To appear in Proc. of the 4^{th} Int. Conf. on Simulation of Adaptive Behavior (SAB'96) - Sept., 1996 - Cape Cod, MA, USA

Learning Control Composition in a Complex Environment

E. G. Araujo and R. A. Grupen
Laboratory for Perceptual Robotics *
Department of Computer Science
University of Massachusetts
Amherst, MA, 01003
araujo@cs.umass.edu, grupen@cs.umass.edu

Abstract

In this paper, reinforcement learning algorithms are applied to a foraging task, expressed as a control composition problem. The domain used is a simulated world in which a variety of creatures (agents) live and interact, reacting to stimuli and to each other. In such dynamic, uncertain environments, fast adaptation is important, and there is a need for new architectures that facilitate online learning. Recently, control composition has shown its potential in reducing the complexity of learning in many domains, permitting more complex tasks to be addressed. Results presented here restate the above claim and also show that the use of a modified version of the Q-learning algorithm performs better in the non-Markovian sequential decision task proposed. In addition, it is shown that shaping can be successfully used to improve the performance of these learning algorithms in complex environments.

1 Introduction

Fast adaptation is crucial to any agent that interacts with a dynamic, uncertain environment. In several domains such as artificial life and robotics new architectures that ease the learning process in order to produce on-line responses are required (Maes and Brooks, 1990; Sutton, 1990; Lin, 1993). Recently, the interest in learning of control compositions has increased (Mahadevan and Connell, 1990; Singh et al., 1994). Controllers ease the learning task by compressing the state space, consequently allowing the solution of more complex tasks (Grupen et al., 1995; Huber et al., 1996). Moreover, controllers can be designed to avoid unsafe situations during the learning process and to incorporate explicitly the dynamic constraints of the agent, freeing the learning process from dangerous and complex learning situations. This paradigm defines a new set of problems, as for example, how to synthesize and compose controllers.

In this paper, two reinforcement learning algorithms applied to a control composition task are compared. Reinforcement learning algorithms learn an optimal control policy - the best action in each state - for a sequential decision problem by receiving delayed reinforcement (Barto et al., 1990). The reinforcement learning framework was selected mainly because of the small amount of supervision necessary for learning, and the fact that the reinforcement can be delayed. These are very important characteristics, allowing for the solution of problems without complete knowledge of the processes involved. For example, animal conditioning can be framed as a reinforcement learning tasks (Sutton and Barto, 1990).

The task selected to demonstrate the advantages of control composition over learning from scratch is foraging in a simulated environment, where prey are guided by a single controller, and the learning agent (predator) composes all the controllers available. The control basis selected and the idea of using a control composition approach on this task were inspired by the work of Valentino Braitenberg (Braitenberg, 1984). For a review of the challenges presented by foraging tasks on the reinforcement learning framework, see (Watkins, 1989).

The control composition model is introduced in Section 2. Section 3 describes the task, the specifics of the control composition model for the task, and the two simulated environments where the task is performed. The learning methods and the results obtained are presented in Section 5. Section 6 introduces a strategy based on shaping to facilitate learning. Finally, Section 7 presents conclusions and future work.

2 Control Composition

There are several advantages to solving a task by activating controllers instead of learning sensorimotor commands from scratch. By selecting a suitable set of controllers, the state space of the learning task is dra-

*http://piglet.cs.umass.edu:4321

matically compressed, and the resulting higher performance adaptive algorithm can be applied to more complex tasks. In addition, controllers can more easily deal with the dynamics of the agents. To learn dynamics from scratch usually requires a large state space resulting in a low quality solution considering the amount of time required for learning. In addition, controllers can focus the learning on the relevant situations and avoid the danger of exploring unsafe states. Another motivation for control composition is that behaviors considered to be complex by an observer can be generated by a composition of very simple control laws (Braitenberg, 1984).

One may argue that learning from scratch will produce better control policies. The argument made here is that this could be the case, however it will in general take more time, and it will make the learning of more complex tasks impractical. The main idea is to create a control basis able to span a wide variety of behaviors by control composition, and to obtain more complex behaviors by successive composition of previously learned controllers.

The diagram in Figure 1 is a general control composition model. Its components are loosely related to biological structures: the initial set of controllers can be associated with reflexes, the mechanism for control adaptation can be the product of evolution, and the further composition of controllers can be associated to lifetime learning. Notice that the control level could be augmented over time with the controllers previously learned by control composition, allowing a continuous increase in complexity of the behaviors that the agent exhibits.

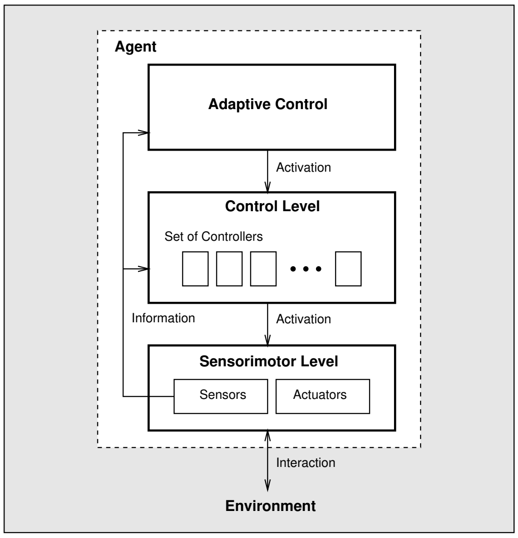


Figure 1: Control composition model of an agent.

3 Composition Task Description

The learning domain consists of a simulated world, described in Section 4, in which a variety of creatures live and interact, reacting to each other. The behaviors that the creatures display and the corresponding controllers are described in Section 3.2.

The task is foraging: by activating controllers, a learning agent (*predator*) has to search for other creatures (*prey*), approach them, and engage in a precise attack. If successful in this sequential decision task, the learning agent receives a positive reinforcement and an extra amount of energy (*food*) for each successful attack. Energy is consumed over time, and a trial ends when there is no energy left.

This problem can be described as a stochastic sequential decision task. It is non-Markovian, because of the presence of partially observable states (hidden states) in its discrete state representation. Partially observable states are caused by the limited amount of sensory information available to the agent, and also by the discretization of the state space.

The control composition model used to address the foraging task is shown in Figure 2, and its three levels are described in the following sections.

3.1 Sensorimotor Level

The sensorimotor level receives as input the torque commands to the actuators, and relays the internal and external sensor readings to the control and adaptive levels.

All creatures in the environment are inertial bodies (vehicles) with two light sensors coupled to two drive motors through some simple control logic as in Figure 3. Each creature is perceived as a light source by the others, and their light intensity sensors have a limited range of perception. In addition, the learning agent is capable of identifying the closest creature's type, relative distance at three relative depths, front, and side. Furthermore, the learning agent can perceive its own energy and internal reinforcement also in discrete levels.

3.2 Control Level

The control level receives as input a selection signal, indicating, in a mutually exclusive way, which controller to activate, and the light intensity detected by each sensor. It outputs the torque commands to each wheel, based on the control law of the activated controller.

The five different controllers used were inspired by the behaviors in (Braitenberg, 1984), and implemented as control laws as shown in Figure 3. τ_1 and τ_2 are the torques applied to each respective motor, i_1 and i_2 are the light intensity captured by each light sensor, and τ_{max}, τ_{min}, τ_{turn}, and *gain* are constants representing: the maximum torque, a small torque disturbance to avoid straight paths (allowing a better exploration of the environment), and the gain applied to transform light intensity into torque commands, respectively.

Each controller produces a different behavior: *Sleep*

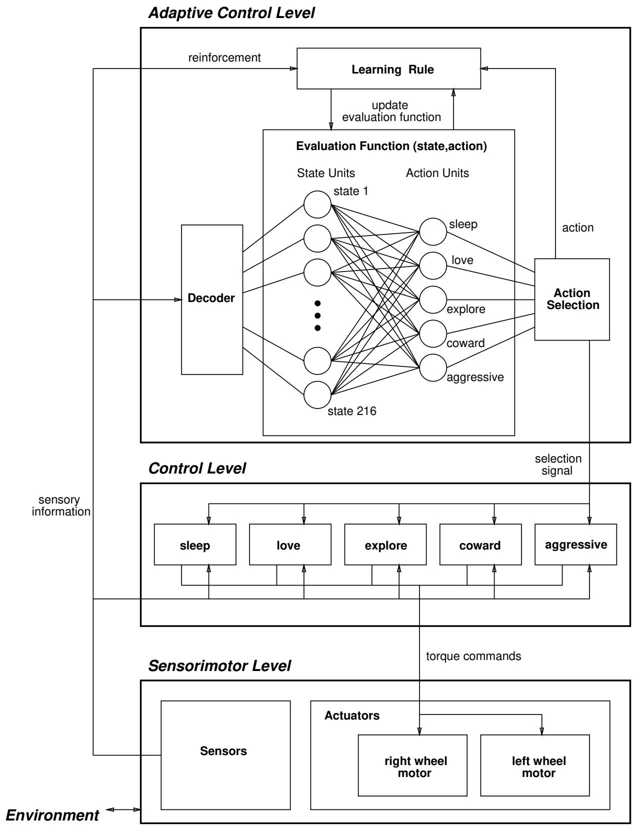


Figure 2: Task specific control composition model

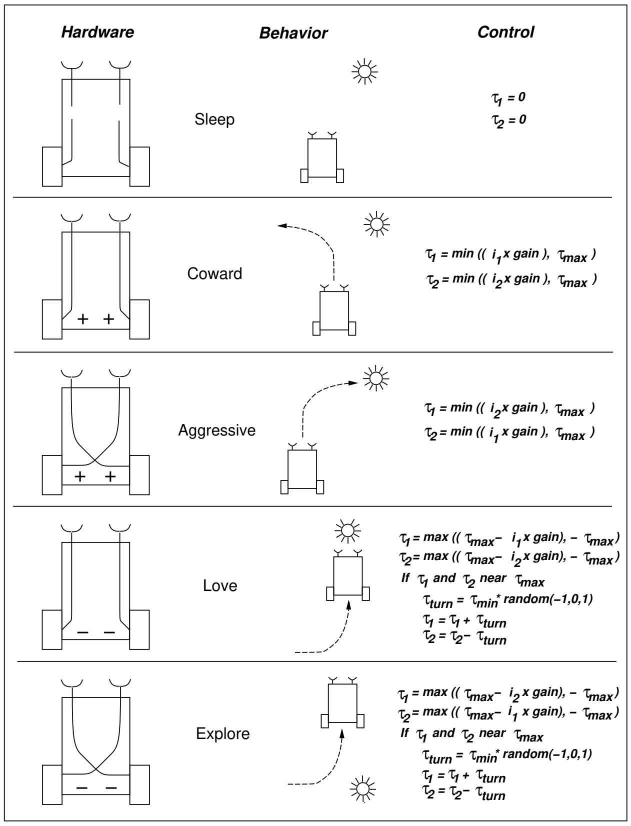


Figure 3: Description of the controllers

applies no torque to the wheels; *Coward* applies a greater positive torque ipsilaterally, turning the creature away from the light and then coming to a stop; *Aggressive*, transforms visual stimulation into contralateral motor activation, runs into the light source and, when no light is detected, comes to a stop; *Love* employs inhibitory, ipsilateral motor actuations, causing it to fix on stimuli; *Explore* employs inhibitory, contralateral motor actuations to avoid contact with any light. The learning agent is the only creature that has access to all the controllers, all the other creatures are tied to a unique controller, expressing a unique behavior.

3.3 Adaptive Control level

The adaptive control level selects which controller to activate at each moment, and computes the reinforcement signal used by the learning procedure. In addition, it receives sensory information and uses a *decoder* to transform the perceived state of the system (state of the environment plus internal state of the agent) into the corresponding learning state. Note that perceived states (or situations) usually do not describe unambiguously the true states of the system. Some states of the system are partially observable by the sensors employed. The *action selection* module selects a controller to be activated at each time and sends this information to the *learning procedure*. The evaluation function is updated over time by the *learning procedure*, determining the best controller activation command (action) for each state.

4 Simulated Environment

The environment contains a fixed number of creatures and two indicators of performance: the score which is used as the reinforcement signal, and the elapsed time since the beginning of a trial. There are up to six different types of agents: five of them assume a unique behavior, and the sixth one, the learning agent (depicted as a *frog* in Figure 4), is able to select which of the five behaviors to assume at each time. Each behavior is generated by a control law that determines the dynamics of the creature. The only way to influence the learning agent's dynamics is by sequencing controllers.

Figure 4 shows the complex environment that possesses a random set of all six kinds of creatures (*bee* - sleep behavior, *butterfly* - lover, *ant* - explorer, *roach* - coward, *beetle* - aggressive, and *frog* - learning agent). A simplified environment is shown in Figure 5 where the only creatures are *bees* and the learning agent.

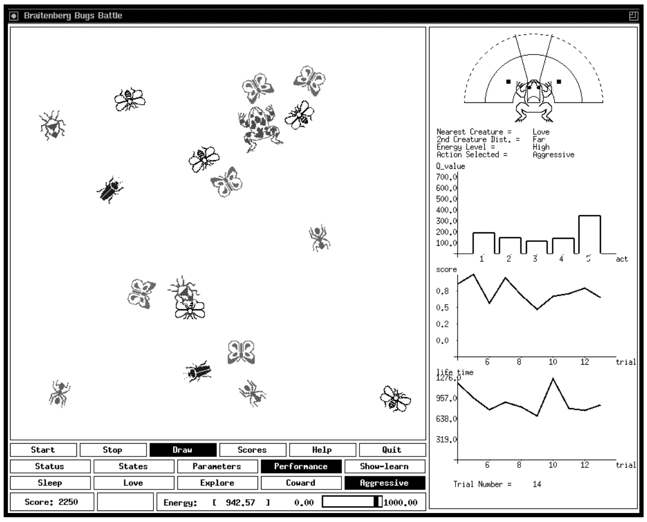


Figure 4: Complex environment

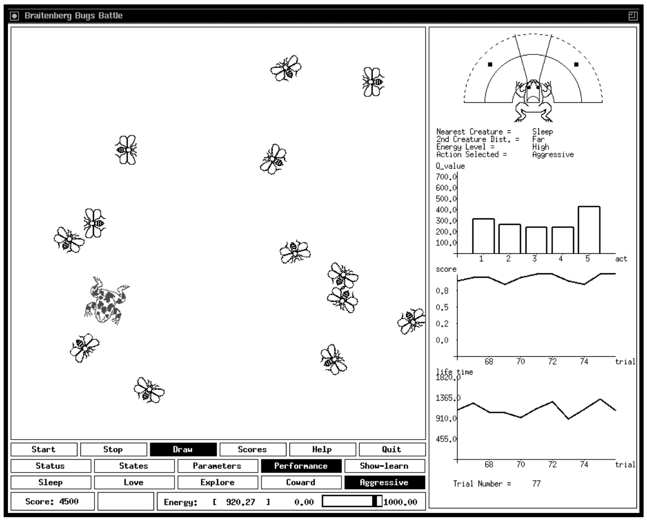


Figure 5: Simple environment

4.1 State Representation

A continuous state space can be represented by a set of discrete or continuous variables. A discrete representation permits faster learning rates and allows for a more direct interpretation of the resulting control policy, since the evaluation function is represented by a table. However the number of discrete variables increases exponentially with the problem size, becoming easily unmanageable. In the case of continuous variables, the evaluation function is represented by a function approximator (e.g. a neural network). This approach generates a compressed encoding, which is less sensitive to problem dimensionality, and induces generalization, but it may take an impracticable amount of time to be learned.

The decision of using a discrete representation was made considering the desired rate of adaptation, the fact that the control composition approach compresses the state space, and the uncertainty inherent to sensory information, forcing the use of qualitative information in discrete ranges.

When selecting a discrete representation, several issues should be considered, such as:

- Size of the state space. Unnecessary states may slow down the learning process without a significant increase in performance;
- Elimination of hidden states that degrade the learning.

The discretization of the state space used for this task is presented in Figure 6 (right). The following are the five state categories perceived by the learning agent (*frog*). The *Nearest creature distance* and the *Creature position* are represented by the distance and the angular position with respect to the learning agent. The sensor range of all agents is limited in distance, and in field of view (front 180 degrees). The dashed semi-circle in Figure 6 represents the limited range of the light sensors, and, in the case of the learning agent, also the limited range of type of creature detection. The internal semi-circles represent (starting from the innermost): the region where the center point of a prey must be in order to be considered a capture [1], *near*, *intermediate*, and *far* distances. The two central lines separates the *front* and *side* regions.

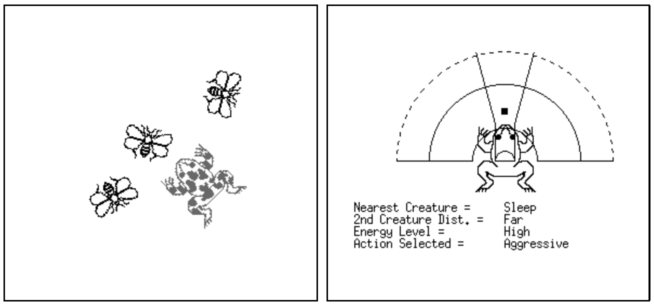


Figure 6: State representation

Usually, discretization produces ambiguous states due to deprived sensory information. Figure 7 presents a situation of a partially observable state (hidden state). These are different real states, but the agent cannot distinguish them, since they are perceived identically, causing ambiguity in the learning process. In the top example the aggressive behavior will cause the agent to miss both *prey*, simply because it is sensing similar stimuli in both light sensors and therefore moving in a straight path. In the bottom example it will result in a successful attack.

[1] As a second condition for considering a capture, the action selected by the learning agent must be *aggressive*.

State Representation:

- *Number of states: 216*
- *State Categories:*
  - Nearest Creature Type*:*
    - *(Sleep, Love, Explore, Coward, Aggressive, No Creature in Sensor Range)*
  - Nearest Creature Distance*:*
    - *(Far, Intermediate, Near)*
  - 2nd Nearest Creature Distance*:* [2]
    - *(Far, Near)*
  - Creature Position*:*
    - *(Front, Side)* [3]
  - Energy Level*:*
    - *(High, Middle, Low)*

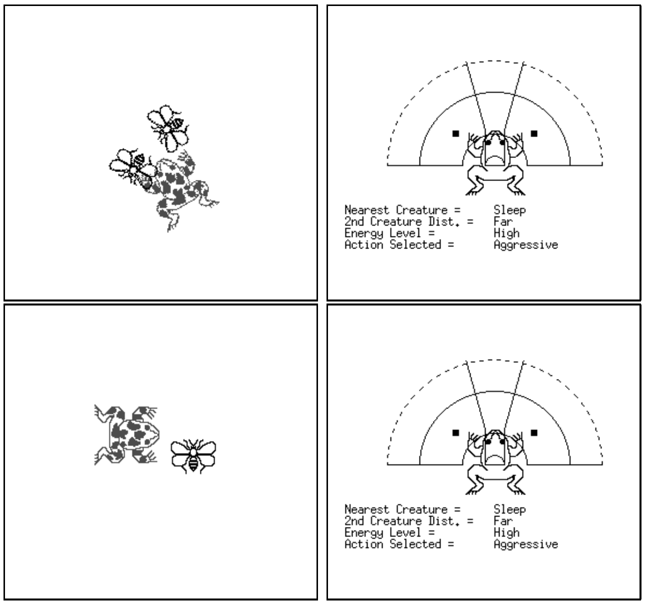


Figure 7: Hidden states

4.2 Reinforcement Signals

The selection of the reinforcement structure is also important for the performance of the learning agent, since the reinforcement and the perceived state space are the only information available to the learning process.

The learning agent receives one of the following reinforcement signals at each step of the simulation (negative reinforcement is consider a punishment and positive reinforcement, a reward for the actions performed): 750 for capturing a *Sleep* or a *Coward*; 1500 for a *Lover*; 1000 for an *Explorer*; -750 for an *Aggressive* [4]; -1 at the end of the trial; 0 otherwise. The energy level is incremented by 300, limited by the maximum energy 1000, every time that a prey is captured. Notice that the rewards are larger than the end-of-trial punishment. This structure gives priority to foraging over increasing its life-span. Since capturing prey is related to survival, the agent could, in principle, also learn foraging with only the end-of-trial reinforcement; however the sooner a good action sequence is rewarded, the faster the agent learns the task.

5 Learning Methods

Reinforcement learning is a method that does not rely on building models to attain an optimal control policy. It learns an optimal evaluation function by only receiving small reinforcements (rewards or punishments). The agent interacts with the environment, composing trial-and-error experience into reactive control rules. This paper applies two reinforcement learning algorithms to the problem of control composition: Q-learning and Q-learning with eligibility trace.

5.1 Exploration

Many reinforcement learning systems alternate between exploration and acting to improve the evaluation function. This tradeoff becomes an important issue when an optimal control policy is sought, because the amount of exploration that is necessary to find such a function is an open question. For example, the Q-learning convergence criterion to an optimal evaluation function requires each state-action pair to be tried infinitely often.

In the learning experiments presented here, exploration was attained by using a stochastic action selector that uses the Boltzmann probability distribution, which defines the probability of selecting action a in state x as:

$$p(a|x,T) = \frac{e^{Q(x,a)/T}}{\sum_{b\in A} e^{Q(x,b)/T}},$$

where T is the temperature, $Q(x,c)$ is the evaluation function for the state-action pair (x,c), for any $c \in A$, and A is the set of actions.

The percentage of exploration in a state x is determined by the temperature T and how distinct the evaluation function in this state is for different actions. Therefore, a key aspect of this method is the selection of the initial temperature value and its decay factor.

[3]This category was created to eliminate some hidden states that significantly degraded the learning, and it only discriminates if a second creature is nearby or not.

[3]Notice that there is no distinction between right and left side.

[4]After training, the learning agent will avoid the *aggressive* creatures.

5.2 Q-Learning

Q-learning is a Temporal-Difference reinforcement learning algorithm that learns an evaluation function of state-action pairs (Watkins, 1989). It is proven to asymptotically converge to an optimal evaluation function under the following conditions (Watkins, 1989; Watkins and Dayan, 1992):

- the task can be described as a Markovian decision process;
- the evaluation function is a lookup table;
- each state-action pair is tried infinitely often;
- a proper set of learning rates is selected.

These conditions are very difficult to fulfill. For example, by discretizing the input state space the second condition is fulfilled, but this usually generates hidden states, yielding a non-Markovian decision process. The cost of fulfilling both first and second conditions may be a huge state space, slowing down the learning. The third condition is subjective and dependent on the exploration procedure employed. The last condition is achieved empirically, which is, by itself, a hard task. Nevertheless, Q-learning shows normally a good performance even when these conditions are not completely satisfied. The Q-learning algorithm follows:[5]

Q-learning Algorithm:

1. *Define the current state **x** by decoding the sensory information available;*
2. *Use the stochastic action selector to determine an action **a**;*
3. *Perform action **a**, generating a new state **y** and a reinforcement **r**;*
4. *Calculate the temporal difference error $\hat{\boldsymbol{r}}$:*
$$\hat{r} = r + \gamma Q_{max} - Q(x,a)$$
5. *Update the Q-value of the state / action pair **(x, a)**:*
$$Q(x,a) = Q(x,a) + \beta\hat{r},$$
where $\boldsymbol{\beta}$ is the learning rate, $\boldsymbol{\gamma}$ is the discount factor, and
$$Q_{max} = \max_{k \in A}(Q(y,k));$$
6. *Return to step 1.*

---

[5]This algorithm is better described in (Watkins, 1989), and is only presented here for comparison with Q-learning with eligibility trace.

5.3 Q-Learning with Eligibility Trace

The idea of using an eligibility trace in Q-learning came from its application in another Temporal Difference Method ($TD(\lambda)$) (Barto and Sutton, 1983). The main idea is to keep events "eligible" (or retain the knowledge of their occurrence), allowing for the association between those events and later ones. Therefore, if a state receives a reinforcement, all the states that contributed to its achievement get a discounted reinforcement. The objective is to accelerate the learning process in delayed reward tasks like the one presented here.

A similar algorithm (Q(λ)) that also uses eligibility trace with Q-learning was described by (Peng and Willians, 1994). An analysis of the advantages of using eligibility trace in the way described in our paper was recently published (Singh and Sutton, 1996).

The Q-learning with Eligibility Trace algorithm operates as follows:

Q-learning with Eligibility Trace Algorithm:

1. *Define the current state **x** by decoding the sensory information available;*
2. *Use the stochastic action selector to determine an action **a**;*
3. *Perform action **a**, generation a new state **y** and a reinforcement **r**;*
4. *If ($a \neq arg\ max_{b\in A}(Q(x,b))$), erase the eligibility trace by setting all $e(x_i,a_i)$ to 0;*
5. *Calculate the temporal difference error $\hat{\boldsymbol{r}}$:*
$$\hat{r} = r + \gamma Q_{max} - Q(x,a)$$
6. *Set $e(x,a)$ to 1.0;*
7. *Update all $Q(x_i,a_i)$ and $e(x_i,a_i)$ values:*
$$Q(x_i,a_i) = Q(x_i,a_i) + \beta\hat{r}e(x_i,a_i)$$
$$e(x_i,a_i) = \alpha e(x_i,a_i),$$
where $\boldsymbol{\beta}$ is the learning rate, $\boldsymbol{\gamma}$ is the discount factor, $\boldsymbol{\alpha}$ is the eligibility factor, and
$$Q_{max} = \max_{k \in A}(Q(y,k));$$
8. *Return to step 1.*

The eligibility trace is erased if the action that will be performed in the new state is not the one associated with Q_{max}. This is required to maintain coherence, avoiding to follow non-optimal policies (Lin, 1993).

5.4 Experiments

All the experiments presented in this section are related to the simple environment (Figure 5) where only one type of prey, representing the sleep behavior, is present. Figure 8 shows the learning performance of Q-learning and Q-learning with eligibility trace. In these graphs, each point represents the sample mean of scores obtained over the same set of 30 random trials using the correspondent evaluation function (without exploration). The error bars represent the 95% confidence intervals for the scores' population means. These graphs show the two most distinct learning curves (*Run 1* and *Run 2*) out of 10 runs. For both algorithms the runs differ in the random sequence used at each trial. Q-learning exhibited the lower-type performance (*Run 2*) for 70% of the runs, converging on average to a sub-optimal evaluation function. On the other side, Q-learning with eligibility trace always converged to a near-optimal evaluation function, outperforming Q-learning on this problem, under the learning parameter setting described below.

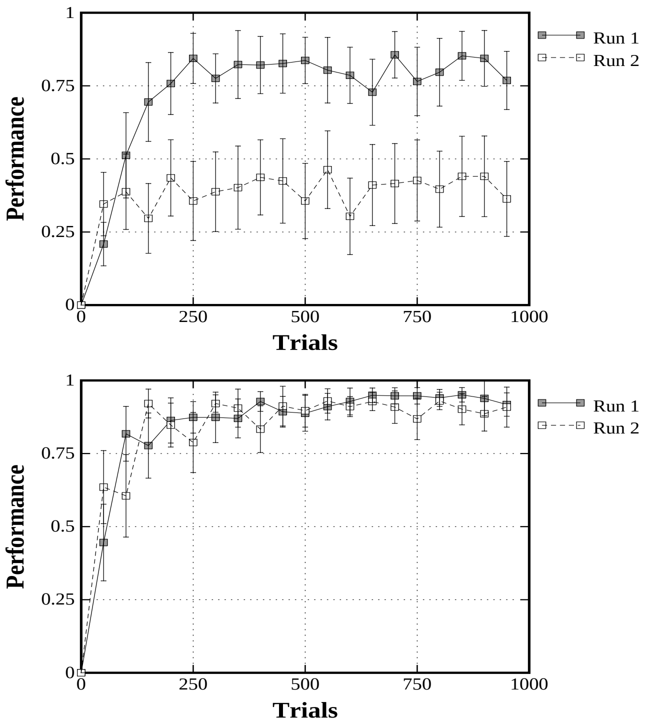


Figure 8: Learning curves using Q-learning (top) and Q-learning eligibility trace (bottom)

From the runs depicted in Figure 8, the evaluation functions that produced the best average performance on each run were saved. Their performance distributions over a set of 300 random trials are shown in Figures 9 and 10. Again, this test is performed on fixed state-action tables generated by the evaluation functions - no exploration is allowed during the test. Figure 11 shows a comparison between a state-action table produced by Q-learning with eligibility trace and a carefully handcrafted one. This indicates that the solutions found by this learning algorithm are near-optimal, although not significantly different from the handcrafted solution.

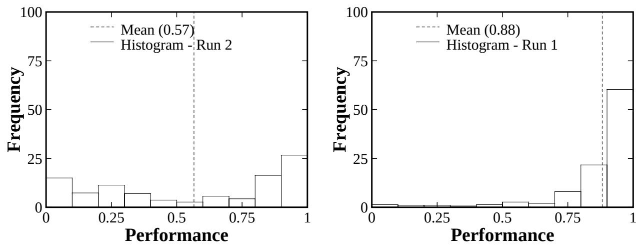


Figure 9: Q-learning performance histograms

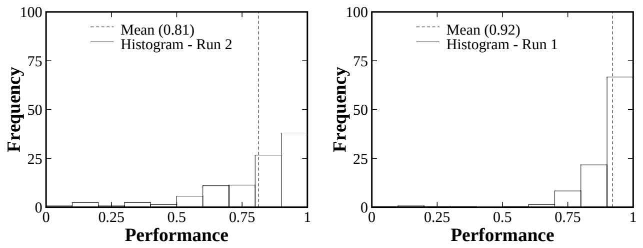


Figure 10: Q-learning eligibility trace histograms

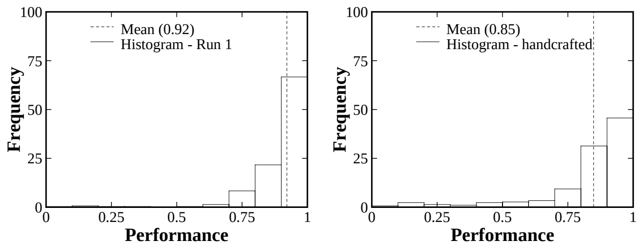


Figure 11: Q-learning eligibility trace (left) and a handcrafted state-action table (right) histograms

---

Learning Parameters:

- *learning rate* ($\beta = 0.025$);
- *discount factor* ($\gamma = 0.95$);
- *eligibility factor* ($\alpha = 0.95$);
- *initial temperature* $= 2000.0$;
- *final temperature* $= 1.0$;
- *temperature decay factor* $= 0.98$;
- *Q-values and eligibility trace were initialized with* 0.0;
- *The state-action table is updated at each step of the simulation.*

All the parameters were empirically selected. The exploration level does not decay completely after reaching the final temperature. An average of 10% of all the actions at the end are still caused by exploration. This happens because some states do not have a definite best action, due to hidden states.

6 Learning in a Complex Environment

After testing the above learning methods with the simple environment, the next step is to learn the complex environment (all the six types of creatures present). This environment presents an additional challenge, since the new four types of creatures continuously disturb the environment with their dynamics. The chance of capturing a prey by chance in this new situation is minimal, since the new types of creature (*love*, *coward*, and *explore*) are tuned to react to other creatures, or represent a negative reinforcement (poison) when captured (*aggressive*).

As depicted by the learning curve in Figure 12 (top), learning on this new environment is a harder task, the best performance produced by the Q-learning with eligibility trace does not reach the 75% level.

Shaping is known in experimental psychology and reinforcement learning as a method for accelerating the learning process (Gullapalli and Barto, 1992), and in extreme cases, making learning of complex behaviors possible. Shaping methods divide the learning task in stages; initially, reinforcement is given whenever a set of states containing the goal state is reached. As the agent reaches proficiency in each stage, it moves to the next stage, in which the reinforcement structure is refined, being applied only to states closer to the goal state. This process continues until the complex task is mastered.

The shaping approach used here manipulates the dynamics of the creatures (prey) over time. The idea is to start with almost static prey (high impairment) that are easy to capture, increasing the chance of reinforcement. Over the learning process, the impairment is reduced, letting the dynamic effects increase slowly. Appendix A presents a description of the dynamics of the creatures. The shaping procedure used is described below:

Shaping Procedure:

1. *Set impairment coefficient k ($k \in \{10, 5, 2\}$) to the maximum value ($k = 10$);*
2. *Set learning exploration to its maximum;*
3. *Learn new shaping stage on top of the evaluation function previously learned, until a performance index of 70% or 500 trials are reached;*
4. *If not in the last shaping stage ($k \neq 2$), set impairment coefficient to next value, and return to 2.*

Figure 12 compares the learning curves obtained by Q-learning with the shaping procedure (bottom) and without (top). As shown, the maximum performance of the learning algorithm without shaping is on average lower than 75%, while shaping produces an average performance above this limit. All curves presented here were obtained using the same experimental procedure as in Section 5.4, except that only 3 runs where performed.

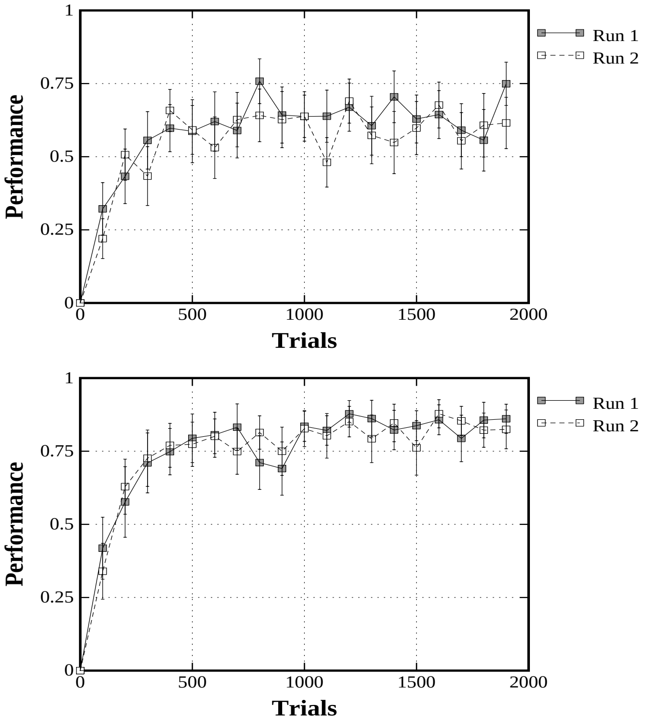


Figure 12: Learning curves using Q-learning with eligibility trace without (top) and with shaping (bottom)

Figures 13 and 14 show the improvement in performance given by the shaping procedure. In Figure 15, there is a comparison between the evaluation function created by the learning algorithm with shaping, and a carefully handcrafted version, indicating that the learning algorithm converged to a near-optimal solution.

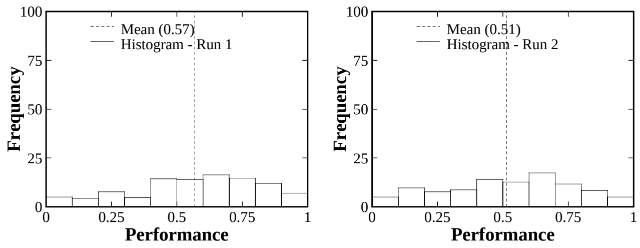


Figure 13: Performance histograms for learning without shaping

7 Conclusion and Future Work

In the simple environment, both learning methods presented interesting results. The discrete representation seems to generate too many hidden states, degrading the performance of the Q-learning algorithm. However,

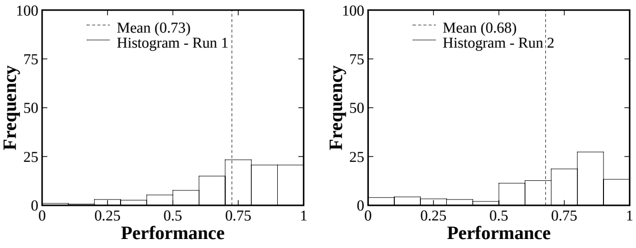


Figure 14: Histograms for learning with shaping

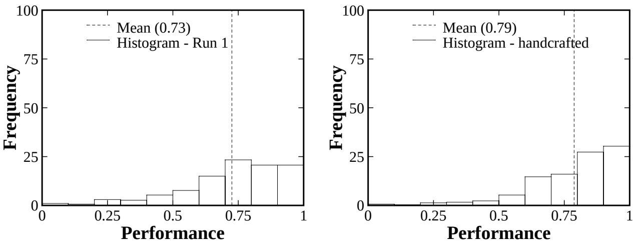


Figure 15: Histogram for shaping (left) and a hand-crafted state-action table (right)

Q-learning with eligibility trace seems to be less sensitive to these problems, performing similarly to our best guess for state-action tables. The complex environment is extremely dynamic, increasing even more the occurrence of hidden states. In this situation, the introduction of shaping improves the final performance of the learning algorithm.

The complexity of the problem presented was reduced by at least two orders of magnitude in state space size by expressing it as a control composition task. A set of very simple controllers and a simple learning procedure were the final requirements for solving this simulated foraging task. Control composition can be understood itself as a shaping procedure, where the agent acquires complex behaviors by composing relevant behaviors.

Research in control composition is starting and while this new paradigm shows some promise, it also creates a new set of challenges, such as how to design a control basis able to span the control complexity desired, how to compose adaptive controllers, and how to generalize control strategies over tasks.

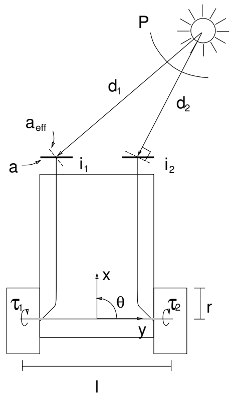
Acknowledgments

The research reported here was conducted in the Laboratory for Perceptual Robotics at UMASS and supported in part by NSF CDA-8922572, IRI-9116297, and IRI-9208920.

The authors would like to acknowledge the contribution of Manfred Huber especially in the design of Q-learning with eligibility trace, the comments of Prof. Paul Utgoff from an early version of this paper presented at a machine learning seminar in 1993, and to Sergio Gúzman-Lara for his advice on the experimental part of this paper.

A Simulation Dynamics

Each creature is modeled as a vehicle with two wheels, as in Figure 16.



Creature Parameters:

m	*Mass;*
I	*Moment of inertia;*
r	*Wheel radius;*
l	*Axle length;*
$i_1,\ i_2$	*Light intensity captured by left and right sensors;*
$d_1,\ d_2$	*Distances from light source to left and right sensors;*
$\tau_1,\ \tau_2$	*Torque applied to the left and right wheels;*
P	*Power irradiated;*
a	*Light sensor area;*
a_{eff}	*Effective light sensor area;*

Figure 16: Creature model

The following formulas approximate the dynamics of the vehicle. The velocity v was derived considering constant torque between time steps:

$$\begin{aligned} v_t &= v_{t-\Delta t} + \int_{t-\Delta t}^{t} \left(\frac{F}{m} - \frac{D}{m} v_t \right) dt \qquad (1) \\ \bar{a}_t &= \frac{v_t - v_{t-\Delta t}}{2} \\ s_t &= s_{t-\Delta t} + v_{t-\Delta t}\Delta t + \frac{1}{2} a_t (\Delta t)^2, \end{aligned}$$

where v_t is the velocity, $\bar{a}_t$ is the average acceleration, and s_t is the position, at time t. F is the force applied to the center of mass of the vehicle, and D is the damping factor.

Solving Equation 1 for the 3 degrees of freedom of the vehicle (x, y, θ), results in:

$$\begin{aligned} \dot{x}_t &= \left(\dot{x}_{t-\Delta t} - \frac{F cos(\theta_{t-\Delta t})}{D_{tr}} \right) e^{\frac{-D_{tr}}{m}\Delta t} + \frac{F cos(\theta_{t-\Delta t})}{D_{tr}} \\ \dot{y}_t &= \left(\dot{y}_{t-\Delta t} - \frac{F sin(\theta_{t-\Delta t})}{D_{tr}} \right) e^{\frac{-D_{tr}}{m}\Delta t} + \frac{F sin(\theta_{t-\Delta t})}{D_{tr}} \\ \dot{\theta}_t &= \left(\dot{\theta}_{t-\Delta t} - \frac{\tau}{D_{rot}} \right) e^{\frac{-D_{rot}}{I}\Delta t} + \frac{\tau}{D_{rot}}, \end{aligned}$$

where D_{tr} and D_{rot} is the translational and rotational damping, respectively.

The force F and torque τ are given by:

$$F = \frac{\tau_1 + \tau_2}{2kr} \qquad \tau = \left(\frac{\tau_1 - \tau 2}{kr} \right) \frac{l}{2},$$

where k is the impairment coefficient (1 for the learning agent, and 2 for the other creatures).

The light intensity $(i_j(k))$, collected by sensor j of vehicle k from all other vehicle in the environment($s \neq k$), is computed by the formula below:

$$i_j(k) = \sum_{s \in S, s \neq k} \left(\frac{P}{4\pi\ d(s,k_j)^2} a_{eff}(s,k_j) \right),$$

where $a_{eff}(s,k_j)$ is the effective area of incidence for the energy irradiated from vehicle s onto sensor j in vehicle k, $d(s,k_j)$ is the distance from vehicle s to sensor j of vehicle k, and S is the set of all vehicles in the environment which are in sensor range R, and within the front semi-circle. Also, if $d(s,k_j)$ is less than a minimum distance d_{min}, the distance is set to d_{min}. For details, see diagram on Figure 16.

The energy (e) of the learning agent is incremented by 300 every time that a creature is captured, and decreased at each step of the simulation in a rate that varies from 1 to 2 units, depending on its translational velocity, as described in the formula below:

$$\begin{aligned} e_t &= e_{t-\Delta t} - 1 - \frac{\sqrt{\dot{x}^2_{t-\Delta t} + \dot{y}^2_{t-\Delta t}}}{V_{max}} \\ V_{max} &= \frac{F}{D_{tr}}, \end{aligned}$$

where V_{max} is the maximum translational velocity.

References

Barto, A. G. and Sutton, R. S. (1983). Neural problem solving. Technical Report 83-03, Department of Computer Science, University of Massachusetts, Amherst.

Barto, A. G., Sutton, R. S., and Watkins, C. J. C. H. (1990). Learning and sequential decision making. In Gabriel, M. and Moore, J., editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, chapter 13, pages 539–602. M.I.T. Press.

Braitenberg, V. (1984). *Vehicles - Experiments in Synthetic Psychology.* M.I.T. Press, Cambridge, MA.

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

Gullapalli, V. and Barto, A. G. (1992). 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. IEEE.

Huber, M., MacDonald, W. S., and Grupen, R. A. (1996). A control basis for multilegged walking. In *Proceedings of the International Conference on Robotics and Automation*, volume 4, pages 2988–2993, Minneapolis, MN. IEEE.

Lin, L.-J. (1993). *Reinforcement Learning for Robots using Neural Networks.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Maes, P. and Brooks, R. A. (1990). Learning to coordinate behaviors. In *Proceedings Eighth Nacional Conference on Artificial Intelligence*, volume 2, pages 796–802. AAAI Press / The MIT Press.

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

Peng, J. and Willians, R. J. (1994). Incremental multi-step q-learning. In Cohen, W. W. and Hirsh, H., editors, *Machine Learning: Proceedings of the eleventh International Conference*, pages 226–232, New Brunswick, NJ. Morgan Kauffmann Publishers.

Singh, S., Barto, A., Grupen, R., and Connolly, C. (1994). Robust reinforcement learning in motion planning. In *Advances in Neural Information Processing Systems 6*, pages 655–662. Morgan Kaufmann Publishers.

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with eligibility traces. *Machine Learning - Special Issue on Reinforcement Learning*, 22(1/2/3):123–158.

Sutton, R. S. (1990). Reinforcement learning architectures for animats. In *From Animals to Animats : Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 288–296, Paris, France. M.I.T. Press.

Sutton, R. S. and Barto, A. G. (1990). Time-derivative models of pavlovian reinforcement. In Gabriel, M. and Moore, J., editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, chapter 12, pages 497–537. M.I.T. Press.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards.* PhD thesis, Cambridge University, Cambridge, England.

Watkins, C. J. C. H. and Dayan, P. (1992). Techinal note: Q-learning. *Machine Learning - Special Issue on Reinforcement Learning*, 8(3/4):279–292.