (models, in our case). The framework can in principle deal with noisy observations, the presence of outliers (i.e., state observations that do not come from any model in the database), and discrete and continuous state spaces.

2.4. *Policy Selection with Augmented States*

There are many ways of incorporating the estimates of future performance into the agent's state description, as discussed in Section 4. Once a state augmentation scheme is selected, standard reinforcement learning can be use to derive an optimal switching policy for a given reward function. Notice that even for continuous observation variables, the augmented state space will be discrete, with the use of model labels. The action space consists of a set of control policies available to the agent, each of which offers a reasonable level of performance. The reward structure is task dependent.

3. THE FORAGING TASK

The experimental task used throughout this paper is the foraging task described in [1] in which a dynamic learning agent with limited sensing capabilities has to acquire policies that allow it to capture prey in a dynamic world. This in turn requires strategies for finding, approaching, and attacking a prey under the time pressure introduced by a limited store of energy. To succeed at this task and to optimize performance, the agent has to solve a complex stochastic sequential decision task. The agent acquires a policy for the combination of primitive control elements taken from a set of controllers [4]. To evaluate the agent's performance at this foraging task, two types of environments are used which differ in their degree of complexity. While in both cases the environment is initialized randomly at the beginning of each trial, all prey in the first environment are stationary and do not react to the dynamics of the agent. In the second environment, different types of prey are present and each type exhibits a characteristic behavior that interacts with the learning agent's dynamics. In addition, different types of prey provide different amounts of reward.

Figure 1 shows the second environment where the learning agent is represented by a frog and the different species of prey indicate their distinctive behaviors. The relative position of the bugs, their identity, and their velocity define state of the environment. To the agent, however, most of this information is not accessible due to its sensory limitations. In this foraging task the learning agent can only perceive the location and identity of the closest prey within a neighborhood; this neighborhood is coarsely partitioned in 3 distance levels and 2 angular zones (*center* and *off-center*)[1]. Also available are the presence of a second prey within range, and the agent's own energy level discretized into 3 levels. Due to this limited sensory capability, the agent's representation of the world is generally quite ambiguous, and the environment behaves as a non-Markovian system.

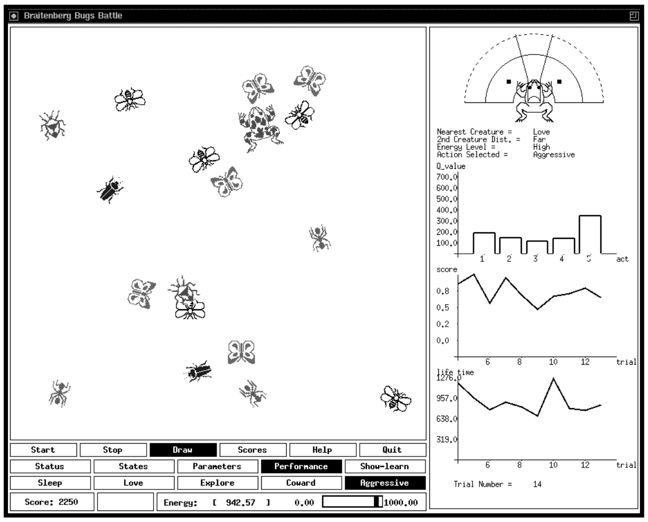

Figure 1: Simulated Foraging Environment

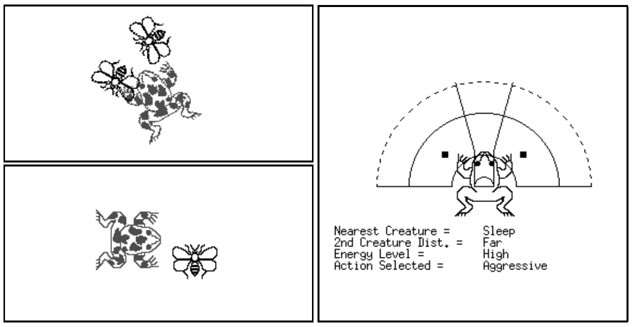

Figure 2: Hidden states

Figure 2 shows two states that are indistinguishable for the agent. The left side of this figure shows the two partially observable world states while the right side represents the discretized perceptions available to the agent, i.e its current situation. In this example, the second prey in the top state is out of range and thus cannot be perceived by the agent, leading to hidden state. The presence of such hidden state in a reinforce-

[1]The agent can not differentiate between left and right.

ment learning system can severely hamper the performance of the agent. Furthermore, methods like Q-learning are not guaranteed to converge under these conditions [16]. As expected, the use of standard reinforcement learning procedures on this foraging task resulted in a set of different policies with varying performance. Techniques such as shaping [6], Q-learning with eligibility trace [1] can significantly improve the quality of solutions compared to standard Q-learning.

4. CPS AND FOOD FORAGING

The food foraging task just described is attractive for testing the ideas introduced in Section 2: it has many hidden states, a family of control policies with (possibly) distinct characteristics, and a well-defined performance measure.

However, in order to apply the CPS approach to this task, we segmented the trials into attacks, that start and end when the frog adopts the "aggressive" behavior as it attempts to prey on some other bug near it. Each attack results in either success (the bug is eaten) or failure (poison bug is eaten or no bug is eaten). The dynamic models database for a policy π_i is then built over a number of trials, while the agent follows π_i.

We experimented with two implementations of the performance estimation procedure; for the first implementation, the one step transition from the previous situation to the current situation is used as the estimate of the system dynamics. In this case, the dynamic models database can be simplified to a database of transitions and the respective probability that a bug is captured. We decided to build this database using 10 trials for the static environment (described in Section 3) and 30 trials for the dynamic environment. Figure 3 depicts the percentage of new transitions over the total number of transitions, as a function of the number of trials. The top curve corresponds to the dynamic environment, and the bottom curve corresponds to the static environment. New transitions are found mostly in the first trials, and the chance of finding new transitions decay with the number of trials. Figure 3 also shows that our dynamic models are not complete, if only the first 30 and 10 trials are used in their construction.

The second implementation draws context information from the set of possible paths in which the system may find itself, given the sequence of sensed states. The dynamic models database was constructed based on data collected over 40 trials, for the static environment, and over 300 trials, for the dynamic one. The determination of the number of trials was done empirically, and it corresponds to the number of trials required before the percentage of new transitions taken during the last trial falls below 0.5%.

While executing a given policy, the performance estimation procedure relies on information extracted from the set of paths consistent with the observations made so far by the agent. As more transitions are performed, this set of paths typically will shed some of its elements and supply a more refined estimate.

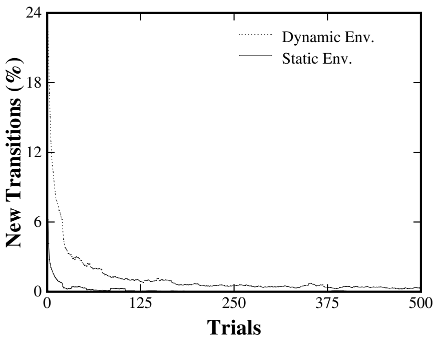

Figure 3: Percentage of new transitions found as a function of the number of trials. The top and bottom curves correspond to the dynamic and static environments, respectively.

We are currently running experiments with this implementation, and the results are preliminary at this stage.

The augmented state representation is constructed by adding extra bits to the original state representation. The total number of extra bits will depend on the number of policies one wishes to compose, and on the discretization adopted for the percentage of good solutions ("successful attacks", for the foraging task) returned by the performance evaluation procedure. The results reported in Section 5 considered at most the composition of two policies, and only one level of discretization; therefore, only two extra bits were used to define the augmented state representation.

In the case of the transition-based evaluation procedure, the decision between "good" and "bad" is made based on the quality of the average incoming transition to the current state. For instance, in Figure 4, the average quality of incoming transitions to state s_1 is 20%. Therefore, transitions t_1 and t_2 would be flagged as good, since they are above average.

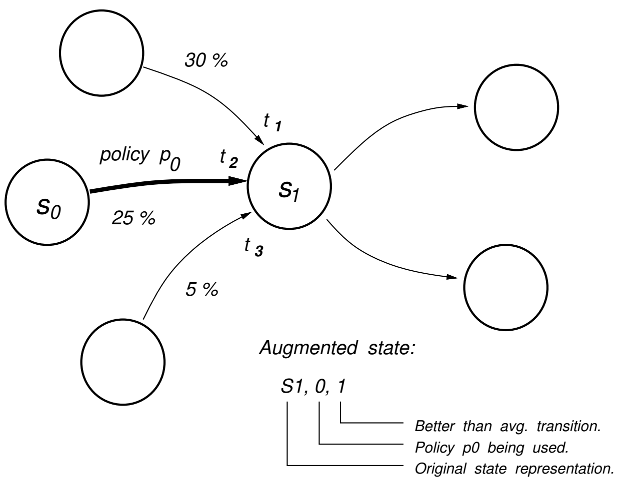

Figure 4: Augmented state information.

For the path-based evaluation procedure, the decision between “good” and “bad” prospects is based on the average percentage of successful attacks, as measured during the database construction stage. Given a set of paths, and respective success expectation, the system will regard the current context as good if the success expectation is above average.

5. EXPERIMENTS

For both the static and dynamic environments, the agent learns about the utility of switching control policies using the Q-learning eligibility trace algorithm[2] with the following parameters:[3]

Learning Parameters:

- *learning rate ($\beta = 0.025$);*
- *discount factor ($\gamma = 0.95$);*
- *eligibility factor ($\alpha = 0.95$);*
- *initial temperature* $= 1000.0$ *(static environment),* 2000.0 *(dynamic environment);*
- *final temperature* $= 1.0$;
- *temperature decay factor* $= 0.98$;
- *Q-values were initialized with zero;*
- *the eligibility trace was initialized with zero;*
- *the state-action table is updated at each step of the simulation.*

The first experiment involved composing two policies for the static environment, one derived using Q-learning (P1) and the other derived with Q-learning eligibility trace (P2). Figure 5 depicts the performance histograms for both policies, computed over 300 test-trials (without exploration). P2 is a better policy, in terms of its mean performance (81% versus 57%) and its (smaller) variance.

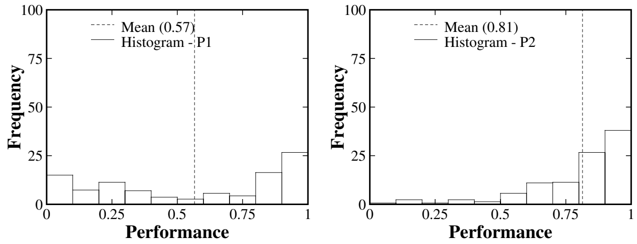

Figure 5: Performance histograms of policies generated with Q-learning (P1, left) and Q-learning eligibility trace (P2, right)

For this first experiment, the transition-based context evaluation was used (case M1). The left column of Figure 6 depicts the learning curve for this case, preceded by a fragment of the learning curves for the policies being composed, for comparison purposes. Each point of the learning curve represents the average performance of 30 test-trials, and the error bars indicate the 95% confidence interval. A total of 950 learning trials were executed, and the histograms on the right correspond to the performance distribution over a total of 300 test-trials.

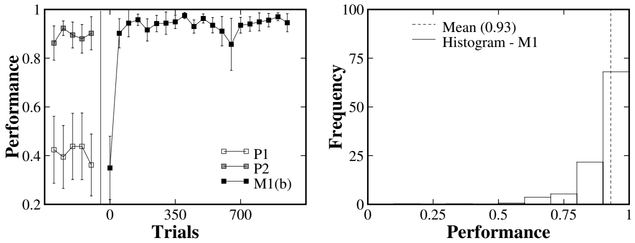

Figure 6: Learning curves and histograms for the policy composition using transition-based evaluation (M1).

When compared to the best policy being composed (P2), the policy M1 is about 15% better in terms of average performance, and also exhibit a significantly smaller variance.

Figure 7 presents the learning curve for the composition of the same policies when the path-based evaluation procedure is used to predict context information. When compared to the learning curve in Figure 6, the learning curve for this case (M2) appears to be slower, more stable, with less peaks/valleys, and more consistent, as indicated by the smaller error bars. The average performance, however, is approximately the same as for the transition-based context evaluation procedure.

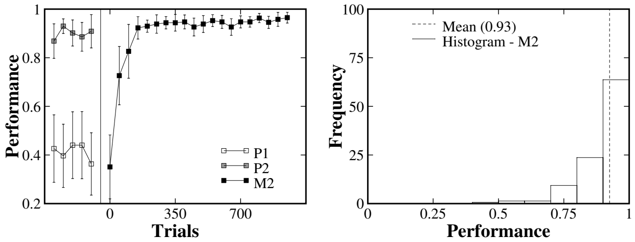

Figure 7: Results from the policy obtained using path-based context evaluation procedure.

The following experiment explored the benefits of composing policies for the dynamic environment. In this case, two policies constructed using the Q-learning eligibility trace algorithm were selected for composition; Figure 8 depicts the performance histograms for both policies (P3 and P4). As one can see from the average performance figures, the dynamic environment is a much harder learning problem.

Figure 9 shows the corresponding learning curve and performance histogram for the transition-based evaluation procedure (M4). As before, the average performance increased approximately 23%, from 57% to 70%, with a simultaneous decrease in variance.

[2] It is described in detail in [1].

[3] Little experimentation was done on the values of the learning parameters, due to time limitations.

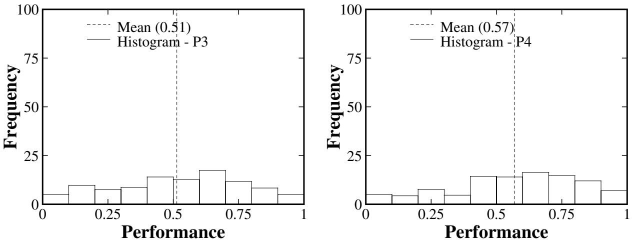

Figure 8: Performance histograms of two policies obtained with Q-learning eligibility trace (P3 and P4) in the dynamic environment.

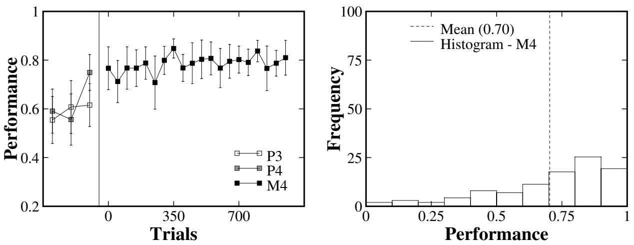

Figure 9: Results from the policy obtained with transition-based context evaluation procedure (M4).

The combined policy achieves the same performance level that was reached using hand-crafted shaping procedures in the original reinforcement learning experiments. As opposed to shaping techniques, however, the composition approach presented here can achieve this performance autonomously. Furthermore, in an identical composition experiment using two policies learned in the dynamic environment using shaping, a composition policy was learned with an average performance of 80% which is comparable with the best hand-crafted policies and outperforms all previously learned policies. These results suggest that the CPS approach presented in this paper might offer a relatively simple way to achieve higher performance solutions automatically from lower quality ones.

6. CONCLUSION AND FUTURE WORK

The contextual policy selection approach maps system dynamics into performance expectations under an existing control policy. The resulting evaluation augments the available state information, and defines a search space in which existing control policies may be combined to improve overall performance. Our results showed that policy composition indeed performed better ($\approx 15\% - 23\%$), even though there was little room for improvement, given the quality of the policies being combined.

As for future work, we plan to evaluate how much contextual information the system is building as it learns to combine policies. In principle, the learning algorithm may improve its performance by not switching policies and building up context information until a more informed decision may be taken.

References

[1] E. Araujo and R. Grupen. Learning control composition in a complex environment. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 333–342, Cape Cod, MA, September 1996. MIT Press/Bradford Books.

[2] C. Atkenson. Using locally weighted regression for robot learning. In *Proc. 1991 IEEE Int. Conf. Robotics Automat.*, volume 2, pages 958–963, Sacramento, CA, May 1991.

[3] A. G. Barto, R. S. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cyber.*, 13(5):834–846, 1983.

[4] V. Braitenberg. *Vehicles - Experiments in Synthetic Psychology*. M.I.T. Press, Cambridge, MA, 1984.

[5] R. Grupen, M. Huber, J. A. Coelho Jr., and K. Souccar. Distributed control representation for manipulation tasks. *IEEE Expert, Special Track on Intelligent Robotic Systems*, 10(2):9–14, April 1995.

[6] V. Gullapalli and A. G. Barto. Shaping as a method for accelerating reinforcement learning. In *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, pages 554–559, Glasgow, Scotland, UK, August 1992. IEEE.

[7] R. Hanson, J. Stutz, and P. Cheeseman. Bayesian classification theory. Technical Report Technical Report FIA-90-12-7-01, Artificial Intelligence Research Branch, NASA Ames Research Center, 1990.

[8] M. Huber and R. A. Grupen. A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems*, 22(3-4):303–315, December 1997.

[9] L.-J. Lin. *Reinforcement Learning for Robots using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, January 1993.

[10] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Twelfth International Conference on Machine Learning*, 1995.

[11] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. Technical report, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598, December 1990.

[12] M. J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[13] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

[14] R. A. McCallum. Hidden state and reinforcement learning with instance-based state identification. *IEEE Trans. Syst. Man Cyber.*, 26B(3):464–473, 1996.

[15] M. Meila and M. I. Jordan. Markov mixtures of experts. In R. Murray-Smith and T. Johansen, editors, *Multiple Model Approaches to Modelling and Control*. Taylor and Francis, 1997.

[16] C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.

[17] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.