Learning Hierarchical Control Structures for Multiple Tasks and Changing Environments

Bruce L. Digney*
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, 15232, USA
Phone (412) 268-7084
bdigney@ri.cmu.edu

Abstract

While the need for hierarchies within control systems is apparent, it is also clear to many researchers that such hierarchies should be learned. Learning both the structure and the component behaviors is a difficult task. The benefit of learning the hierarchical structures of behaviors is that the decomposition of the control structure into smaller transportable chunks allows previously learned knowledge to be applied to new but related tasks. Presented in this paper are improvements to Nested Q-learning (NQL) that allow more realistic learning of control hierarchies in reinforcement environments. Also presented is a simulation of a simple robot performing a series of related tasks that is used to compare both hierarchical and non-hierarchal learning techniques.

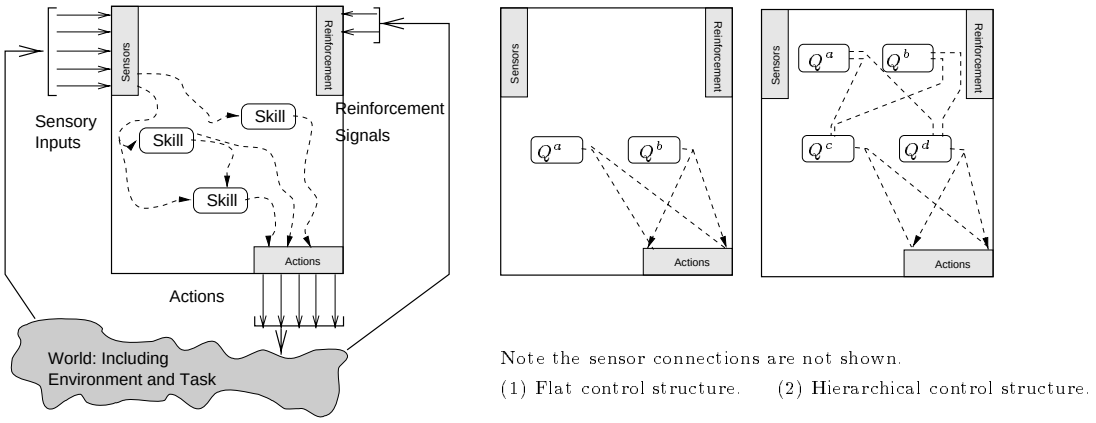
1. Introduction

The need for hierarchical structures within learning control systems is clear. Without some form of hierarchy, a learning system would be bogged down in the innumerable details of the lowest level of control. Many researchers have recognized this and have imposed hand crafted hierarchies. Although the imposed hierarchies result in more tractable problems, they also impose the designers preconceived notions on the control system. It is desirable that the control system able to generate its own hierarchical structure. Recently, work on Nested Q-learning (NQL) (Digney, 1996) has shown that a hierarchical structure can be learned in a reinforcement learning environment. Although this method generated hierarchical structures, it was considered to be seriously handicapped by the need to classify every distinct sensory state as a feature. With each distinct feature becoming the termination point for a new behavior the control system quickly became overwhelmed with choices of behaviors, of which the vase majority were irrelevant. In this paper, extensions able to remove that handicap are introduced. The solution is to have only useful features emerge. To do this two emergence criteria are introduced; high frequency of state occurrence and high reinforcement gradients. As only features with a high probability of relevance emerge learning will be much faster. A hierarchical learning control system is developed and tested against a non-hierarchical learning system.

2. Background

An issue closely linked to robot learning is the architectures in which the control strategies are implemented (Tyrrel, 1992). The two main ideologies are flat and hierarchical. Figure 1 shows the flat and the hierarchical architectures schematically. Flat architectures have direct connections from sensors to actions, through a single level of control. When some situation is perceived all behaviors compete for control with the strongest response winning. Hierarchical architectures have a more indirect coupling of perceptions to actions through a hierarchical control structure. When a situation is perceived, some high level behavior becomes active and issues commands to one or more lower level behavior(s). In turn, these lower level behaviors control yet lower level behaviors until primitive actions are activated and some physical action(s) is performed. Both architectures have benefits and drawbacks and have been used to implement learning techniques. One of the major benefits of hierarchical structures in robot learning is that learning difficult tasks can be made more tractable by clever design at the hierarchical structure. By abstracting away many details of a complex world to lower level behaviors, learning can more easily be implemented in the higher levels.

Many researchers have recognized the need for hierarchical structures in learning control systems (Maes and Brooks, 1990) (Long-Ji, 1993) (Dayan and Hinton, 1993) (Singh, 1992). These approaches are all similar in the respect that the structures are hand designed and individual components are learned or vise versa. Previous work in Nested Q-learning (NQL) (Digney, 1996) demonstrated the autonomous construction of a control hierarchy able to learn both the structure and the behaviors using a reinforcement learning technique based on Q-learning (Barto and Watkins, 1989). When the agent initially started out, no information about structure or



(a) Control schematic.

(b) Flat and Hierachical control structures.

Figure 1 Schematic of control architectures: (a) sensory, action and reinforcement signal configuration for a learning control system and (b) control architectures: (1) flat structure and (2) hierarchical structure. Note: $Q^?$ represents behaviors and the sensory connections in (b) have been removed for clarity.

useful behavior was given to it. The control system discovered distinct recognizable features in its world and learned the relationship between different features, its environment and its tasks. These relationships formed behaviors which were usually a hierarchical assembly of other behaviors and primitive actions. The structure that emerged was of as many levels as the task required. Using NQL, the control system also learned bottom-up reactive/opportunistic functions for each behavior which served to 1) provide a high level command source at the top of the hierarchy and 2) facilitate the invocation of previously learned beneficial behaviors in new situations without the necessity of relearning them.

One of the simplifying assumptions made during the initial work on NQL was that all perceivable states in the environment were valid termination points for possible behaviors. Useful behaviors would then emerge out of this pool of candidate behaviors. This assumption works for only very small problem spaces. When the robot is faced with problems of any realistic size, the pool of candidate behaviors grows too large very quickly. In this paper, a method for removing this assumption and allowing only behaviors with a high probability of relevance to emerge, leaving all other behaviors undiscovered is presented.

3. Nested Q-Learning with Emergent Behaviors

Consider an animat placed in some world and expected to perform some task defined by delayed rewards. As the animat moves about in its world, it experiences reinforcements from various sources. High negative reinforcements result from damaging actions, nominally negative reinforcements represent energy expended and positive reinforcement signals represent successful performance of some task(s). It is also able to perceives changes in its state as it moves around. A typical reinforcement learning problem is to learn the state, x, action, u, evaluation function $Q(x,u)$ such that the total positive reinforcement received by the animat is maximized. This results in a single monolithic evaluation function $Q(x,u)$ that, although able to solve the current problem, the information gained, albeit likely gained at less cost, will be useless in new tasks.

Nested Q-learning decomposes the monolithic evaluation function into a hierarchical structure of smaller evaluation functions, each representing a behavior which can later be invoked and reused in a new task or environment. To achieve this, the NQL algorithm allows the selection of behaviors in addition to primitive actions. These behaviors are defined as learned sequences that bring about some distinct termination state. Originally, these states were defined as any distinct state. This was a severely limiting and unnatural assumption. In this paper, only behaviors that prove useful emerge while all other possible behaviors remain undiscovered. These emergent behaviors will be selected based on two criteria. First, high changes in reinforcement gradient and secondly, a high state occurrence rate or frequency of visits to a particular state.

As the animat moves around, it perceives itself in various states and entering most of these states results in nothing more than normally occurring reinforcement signals. However, some states will represent dramatic changes in reinforcement. It is reasonable to assume that high reinforcement signal gradients might represent a non-typical and likely useful location in the state space. The high reinforcement signal gradients will be used to differentiate useful features from irrelevant ones. The animat's state space, as perceived

by sensors will be discrete and represented by x, with $x \epsilon \{ 0, 1, \cdots x \cdots x_{max} \}$ where x_{max} is the maximum number of distinct states currently existing but can increase as new states are discovered and x is the animat's current state. As the agent arrives at state x it experiences some total reinforcement signal, R_{TOTAL}. This reinforcement signal can be high negative in situations such as collisions with obstacles, but is usually some nominal value indicating the energy expended in reaching the current state. In addition to driving the adaptive mechanism, this signal can be used to learn a squared external reinforcement signal gradient mapping function $m(x)$ which learns the areas of non-typical reinforcement in the state space. This mapping will be used to generate potentially useful features in the state space and is determined as follows:

$$e_m = m(x) - \left(\frac{\partial R_{TOTAL}}{\partial t} \right)^2 \quad (1)$$

or

$$e_m = m(x) - \left(\frac{\Delta R_{TOTAL}}{\Delta t} \right)^2 \quad (2)$$

and

$$m(x) \longleftarrow m(x) - \eta e_m \quad (3)$$

where e_B is the environment, x is the current state, η is the learning rate and $\longleftarrow$ represents learning by some type of incremental parametric storage device such as a neural network. High changes in the gradient of the reinforcement signal during the transition from state to state, learned as a high value in $m(x)$, indicates a non-typical location in state space and a potentially useful feature. Eventually, as $m(x)$ converges, potentially useful features emerge using a simple threshold operator

$$x = \begin{cases} \text{feature} & \text{if } m(x) > T_1 \\ \text{non-feature} & \text{otherwise.} \end{cases} \quad (4)$$

where T_1 is the minimal squared external reinforcement gradient for a state to be considered as a feature.

Another criteria for useful behavior emergence is the occurrence or frequency of visits to particular state space locations. During operation, important states will be visited more often than less important ones. However, the animat must already be successfully performing one or more tasks for these features to emerge. At first, the reason for doing this might seem a bit unclear. Why generate new behaviors once the animat is operating successfully? The reason is that features useful in one or more tasks are likely to be useful in other, yet to be learned tasks. By learning the decomposition of the currently well learned tasks, portions of what has been learned can be used in new tasks. To learn this decomposition, a occurrence function $c(x)$ is defined. The strength of this function is increased by some small amount upon each visit to each state,

$$c(x) \longleftarrow c(x) + \epsilon \quad (5)$$

where ϵ is some incremental amount. This function is also periodically decayed by some decay factor,

$$c(x) \longleftarrow \lambda c(x) \text{for all } x \quad (6)$$

where λ is the decay factor, $0 < \lambda < 1$ and x denotes all currently discovered states. The result is the emergence of states that are more commonly visited or high occurrence. These commonly visited states can then be extracted as potentially useful features and used to generate behaviors. For the animat to extract such useful features from the occurred function $c(x)$, a threshold function similar to Equation 4 is used

$$x = \begin{cases} \text{feature} & \text{if } c(x) > T_2 \\ \text{non-feature} & \text{otherwise.} \end{cases} \quad (7)$$

where T_2 is the criteria for determining whether or not a state is visited often enough to warrant it emerging as a feature. When a single task is being performed, key states from the state trajectories are extracted. That is, the key states of the action state sequence which must be visited for successful outcomes. When more than one task is being performed, these equations decompose the action sequences of the tasks into sub-trajectories (sub-behaviors) that the tasks have in common.

Now that the two criteria for useful feature emergence have been developed, the extension to NQL can be discussed. The core of the NQL algorithm remains very much as previously described (Digney, 1996). However, instead of having all possible behaviors discovered and available for learning and use very soon after the animat begins operation, features emerge and are learned and used continuously throughout the life of the animat. This continual emergence of features causes added difficulties with learning and must be taken into consideration. This is because behaviors that emerge early will not be able to benefit from behaviors that emerge later. Again, consider an animat initially placed within a world without any form of previous experience or knowledge of its environment or task. The animat has some primitive actions, a, from which it can choose to physically act within its world. These are $a \epsilon \{ a_0, a_1, \cdots a_j \cdots a_J \}$ where a_j is a primitive action and J is the total number of primitive actions. Initially, without any behaviors having emerged, the possible actions available to the animat, u, are only primitive actions, $u \epsilon \{ Q^{a_0}, Q^{a_1}, \cdots Q^{a_j} \cdots Q^{a_J} \}$ where Q^{a_j} is a non-adaptive behavior representing the performance of primitive action a_j and should not be confused with the adaptive behaviors Q^{f_i} and u is the action to be taken by the animat. As the agent gains experience within its task and environment, potentially useful features will emerge according to the previously described criteria. These features, f_i, are labeled $f \epsilon \{ f_0, f_1, \cdots f_i \cdots f_I \}$ where f_i is a discovered feature and f_I is currently the last discovered feature. The total number of action (both primitive actions and

behaviors) that may be taken by the agent now includes both the primitive actions and the emergent behaviors, $u \epsilon \{ Q^{a_0}, \cdots Q^{a_j} \cdots Q^{a_J}, Q^{f_0}, \cdots Q^{f_i}, \cdots Q^{f_I} \}$. The total number of possible actions is the sum of all primitive actions and all currently possible behaviors, $u_{total} = J + I$. The state of the agent is established by the state of all the incoming sensors and is represented by a discrete value x_l which ranges from x_1 through x_{max}. The evaluation function for each feature is a function of the robot's current state and all possible actions, $Q^{f_i} = f(x, u)$. The evaluation function for each behavior now becomes, $Q^{f_i} = f(x_1, \cdots x_{max}, Q^{a_1} \cdots Q^{a_J}, Q^{f_1}, \cdots Q^{f_I})$, where both the number of states, x_{max}, and the number of behaviors, f_I, are open ended and subject to initial discovery and then to increases or decreases due to ongoing changes. It is seen that the learning algorithm described above becomes nested and possibly recursive. That is, the evaluation function, $Q^{f_{desired}}$, can invoke other behaviors including itself while attempting to reach the feature, $f_{desired}$. It is this nested nature that will allow hierarchical control structures to emerge.

As the agent interacts with the environment it receives an external reinforcement signal(s), r_{EXT}. It is through this signal that the agent is driven to perform tasks of external benefit. The reinforcement signal is defined as

$$r_{EXT} = \begin{cases} 0 & \text{if external task is achieved} \\ -R_{EXT} & \text{otherwise} \end{cases} \quad (8)$$

where r_{EXT} is an external reinforcement signal and R_{EXT} is a positive constant. In addition, there are various internal reinforcement signals, r_{INT}. These drive the agent to perform tasks of internal benefit such as *avoid danger* and *find fuel*. In the nested Q-learning algorithm there is also a reinforcement signal that effects only the currently active behavior(s). The reinforcement signal drives the action of the agent to reach the desired feature

$$r_{FEAT} = \begin{cases} 0 & \text{if } x = f_{desired} \\ -R_{FEAT} & \text{if } x \neq f_{desired} \end{cases} \quad (9)$$

where r_{FEAT} is the feature's reinforcement signal and R_{FEAT} is a positive constant. Upon performing a particular selected action, u^*, be it a primitive action or a behavior, the robot advances from state x_v to the next state x_w and incurs a total reinforcement signal, r_{TOTAL}. Included in this total reinforcement signal is the cost of performing the selected action. This cost includes the physical cost of performing the action and possibly the mental (computational) cost of choosing the action. These costs are designated as r_{LOW}, with

$$r_{LOW} = \begin{cases} -C \\ \quad \text{if } u^* \text{ is a primitive action} \\ \sum_{k=0}^{K} r_{TOTAL}^{u^*}(k) \\ \quad \text{if } u^* \text{ is an adaptive behavior} \end{cases} \quad (10)$$

where $\sum_{k=0}^{K} r_{TOTAL}^{u^*}(k)$ is the total reinforcement signal from the invoked behavior summed over the number of steps, K, required to perform the behavior, u^*, and C is a constant that reflects the cost of performing the primitive action. Of course, if many lower levels of behaviors are employed, this process becomes nested at r_{LOW} which represents the experiences (physical and mental) of all lower levels. The total reinforcement signal for the invoking behavior becomes

$$r_{TOTAL} = r_{EXT} + r_{INT} + r_{FEAT} + r_{LOW} \quad (11)$$

where r_{TOTAL} is the sum of all reinforcement signal sources. This total reinforcement is used to construct the Q functions of expected reinforcement, from which useful top-down control strategies will emerge. The error, e_Q, is defined to be,

$$e_Q = \gamma \cdot \max_u \{Q_{x_w,u}\} - Q_{x_v,u^*} + r_{TOTAL} \quad (12)$$

where γ is the temporal discount factor $0 < \gamma < 1$ and $\max_u \{Q_{x_w,u}\}$ is the current prediction of the maximum total future reinforcement remaining when the agent leaves state x_w. This error is used to adapt the evaluation functions,

$$Q_{x_v,u=u^*}(k+1) = Q_{x_v,u=u^*}(k) + \eta_Q \cdot e_Q \quad (13)$$

$$Q_{x_v,u\neq u^*}(k+1) = Q_{x_v,u\neq u^*}(k) \quad (14)$$

where η_Q is the rate of adaptation and k is the index of adaptation.

Features may emerge at any time during operation and their corresponding behaviors are added to the evolving behavior pool. Incorporating these newly introduced behaviors into the action selection mechanism is required. This makes action selection more difficult than in the action selection mechanism described in previous work (Digney, 1996). For the top-down goal directed action/behavior selection, a random based exploration policy is used. Although this is not the most effective exploration policy, it is easily implemented and other more efficient forms of exploration have been studied in depth elsewhere (Thurn, 1992). For the currently invoked behavior, $Q^{invoked}$, action/behavior selection is determined according to

$$u^* = \begin{cases} argmax_u\{Q^{ALL} + E_{invoked}\} \\ \quad \text{if } Q^{invoked} = Q^{f_i} \text{(a behavior)} \\ a_j \\ \quad \text{if } Q^{invoked} = Q^{a_j} \text{(a primitive action)} \end{cases} \quad (15)$$

where argmax_u is a maximum function taken over all primitive actions and existing behaviors, $E_{invoked}$ is the exploration policy and Q^{ALL} is all possible choices including both other behaviors and primitive actions. Note, that there can be more that one behavior invoked at any

time. In fact, it is usual that many behaviors will be active at the same time in response to either top-down or bottom-up directives. The nested nature is seen again in Equation 15 where the currently active behavior is able to select another behavior or a primitive action. If another behavior is selected that behavior may also go on and select yet another behavior and so on. If a primitive action is selected, Q^{a_j}, then the physical actuator action a_j is performed. The exploration policy for behavior $Q^{invoked}$ that is discovered at time d_i is

$$E_{inv} = RAND(k_{inv} STEP(t^{inv} - d_{dis}^{inv}) e^{-\tau(t^{inv} - d_{dis}^{inv})} + k_{other} \sum_{i=0,\ i \neq inv}^{I} STEP(t^i - d_{dis}^i) e^{-\tau(t^i - d_{dis}^i)} \quad (16)$$

where t_i is the behavior local time measured relative to each behavior (and is effectively proportional to the number of times each behavior has been invoked), d_{dis}^{inv} is the time of discovery for the invoked behavior, d_{dis}^i is the discovery time for behavior, i, and t^{inv} and t^i are the behavior local time for the invoked, $Q^{invoked}$, and the other behaviors Q^i. The time constants, τ, control the rate at which the exploration contribution to action selection is reduced. The constants k_{inv} and k_{other} controls how much exploration is due to the invoked behavior and how much is due to other behaviors that may have been discovered later. The $STEP(w)$ is the a step function,

$$STEP(w) = \begin{cases} 1 & \text{if } w > 0. \\ 0 & \text{if } w < 0. \end{cases} \quad (17)$$

The first term directly following k_{inv} in Equation 16 represents the contribution to exploration due to the discovery of the currently invoked behavior itself. That is, the behavior will have an initially high exploration component, decaying towards pure exploitation. The second term of Equation 16 represents the contribution to exploration due to the discovery of other behaviors. If this second term were zero then behaviors that are discovered after earlier behaviors have converged would never be tried by those converged behaviors. Without partially refreshing the exploration component of the earlier behaviors, the earlier learned behaviors may never learn to benefit from the new behaviors as they emerge.

The preceding derivation describes a NQL technique through which a top-down action selection mechanism will generate a hierarchical control structure and cascade goal seeking commands downward through the structure as behaviors are discovered continuously throughout the life of the animat. Each behavior, whenever invoked, will in turn invoke other behaviors and/or primitive actions in an attempt to fulfill the desired goals of higher behaviors. The bottom-up or sensory based action selection mechanism and how it fits with the top-down action selection to result in a flexible reactive control hierarchy is described elsewhere (Digney, 1996) and is not explored in this paper.

4. Simulation

To evaluate the NQL algorithm and it's capabilities, (the emergent features and the subsequent learning of a hierarchical structure of behaviors), an animat and the two dimensional environment of Figure 2 were used. The animat was placed in a grid world 6 by 10 units in size as shown in Figure 2(b). This grid world was bounded by an impassable barrier and additional barriers could be erected within the world in any configuration desired. Different from related work (Digney, 1996) in which the animat had a number of different sensors, the animat in this study was only capable of perceiving its location within the grid world. In this study, the animat's primitive actions were capable of moving it *left*, *right*, *up*, *down* or *no motion* shown in Figure 2(a). These actions were non-deterministic and when invoked they only achieved their desired outcome with a probability of 0.75. Otherwise another action was taken randomly. Figure 2(b) also shows the initial configuration of the grid world. The animat always started from the indicated starting region located on the left most side. There were three goal locations labeled A, B and C, placed as indicated in Figure 2(b). Although their positions are indicated, the goals were not necessarily always active or visible to the animat. The goals were activated or made visible to the animat as desired to test the capabilities of the control system. An internal barrier was placed as shown in Figure 2(b), forcing the animat to pass through a narrow passage en route to the goals. In previous work, the animat learned to associate various goals with their appropriate external stimuli (i.e. blue overhead light caused animat to move toward blue floor panel). In this paper, such associations were not studied and the animat was instructed to seek all goals. This was implemented as a random selection over all goals that had been discovered up to that time.

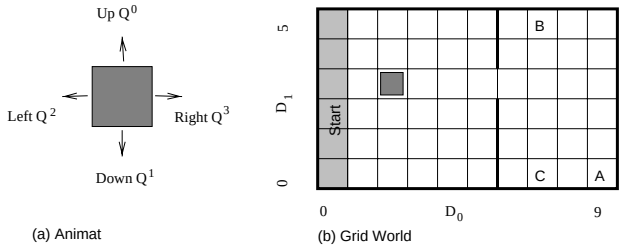


Figure 2 The animat and its grid world. Note the five primitive actions (up down, left, right and stay), the start locations, the initial barrier and the location of the three possible goals.

4.1 Emergent Features, Behaviors and Structures

The world was initially configured with barriers as shown in Figure 2(b). Two goals, A and B, were activated and goal C was left undetectable. The control system initially had no behaviors available to it as shown

in Figure 3(a) and its actions were thoroughly random and undirected. This can be contrasted to previous work (Digney, 1996) in which the control system was quickly flooded with potential useful behaviors and then had to sort out the useful from the many irrelevant. Once placed in its world, the animat randomly moved about undirected. After a short period of time it learned that a high reinforcement signal gradient occurred near active goal locations (the animat first discovered goal A before goal by chance). This constituted the discovery of a feature, f_0, and the behavior Q^5 emerged. The subsequent expansion of the control system at the emergence of Q^5 is shown in Figure 3(b). Next, the second goal location (goal B) feature, f_1, was discovered and the control system expanded to include behavior Q^6 as shown in Figure 3(c).

Note that at this point only two goals are currently active and two behaviors in existence. These have resulted from two reinforcement gradient based features emerging from $m(x)$. As the animat performed these two behaviors, two other features, based on high occurrence peaks in function $c(x)$ were discovered: f_2 at spatial location $D_0 = 5$ and $D_1 = 3$ and f_3 at $D_0 = 6$ and $D_1 = 3$. They emerged as the locations adjacent to the mouth of the barrier; clearly important locations in the state space. The hierarchical control system expanded to include behaviors Q^7 and Q^8, as shown in Figure 3(d). The two existing behaviors quickly integrated the new behaviors into their action sequences. The hierarchical structure of the control system is further shown in the new action sequence when behavior behavior Q^6 was activated is shown in Table 2 and Figure 4. This shows that the sub-behaviors, Q^7 and Q^8 are evident in the state-action-behavior trajectories of the higher level behaviors Q^5 and Q^6 (not shown). For convenience, the behaviors, features and locations discussed are summarized in Table 1.

At time ≈ 150, the third goal (goal C) was activated and made detectable to the animat. Shortly thereafter, its squared reinforcement gradient was learned and another feature, f_4 emerged at location $D_0 = 9$ and $D_1 = 0$. The control system expanded to incorporate the new behavior, Q^9 as shown in Figure 3(e). From the behavior sequence of Table 3 and Figure 5, it is seen that behavior Q^9 quickly exploited the existing behaviors Q^5, Q^6 and to a greater extent Q^7 and Q^8. The performance of the hierarchical control system and its component behaviors are shown in Figure 8(b). The primitive actions have a constant performance and are not shown. The events previously described are shown between time = 0 and time ≈ 150. For graphing purposes, any undiscovered behaviors were given a dummy performance value of -200 until their time of discovery. From this plot, it is seen that the learning of behavior Q^9 was considerably quicker than

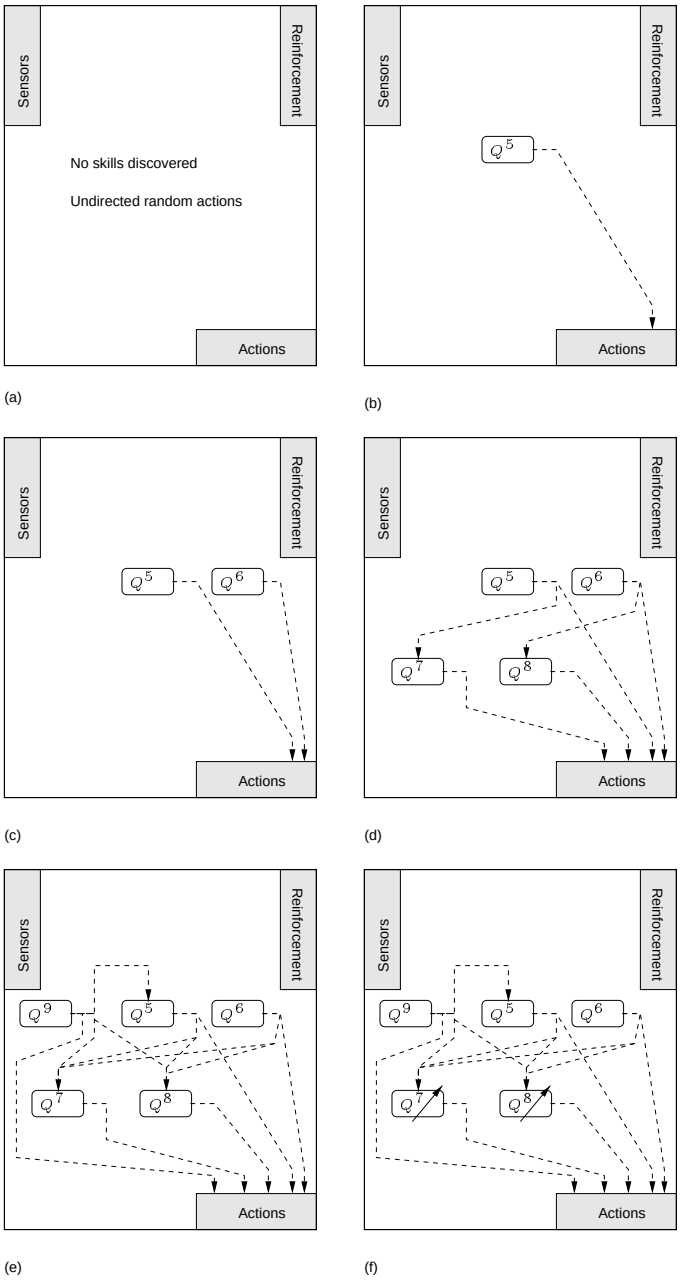


Figure 3 Emergent behaviors and structures.

Feature	Behavior	Location		Discovery	Comments
		D_0	D_1	Time	
NA	Q^0	NA	NA	NA	Primitive action
NA	Q^1	NA	NA	NA	Primitive action
NA	Q^2	NA	NA	NA	Primitive action
NA	Q^3	NA	NA	NA	Primitive action
NA	Q^4	NA	NA	NA	Primitive action
f_0	Q^5	9	0	≈ 50	First goal (A)
f_1	Q^6	7	5	≈ 50	Second goal (B)
f_2	Q^7	5	3	≈ 100	Door Entrance
f_3	Q^8	6	3	≈ 110	Door Exit
f_4	Q^9	7	0	≈ 150	Third goal (C)

Table 1 Behaviors, Features and Locations for the Hierarchical Control System.

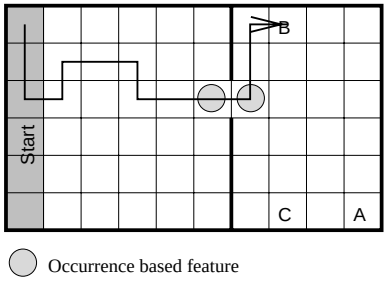


Figure 4 Animats's path for behavior Q^6.

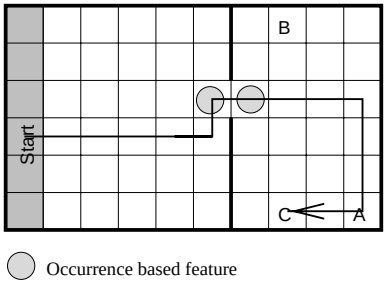


Figure 5 Animats's path for behavior Q^9.

Depth	Behavior	D_0	D_1	Comments
1	$->Q^6$	0	5	Start (Go to B)
2	$-->Q^8$	0	5	Go to door exit
3	$--->Q^1$	0	5	Primitive action
3	$--->Q^3$	0	4	"
3	$--->Q^3$	0	3	"
3	$--->Q^0$	1	3	"
3	$--->Q^7$	1	4	Go to door entrance
4	$---->Q^3$	1	4	Primitive action
4	$---->Q^3$	1	4	"
4	$---->Q^3$	2	4	"
4	$---->Q^1$	3	4	"
4	$---->Q^3$	3	3	"
4	$---->Q^3$	4	3	"
3	$--->Q^3$	5	3	At door entrance
2	$-->Q^0$	6	3	At door exit
2	$-->Q^3$	6	4	Primitive action
2	$-->Q^0$	7	4	At goal(B)

Table 2 Behavior Q^6 before the new barrier. See Figure 4 for path. Note the arrow length represents the depth into the structure.

Depth	Behavior	D_0	D_1	Comments
1	$->Q^9$	0	2	Start(Go to C)
2	$-->Q^3$	0	2	Primitive action
2	$-->Q^7$	1	2	Go to door entrance
3	$--->Q^3$	1	2	Primitive action
3	$--->Q^3$	2	2	"
3	$--->Q^3$	3	2	"
3	$--->Q^3$	4	2	"
3	$--->Q^0$	5	2	"
3	$--->Q^3$	4	2	"
3	$--->Q^3$	4	2	"
3	$--->Q^0$	5	2	At door entrance
2	$-->Q^5$	5	3	Go to A
3	$--->Q^3$	5	3	Primitive action
3	$--->Q^3$	6	3	"
3	$--->Q^3$	7	3	"
3	$--->Q^3$	8	3	"
3	$--->Q^1$	9	3	"
3	$--->Q^1$	9	2	"
3	$--->Q^1$	9	1	"
2	$-->Q^2$	9	0	At A
2	$-->Q^2$	8	0	At goal (C)

Table 3 Behavior Q^9 before the new barrier. See Figure 5 for path. Note the arrow length represents the depth into the structure.

the learning of the two previous reinforcement gradient based features, Q^5 and Q^6. This new behavior, Q^9, was learned more quickly because it was able to exploit the previously learned behaviors, Q^7 and Q^8, and to a lesser extent Q^5 and Q^6. The initial effort expended in learning to decompose the two behaviors Q^5 and Q^6 into Q^7 and Q^8 paid off by speeding the learning of the new behavior Q^9.

4.2 Recovery From Changes

Next the capabilities of the Nested Q-learning technique to adapt to changes were evaluated. After the third goal was introduced an additional barrier was introduced at and the animat was required to recover. At time ≈ 200, the configuration of the world was changed to the configuration shown in Figure 6. This configuration was almost identical to the initial one, except for the addition of a new barrier at $D_0 = 2$. This barrier effectively forced the animat to move to the lower edge of its world when traveling from the start location towards the goal locations on the right side. Behavior performance as displayed in Figure 8 (b) shows that after the introduction of the new barrier, the animat recovered and was able to relearn its control strategies.

From the resulting state-action-behavior trajectories for the relearned behavior Q^6, it is seen that most of the adaptation was confined to the lower level behaviors. This trajectory is shown in Figure 7 and Table 5. Figure 3(f) shows the final hierarchical structure with the lower levels adapted to the new barrier.

4.3 Comparison with Non-hierarchical Control Systems

An obvious criticism of this approach is that the complexities added by introducing intermediate behaviors in addition to the primitive actions will increase the number of choices available to the animat, thus making the learning more difficult. For a single task this is indeed correct, but it is the improvement in the animat's ability to learn

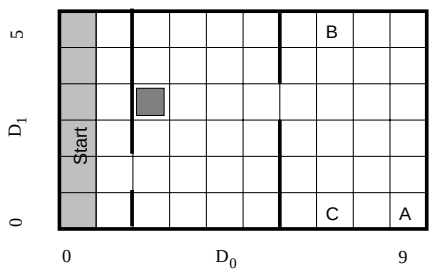


Figure 6 Animat's gridworld with new barrier introduced.

Table 4 Behaviors, Features and Locations for the Non-Hierarchical Control System.

Feature	Behavior	Location		Discovery	Comments
		D_0	D_1	Time	
NA	Q^0	NA	NA	NA	Primitive action
NA	Q^1	NA	NA	NA	Primitive action
NA	Q^2	NA	NA	NA	Primitive action
NA	Q^3	NA	NA	NA	Primitive action
NA	Q^4	NA	NA	NA	Primitive action
f_0	Q^5	9	0	≈ 50	First goal (A)
f_1	Q^6	7	5	≈ 50	Second feature (B)
f_2	Q^7	7	0	≈ 150	Third feature (C)

many tasks over its lifetime that is argued to be worth the poorer performance at learning a single, isolated task. To compare the NQL techniques of emergent hierarchical control with non-hierarchical control, an identical set of tasks was learned by a non-hierarchical control system. The non-hierarchical control system is presented with the same animat/grid world/barrier/multi-task scenario as was described and presented to the hierarchical control system in the previous sections.

The non-hierarchical control system had identical learning parameters to the hierarchical system, except that it was not allowed to generate hierarchies. In fact, the identical simulation code was used, with the hierarchy generating capabilities disabled. The non-hierarchical control system was presented with an identical sequence of events. That is, initially only two goals were detectable, then, later, a third goal and then a new barricade were introduced. The performance of all these tasks, for both the non-hierarchical and hierarchical control systems are shown in Figure 8. It was seen that both hierarchical and non-hierarchical control systems performed poorly at learning their first tasks, namely Q^5 and Q^6. However, at a later time when a new task was introduced, the non-hierarchical system had to learn it from scratch whereas the hierarchical system could use previously learned behaviors whenever beneficial. Similarly, whenever changes occurred (e.g. the new barrier) the non-hierarchical control system had to adapt each behavior individually while in the hierarchical adaption was confined to the the elemental behaviors effected and minimized the effect on the other behaviors. The performance at the third task (locating goal C) for the hierarchical and non-hierarchical control systems is shown in Figure 9(a) and (b) respectively. Note that the third task in the hierarchical control system is Q^9 and in the non-hierarchical control system it is Q^7. This encapsulation of control strategies into behaviors allowed the hierarchical control system to clearly out-perform the non-hierarchical control system, especially when new tasks were introduced and changes in the world occurred.

To further quantify this improvement, a measure of the overall performance was introduced. This measure, is the usage weighted average performance at any given time of all the behaviors that currently existed within both the non-hierarchical and hierarchical control systems and is plotted in Figure 10(a) and (b), respectively. The rate of learning is similar on the first task, but as new tasks and changes are introduced, the non-hierarchical control system is clearly out-performed by the self-generating hierarchical control system.

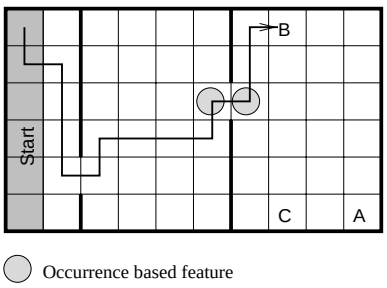
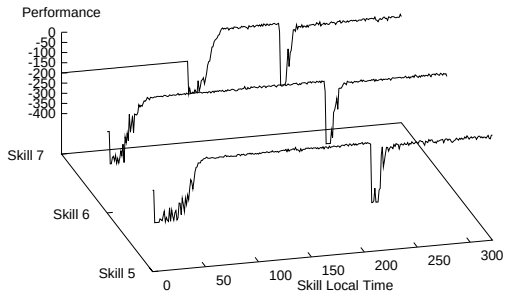


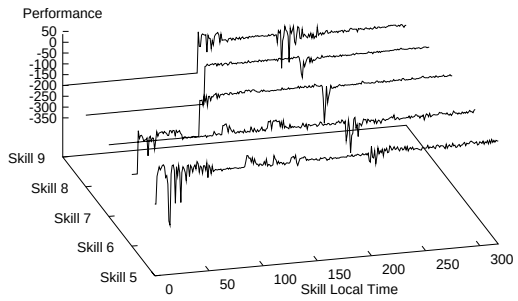
Figure 7 Animats's path for behavior Q^6 after new barrier.

Depth	Behavior	D_0	D_1	Comments
1	$-> Q^6$	0	5	Start (Go to B)
2	$--> Q^8$	0	5	Go to door exit
3	$---> Q^1$	0	5	Primitive action
3	$---> Q^1$	0	4	"
3	$---> Q^1$	1	4	"
3	$---> Q^1$	1	3	"
3	$---> Q^1$	1	2	"
3	$---> Q^7$	1	1	Go to door entrance
4	$----> Q^3$	1	1	"
4	$----> Q^0$	2	1	"
4	$----> Q^3$	2	2	"
4	$----> Q^3$	3	2	"
4	$----> Q^3$	4	2	"
4	$----> Q^0$	5	2	At door entrance
3	$---> Q^3$	5	3	At door exit
2	$--> Q^0$	6	3	Primitive action
2	$--> Q^3$	6	4	"
2	$--> Q^3$	6	5	At goal (B)

Table 5 Behavior Q^6 after the new barrier. See Figure 7 for path. Note the arrow length represents the depth into the structure.

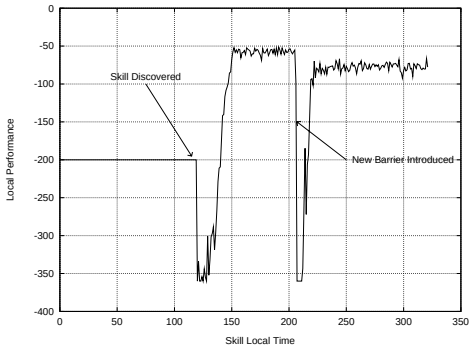


(a) Non-Hierarchical Structure

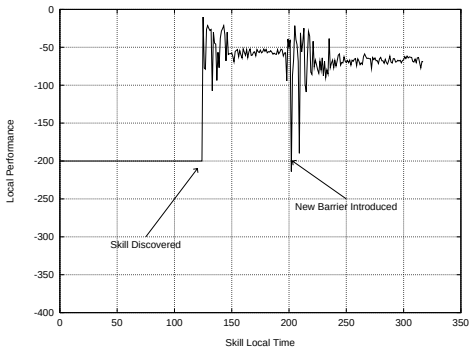


(b) Hierarchical Structure

Figure 8 Comparison of Performance: (a) Non-Hierarchical (b) Hierarchical.



(a) Non-Hierarchical Structure



(b) Hierarchical Structure

Figure 9
Performance of a new behavior for (a)non-hierarchical(Q^7) and (b)hierarchical (Q^9) control systems. Note that the time of discovery and barrier addition are indicated.

5. Discussion

In the preceding simulation studies two criteria were used to discover useful features. The squared external reinforcement signal gradient and the recurrence of states. The squared external reinforcement gradient criteria led to the discovery of goal locations within the state space. Although not explicitly simulated in this paper, had there existed areas of high negative reinforcement signals, these too would have been discovered as useful features. Although they would not be directly useful they would be indirectly practical, representing areas of the state space to be avoided (behaviors that are learned to **NOT** be active). The occurrence criteria led to the discovery of important locations of the state space that were used during task performance. These common locations became the point for decomposition of the tasks into smaller sub-tasks (or sub-behaviors).

By decomposing its behaviors into smaller sub-behaviors, the animat's control system demonstrated a capability to reuse existing knowledge and improve its performance at learning new but related tasks. This was evident when a new behavior (Q^9) was requested once the control system had learned the first two tasks (Q^5 and Q^6) and decomposed them into their elemental components (behaviors Q^7 and Q^8). As the animat began to learn behavior Q^9, it invoked behavior Q^7 to move the animat to the mouth of the barrier in one step. At this time behavior Q^7 was operating very well and this allowed a single command from the infant behavior Q^9 to move all the way from the barrier mouth. Behavior Q^5 was then invoked to bring the animat to within two steps of its goal. The invocation of Q^5 resulted in a somewhat inefficient overall path to the goal, but when one thinks about this, it is how animals and humans often behave. They often prefer to use established behaviors and routines at the expense of efficiency.

Another benefit of the decomposition of behaviors into a hierarchical structure is the resistance to change. High level behaviors and any new behaviors that are to be learned are ultimately composed of some number of lower level behaviors. When changes occur will often be confined to the lower level behaviors that are affected and the higher level behaviors will remain intact. In these simulations, when a new barrier was placed in between the start location and the initial barrier, most of the disruption was confined to the behaviors that moved the animat from the start to the initial barrier's mouth, namely behaviors Q^7 and Q^8. Some disturbance was shown in all behaviors, but this was to be expected because lower level behaviors during their period of recovery effected the higher level behaviors by briefly feeding back higher

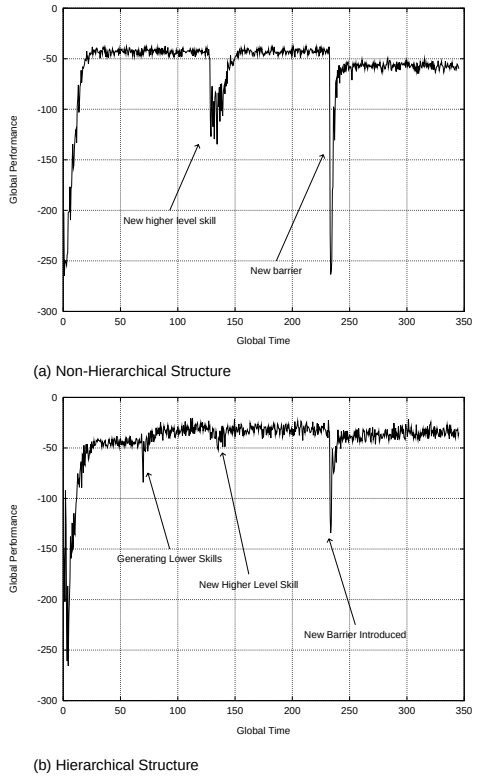


Figure 10 Overall behavior performance for (a)non-hierarchical and (b) hierarchical control systems.

negative reinforcements. Once the brief disturbance settled, the lower level behavior had learned how to navigate through the additional barrier and the higher level behaviors had remained the same. If the simulation were to be continued, additional recurrence based features would be discovered near the doorway of the new barrier.

To compare the benefits of the self-generating hierarchical structure with non-hierarchical control systems, an identical animat was used in an identical situation. From the results, it is seen that the non-hierarchical structure was forced to learn everything from scratch whenever a new task was attempted. Also, when changes occurred, each behavior had to be adapted in its entirety, resulting in much poorer performance.

It is clear that for this example the self-generating hierarchical control structure of NQL can outperform non-hierarchical control structures. However, there is much more work that must be done before such systems can be used in real applications. Most notably, the technique described in this paper still uses discrete spaces, states, actions and behaviors and generalization within these states, actions and behaviors is not possible. Operation within continuous space and generalization methods for NQL are currently being pursued (Digney, 1998) . This should allow the operation of applications in more realistic situations with real sensors and actuators. Also, the results presented indicated that the animat was beginning to view its world at a rudimentary level of abstraction. This abstraction will be necessary for learning at higher levels where seeing the world with a high level of detail recognition would be unnecessary and impeding. Current work will further concentrate on techniques for generating such abstracted features in realistic sensory systems.

6. Conclusions

A Nested Q-learning technique was developed for generating hierarchical control structures for a simple animat. Unlike previous work, this technique employed some simple selective criteria that allowed only those features with a good probability of becoming useful to emerge. This resulted in faster learning of the hierarchical control structure. The structure itself was shown to represent the abstracted higher level behaviors and their decomposed elemental sub-behaviors. Decomposition resulted in information easily being transferred to new but related tasks and a higher resilience to change as changes were confined to the behavior level which they affected, leaving most other behaviors untouched. The self-generating hierarchical control system was shown to outperform a non-hierarchical control system whenever new tasks were added or when change occurred. Although not yet ready for realistic application, the steps necessary for advancement have been outlined.

References

Barto, A.G. Sutton, R. and Watkins, C. (1989). Learning and sequential decision making. In *COINS Technical Report.*

Dayan, P. and Hinton, G. (1993). Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5*, San Mateco, CA. Morgan Kaufman.

Digney, B. (1996). Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments. In Materic, editor, *From animals to animats 4: SAB 96*, pages 363–373, Cape Cod, USA. MIT Press-Bradford Books.

Digney, B. (1998). Learning in continuous domains with delayed rewards. In *SAB 98 (submitted).*

Long-Ji, L. (1993). Hierarchical learning of robot skills. In *IEEE International Conference on Neural Networks*, pages 181–186, San Francisco. IEEE Press.

Maes, P. and Brooks, R. (1990). Learning to coordinate behaviors. In *Eighth National Conference on Artificial Intelligence*, pages 796–802.

Singh, P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3/4):323–339.

Thurn, S. (1992). Efficient exploration in reinforcement learning. In *Technical Report CMU-CS-92-102.* Carnegie Mellon University, School of Computer Science.

Tyrrel, T. (1992). The use of hierarchies for action selection. In *From animals to animats 2: SAB 92*, pages 138–148, Massachussets. MIT Press-Bradford Books.