Emergent Hierarchical Control Structures: Learning Reactive/Hierarchical Relationships in Reinforcement Environments

Bruce L. Digney *
Defence Research Establishment Suffield
Box 4000, Medicine Hat, Alberta, CANADA, T1A 8K6
E_mail bdigney@dres.dnd.ca

Abstract

The use of externally imposed hierarchical structures to reduce the complexity of learning control is common. However, it is acknowledged that learning the hierarchical structure itself is an important step towards more general (learning of many things as required) and less bounded (learning of a single thing as specified) learning. Presented in this paper is a reinforcement learning algorithm called Nested Q-learning that generates a hierarchical control structure in reinforcement learning domains. The emergent structure combined with learned bottom-up reactive reactions results in a reactive hierarchical control system. Effectively, the learned hierarchy decomposes what would otherwise be a monolithic evaluation function into many smaller evaluation functions that can be recombined without the loss of previously learned information.

1 Introduction

The use of Q-learning [1] and other related reinforcement learning techniques is common for the control of autonomous robots [2]. However, long learning times combined with the slow speed (when compared to simulations) and the frailties of robot hardware present considerable problems when scaling up to real applications. Often, to reduce the problem to a more tractable size, the control problem is hand decomposed into a hierarchical structure [3]. This abstracts it into many smaller, more easily learned portions. However, hand decomposition imposes the designer's preconceived notions on the robot which, from the robot's point of view, may be inefficient or incorrect. Furthermore, it is acknowledged that for truly general learning and full autonomy to occur in the face of unknown and changing environments, the structure of the hierarchical control system must also be learned.

Presented in this paper is the Nested Q-learning algorithm that allows the generation of hierarchical control structures in reinforcement learning domains. Once the structure has been learned, skills that have been previously mastered can be used in future tasks and environments. This continual carrying forward of learned information is called life-long learning and provides the robot with a head start when learning new tasks [4]. As the robot moves from task to task and environment to environment it will have the accumulated information of its past experiences available to it as skills. These transportable skills will allow the robot to learn progressively more complex tasks. Eventually this continual learning will allow the robot to learn tasks that would be impossible in a simple monolithic network. Having the robot learn a hierarchical control structure is analogous to a student who invests initial effort in discovering the underlying principles or structure of a problem, rather than simply memorizing a monolithic solution. Once the underlying principles are understood they can be transferred to more difficult tasks. Such is not possible if solutions are only memorized, making the information gained albeit at a lesser expense, useless in new situations. This continual learning lends itself to use in pre-training or shaping, either by chance or through a regimented training program. In nested Q-Learning controlled robots the use of scaffolding actions and staged learning to exploit this online skill transfer between task/environment settings, is discussed in more detail elsewhere [4].

2 Learning Hierarchical Structures

2.1 Introduction

Consider the example of a robot perceiving its world through sensors and receiving internal and external reinforcement as pictured schematically in Figure 1(a). This robot is capable of acting on its world using a number of primitive actuator movements. These primitive actions are at the simplest level of the robot's actuators and although they may utilize feedback control mechanisms, they do not embody any higher intelligence. The robot also receives reinforcement signals which are critical indicators of how the robot is progressing with respect to the completion of some desired task(s). These critical signals are all the direction that can be assumed for the autonomous robot. Critical error signals simply provide negative reinforcement when the actions of the robot do

* The author acknowledges support from the Canadian Department of National Defence (DND) and the Natural Sciences and Engineering Research Council (NSERC).

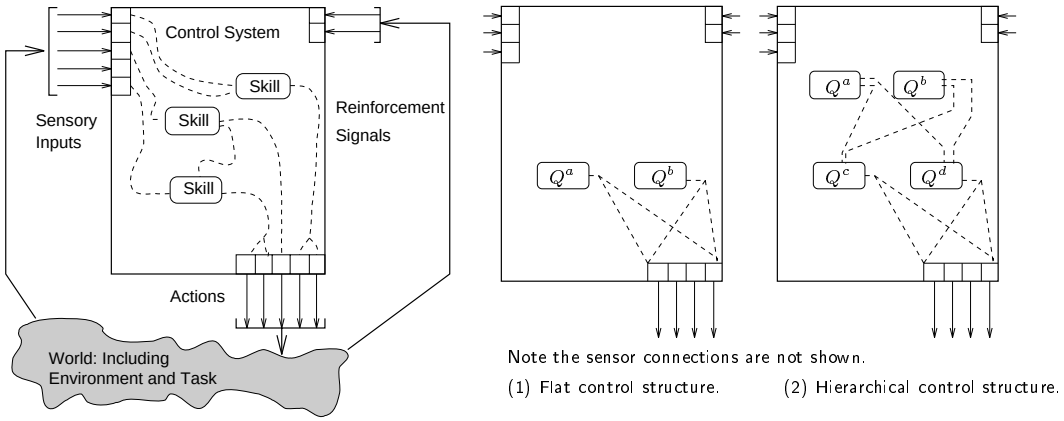


Figure 1: Schematic of control architectures: (a) sensory, action and reinforcement signal configuration for a learning control system and (b) control architectures: (1) flat structure and (2) hierarchical structure.

not achieve the goal and positive (favorable) reinforcement whenever the actions achieve the goal.

In the field of intelligent control, the strategies connecting the sensors to the actions are either hardwired by a designer, taught to the agent or left to be autonomously learned by the agent. There are many variations [5] [6] of architectures in which such control systems may be implemented. The two main variations are flat and hierarchical as shown schematically in Figure 1(b), (1) and (2), respectively. Note that the hierarchical control system relies upon higher level skills being built upon lower level skills while the flat architecture contains only skills at a single level which interact directly with the actuators.

2.2 Related Work

Many researchers have recognized the need for hierarchical structures in learning control systems. Research most closely related to the approach described later in the paper will be briefly summarized.

Hierarchical learning control systems are often handcrafted with a number of low level behaviors controlled by a gating function. The low level behaviors map the current state of the agent into actuator activities while the gating function determines which low level behavior is to be active. Maes and Brooks [7] used this approach by hand-crafting the individual low level behaviors while the gating system was learned from a reinforcement signal during operation. Mahadevan and Connell [8] used the complementary approach in which a hand-crafted gating function and separate reinforcement signals which were supplied for each individual behavior. Lin [9] used a similar approach combined with staged learning to first train the individual behaviors and then train a gating function. Dayan and Hinton's Feudal Q-learning [10] used a hierarchical master/slave type architecture with a set of hand-crafted commands and corresponding reinforcement functions. The commands served as actions available to the high level master. As the master selected a command, a low level slave was tasked with its performance and was subsequently rewarded for the performance of the command regardless of whether or not it achieved the overall task. On the other hand, the master was rewarded only for the completion of the overall task. In this approach, learning occurs at both levels simultaneously. The lower levels learn the commands and the higher level learns when to invoke the low level commands to perform some desired task. Singh's Compositional Q-learning system[11] first learned elemental tasks and then the gating functions that switch between elemental tasks in the correct temporal sequence to achieve some high level goal. Kaelbling's Hierarchical Distance to Goal (HDG) [12] viewed the world on two levels of resolution. Regions, with their centres referred to as landmarks, were used to move an agent into the same region as the goal's location. Then local actions were used to move to the goal.

These approaches are all similar in the respect that the structures are hand designed and individual components are learned. In the nested Q-learning algorithm described in this paper, it is proposed that the structure itself be learned. The structure that emerges can be of as many levels as the task requires and not just two levels. When an agent initially starts out, no information about structure or useful behaviors is given to it. The agent discovers distant recognizable features in its world and learns the relationship between different features, its environment and its tasks. These relationships form skills which are usually a hierarchical assembly of other skills and primitive actions. Currently, features are restricted to being evenly distributed over raw sensor signals, but work to allow the discovery of higher level compound features is being pursued. Using nested Q-learning, the agent also learns bottom-up reactive/opportunistic functions which serve 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. Current work is extending the concept of bottom-up functions to include not only the learning of reactive/opportunistic situations, but the

conditions under which skills are applicable, as well.

In summary, the major advances of nested Q-learning over the described approaches are: 1) the ability to autonomously construct hierarchical structures of arbitrary depth from sensory and reinforcement signals, 2) learning occurs at all levels (converging form the actuators upward) and 3) the use of learned bottom-up reactive functions allows skills to invoke themselves and not necessarily require an invoking signal form another skill. These reactive functions serve to control the highest level of the hierarchy and allow for previously learned skills, of immediate benefit or danger, to be transfered between tasks and environments.

2.3 Nested Q-learning

A method will now be presented that will allow for the autonomous generation of hierarchical control structures as pictured in Figure 1(b)(1). Each incoming sensor has a finite number of perceivable distinct values. These distinct values are referred to as *features*. For example, consider a robot whose task it is to move between two rooms through an open door. In this task, the first feature or sensory condition the robot needs to perceive is the door directly ahead. The next relevant feature would be for the robot to perceive the door frame on either side of it. The last relevant feature would be for the robot to perceive the door directly behind it. Eventually control strategies will be learned to move the robot so as it can perceive these features. Current work will allow such complex compound features, such as doorways, to be recognized, but in this paper a simple even distribution of features over the range of each sensor will be assumed. For sensor s_n these features are represented by the distinct values

$$s_n \epsilon \left\{ \; s_n^1, \quad s_n^2, \cdots \quad s_n^m, \quad \cdots \quad s_n^{M_n} \; \right\} \tag{1}$$

where s_n is the n^{th} sensor, s_n^m, is the m^{th} distinct value and M_n is the number of distinct values for sensor n. For any number of incoming sensors their distinct values or recognizable sensory conditions will constitute *features* which may or may not prove useful in controlling the agent. These features, label f_1 through f_I, are defined over all distinctive values (1 to M_n) for all sensors (1 to N),

$$f_i \epsilon \left\{ \; s_1^1 \cdots \quad s_1^{M_1}, \cdots \quad s_n^{m_n}, \cdots \quad \cdots \quad s_N^{M_N} \; \right\} \tag{2}$$

where f_i is the i^{th} distinct feature and N is the number of sensors. The total number of features is, $I = \sum_{n=1}^{N} M^n$.

A Q-learning evaluation function is now defined for each distinct feature, Q^{f_i}. In this method, with features defined as distinct recognizable sensory conditions, skills become the control strategies required to cause the robot to attain those particular sensory conditions. Eventually the lowest level primitive actions must be invoked by the learned control strategies. These primitive actions are the simple low level actuator movements with which the robot acts on its world. They may contain some form of feedback control mechanism, but remain the simple building blocks out of which complex control strategies (skills) can be constructed. There are J primitive actions, labelled a_1 through a_J.

Each feature, f_i, is by definition the endpoint of a skill with its control strategy represented by the evolving evaluation function, Q^{f_i}. While attempting to reach some desired feature, $f_{desired}$, the control strategy can invoke any of the primitive actions, a_j (shown as Q^{a_j}) or any of the skills represented by all Q^{f_i}. The choice of possible actions, u, is now

$$u \epsilon \left\{ \; Q^{a_1}, \quad \cdots Q^{a_J}, \quad Q^{f_1}, \quad \cdots \quad Q^{f_I} \; \right\} \tag{3}$$

where u is a possible action or skill choice, Q^{a_j} is a non-adaptive primitive action (shown with similar notation to the Q functions for convenience) and Q^{f_i} is an adaptive skill. The total number of possible actions is the sum of all primitive actions and all currently possible skills, $u_{total} = J + I$. The state x_l of the agent is established by the state of all the incoming sensors, x_1 through x_L,

$$x_l \epsilon \left\{ \; x_1, \quad x_2, \quad \cdots \quad x_l \quad \cdots \quad x_L \; \right\} \tag{4}$$

where x_l is a distinct state of the agent. The evaluation function for each feature is a function of the robots current state and all possible actions, $Q^{f_i} = f(x, u)$. The evaluation function for each skill now becomes,

$$Q^{f_i} = f(x_1, \cdots x_L, Q^{a_1} \cdots Q^{a_J}, Q^{f_1}, \cdots Q^{f_I}) \tag{5}$$

where both the number of states, L, and the number of skills, 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 skills including itself while attempting to reach 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 these signals that the agent is driven to perform tasks of external benefit. This reinforcement signal is defined as

$$r_{EXT} = \begin{cases} 0 & \text{if external task is achieved} \\ -R_{\text{EXT}} & \text{otherwise} \end{cases} \tag{6}$$

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 skill(s). This reinforcement signal drives the action of the agent to reach the desired feature.

$$r_{\text{FEAT}} = \begin{cases} 0 & \text{if at desired feature} \\ -R_{\text{FEAT}} & \text{otherwise} \end{cases} \tag{7}$$

where r_{FEAT} is the feature's reinforcement signal and R_{FEAT} is a positive constant.

For the top-down goal directed action/skill selection the action chosen, u^*, is determined by

$$u^* \Leftarrow \begin{cases} \text{argmax}_u\{Q^{f_i \text{ or } a_j} + E\} & \text{if } Q^{f_i} \text{ active} \\ a_j & \text{if } Q^{a_j} \text{ active} \end{cases} \quad (8)$$

where argmax_u is the maximum function taken over all possible primitive actions and skills, Q^{f_i} is an adaptive skill (an evolving function of the state x and all possible actions u), Q^{a_j} is a primitive action, a_j is some specific action to be taken and E is the exploration strategy. The nested nature is seen again in Equation 8. The currently active skill can select another skill or a primitive action. If another skill is selected that skill can go on and select yet anther skill and so on. If a primitive action is selected, Q^{a_j} then the physical actuator action a_j is performed. The exploration component, E, of Equation 8 is required to ensure adequate exploratory coverage of the agent's state space. It can be random, error or recency based. In this development a recency based exploration policy is used which ensures that the actions (primitive/skill) that were taken less recently are favored over the more recent actions [13].

Upon performing the selected action, u^*, be it a primitive action or a skill, 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

$$r_{\text{LOW}} = \begin{cases} -C & \text{if } u^* \text{ is a primitive action} \\ \sum_{k=0}^{K} r^{u*}_{\text{TOTAL}}(k) & \text{if } u^* \text{ is an adaptive skill} \end{cases} \quad (9)$$

where $\sum_{k=0}^{K} r^{u*}_{\text{TOTAL}}(k)$ is the total reinforcement signal from the invoked skill summed over the number of steps, K, required to perform the skill, u^*, and C is a constant that reflects the cost of performing the primitive action. The total reinforcement signal for the invoking skill becomes

$$r_{\text{TOTAL}} = r_{\text{EXT}} + r_{\text{INT}} + r_{\text{FEAT}} + r_{\text{LOW}} \quad (10)$$

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,

$$e_Q = \gamma \cdot \max_u\{Q_{x_w,u}\} - Q_{x_v,u^*} + r_{\text{TOTAL}} \quad (11)$$

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,

$$Q_{x_v,u=u^*}(k+1) = Q_{x_v,u=u^*}(k) + \eta_Q \cdot e_Q \quad (12)$$

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

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

The preceding derivation describes a nested Q-learning technique through which a top-down action selection mechanism will generate a hierarchical control structure and propagate goal seeking commands downward though it. Each skill, whenever invoked, will in turn invoke other skills and/or primitive actions in an attempt to fulfil the desired goals of higher skills. The bottom-up or sensory based action selection mechanism and how it interacts with the top down action selection mechanism will now be described. This bottom-up selection mechanism will allow skills to invoke themselves without the need of a top-down command signal from a higher level skill. Consider the agent at state $x_{invoked}$ when the skill Q^{f_i} is invoked. Upon the arrival at the sensory conditions of the defining feature f_i, two things can occur: 1) an uneventful arrival at f_i with only nominal reinforcements occurring, or 2) arrival at f_i coincides with a markedly high negative or high positive reinforcement signal. Such a distinct change in reinforcement would represent possible reactive or opportunistic behaviors. The invocation of the selected skill (and in turn all other sub-skills and actions) from any other state may not necessarily result in such non-typical results or useful correlation. For instance, the invocation of the *avoid obstacle* skill would not yield any benefit if an obstacle was not present. Similarly, the invocation of the *begin feeding* skill would not be of any benefit if food was not near. The sensory conditions that are a precursor to reactive/opportunistic situations must be learned. In the first example, the sensory situations that indicate the presence of an obstacle must be learned in order to activate the avoid obstacle behavior when it is useful. By learning these situations in a bottom-up manner (outside of the top-down Q^{f_i} functions) it will become possible to have skills invoke themselves in new situations where the top-down functions have not yet been formed. By allowing skills to invoke themselves or make themselves more likely to be invoked will speed the learning of new, but related, tasks.

For example, consider a robot learning to deliver the mail. During the performance of this task it learns that it requires intermittent recharging. It soon learns how to recharge itself and also learns which sensory conditions are associated with an opportunity to recharge; possibly the appearance of a recharge outlet. In this task, recharging will be learned from scratch as a particular skill, composed of a structure of many sub-skills and actions. As the agent learns the recharging skill, a bottom-up activation function is formed for that skill as well. In the presence a fortuitous recharging outlet where there was not one before, the bottom-up function will respond strongly and attempt to override any top-down commands that might be active. Moreover, when the robot is set to learning a different task (e.g. sweep the floor) these bottom-up activations will remain valid. Now, if the robot comes across a recharging outlet at its new task, it will know how to respond, and know the proper actions to respond with, without having to relearn the recharging skill.

To implement these bottom-up reactive relationships an evolving function, $B^{f_i}(x_{invoked})$, is defined for each skill Q^{f_i}. This function learns to predict the reinforcement outcomes of invoking each skill from differ-

ent states. Effectively, these functions evolve into the bottom-up triggering response for reactive and opportunistic skills. Within an emergent hierarchy, such a bottom-up action selection mechanism will be in control at the very top of the hierarchy as there are no higher levels to invoke them in a top-down manner. They will also be able to subsume the current top-down flow of commands should a reactive or opportunistic situation present itself outside of the converged top-down regimented control strategies. If a bottom-up triggering situation presents itself with enough regularity, it will eventually be absorbed by the top-down control strategies. Bottom-up reactive behaviors are most valuable when the agent is in new situations where the top-down structure has not yet formed. They represent chunks of control structure and the skills within that were learned in the past that might be useful in new task/environment settings. Useful skills can be invoked in new situations without the need for the robot to relearn them. As these skills usually contain their own top-down structures, the bottom-up contributions will be important in transferring information between tasks and environments.

The error, e_B, for these functions is determined using the eventual reinforcement, r_{EVT}, that occurs at the defining feature.

$$e_B = r_{EVT} - B^{f_i}(x_{invoked}) \tag{14}$$

where $B^{f_i}(x_{invoked})$ is the bottom-up reactive function for skill Q^{f_i}, $x_{invoked}$ is the state from which Q^{f_i} was invoked and r_{EVT} is the total reinforcement when the feature has been reached. The reactive function is then adapted,

$$B^{f_i}(x_{invoked})(k+1) = B^{f_i}(x_{invoked})(k) + \eta_B \cdot e_B \tag{15}$$

where η_B is the learning rate. Eventually the reactive functions, B^{f_i}, will learn what sensory conditions the invocation of skill Q^{f_i} will be beneficial. Furthermore, the B^{f_i} functions learn which sensory signals that are relevant and which are irrelevant and can be ignored. When B^{f_i} is included in the top-down action selection mechanism of Equation 8, it will favor the selection of proven opportunistic and reactive skills above others, even those actions that it has learned to select. In the current example, if the recharging boxes are marked by a red light, it will be learned that the presence of a red light signal will trigger the recharge skill while all other extraneous sensory signals (such as spatial location and floor color) will be ignored.

3 Simulation

To evaluate nested Q-learning the simple two dimensional animat and world of Figure 2 was used. The animat's primitive actions were capable of moving it to one of four adjacent spatial locations. The animat's sensors perceived the color of the floor panel below it, the color of an overhead signalling light and the animat's spatial location within the world. The world is shown in Figure 2(c). It had white colored floor panels except for one blue and one green floor panel at the locations indicated. It was possible for the location of these blue and green panels to change, moving to locations indicated by either $World_1$ or $World_2$ in Figure 2(c). The animat remained unaware of these changes as it could only sense what was happening at its current spatial location within the world. Tasks for the animat were externally specified using reinforcement signals and could entail anything from movement to a desired spatial location to matching sensed floor panel color with the color of the signalling light. Summarized in Figure 3 are the features found to be relevant by the animat. Although these features were subject to random discovery, they are presented in an orderly list for the readers benefit in the following analysis. Shown in Figure 4 are the sensory, motor and reinforcement connections for the animat. The incoming sensors are the light color, S_{light}, the floor color, S_{floor}, and the spatial location, $S_{spatial}$. The outgoing actuator commands are up, Q^0, down, Q^1, left, Q^2, right, Q^3 and stay, Q^4. The incoming reinforcement signal is r_{EXT}. The initial control system is shown as a random structure of unknown skills, $Q^?$. During these tests the learning rates, η_Q, η_B, and the temporal discount factor, γ, were set to 0.2, 0.01 and 0.9, respectively.

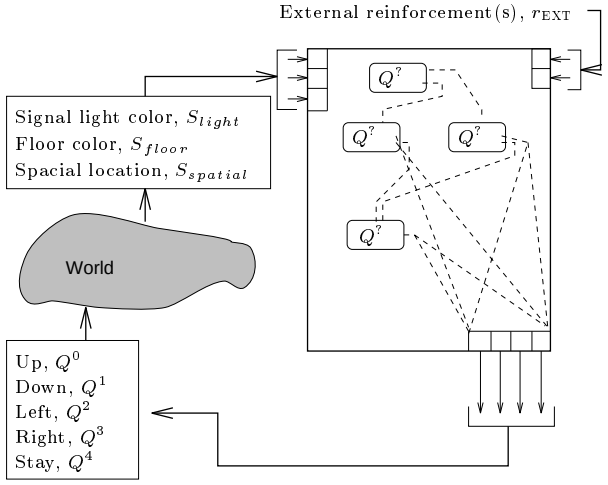


Figure 4: Connections of sensors, actuator and reinforcement signals and initial random structure of control system. For clarity the sensor connection are not shown.

3.1 Generation of Control Hierarchies

To evaluate the capabilities of nested Q-learning to generate control hierarchies the agent was rewarded with the external reinforcement signal of Equation 16.

$$r_{\text{EXT}} = \begin{cases} +2 & \text{if light is blue and floor is blue.} \\ +4 & \text{if light is green and floor is green.} \\ -1 & \text{otherwise.} \end{cases} \tag{16}$$

The locations of the blue and green floor panels were set randomly switching between the two possible configurations as shown in in Figure 2 (c). The signal light was set alternating between blue and green. An animat with no prespecified information was placed in this

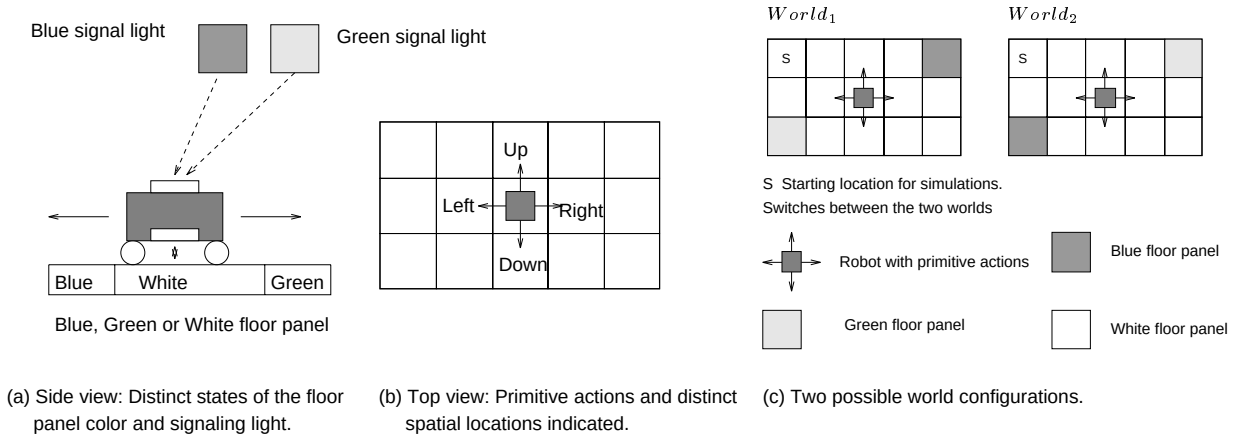


Figure 2: Simulated animat's sensory systems: (a) Floor color and signal light sensors, (b) distinct spatial locations and possible movements of the animat. Simulated world: (c) Blue and green floor panels in two possible configurations as indicated. Locations of the blue and green floor panels can change to that of $World_1$ or that of $World_2$.

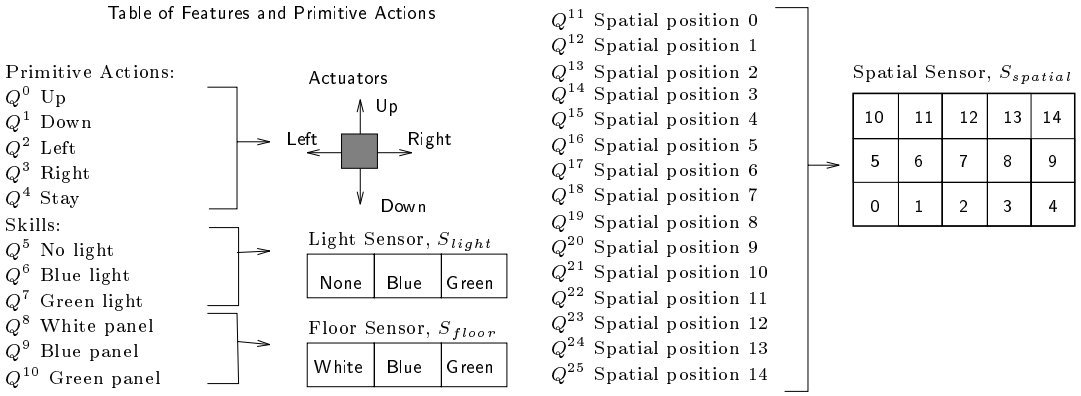


Figure 3: Summary of skills and features.

world and allowed to learn how to maximize its rewards over time. Performance plots for all skills and primitive actions versus time are presented in Figure 5. The vertical axis shows the performance of each skill or primitive action. Performance is taken to be the total reinforcement signal that the skill responds with whenever it is invoked. The axis pointing out of the page is the number of the skill or action and the horizontal axis represents the expired time. For clarity, selected skills have been extracted from the plot in Figure 5 and shown individually in Figure 6. From these figures it is seen that the primitive actions responded consistently with a performance of -1.0 (as they should), while all the adaptive skills performed changed over time and usually improved (if indeed they could). The first skills mastered were the ones defined by spatial locations, for example skills Q^{11} and Q^{25}. The two skills which proved to be higher level skills, Q^9, *find blue panel* and Q^{10}, *find green panel* were subsequently mastered using the two skills of Q^{11} and Q^{25}.

Sensations or features over which the animat had no control were never learned (nor could they be). For instance, Q^5, *find no signal light* could never be learned, as the signal light was always either blue or green and was never off. Interestingly enough, it was originally expected that a similar situation would occur for the other two signal light related behaviors, *find green signal* while the blue light was on and *find blue signal* while the green light was on, because the lights were not directly controlled by the agent. However, the light was alternating blue, then green, then blue, etc., changing upon task completion. The animat soon discovered that if it wanted to see a green signal light, all it had to do was go to the blue panel and once the task triggered by the blue light was completed, the green light would come on in place of the blue light. Instances of the agent finding novel solutions and taking advantage of unintentional loopholes in the simulation were common during these trials.

The bottom-up responses for all the possible skills are shown in Figure 7. Figure 7(a) shows the bottom-up response for an invoking state in which the blue signal light is on. Other details of that state were discovered by the control system to be irrelevant, as would be expected. Figure 7(b) shows the bottom-up response for an invoking state in which the green signal light was on. As indicated, the predominant bottom-up response for the green signal corresponded to Q^{10}, or *find the green panel*

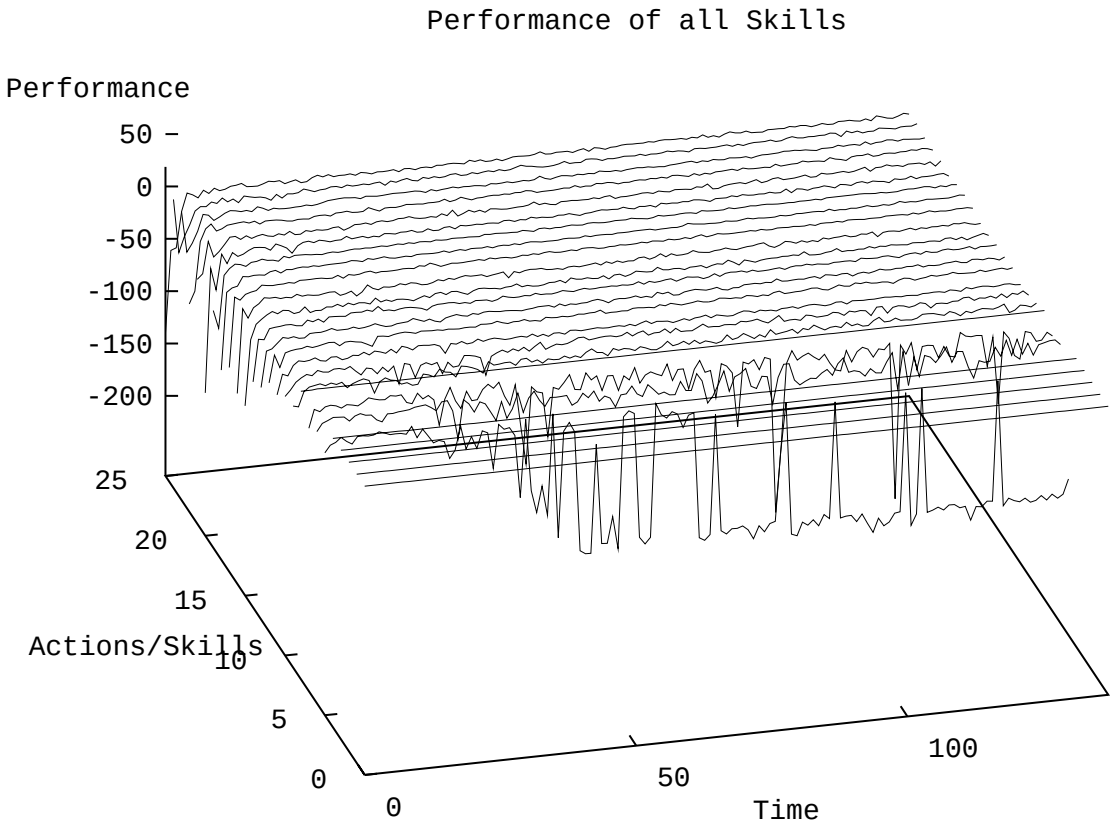
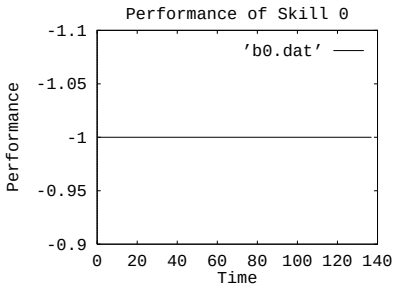
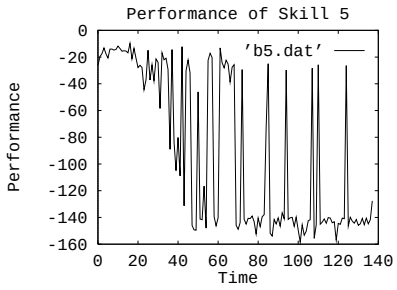


Figure 5: The performance of all skills.

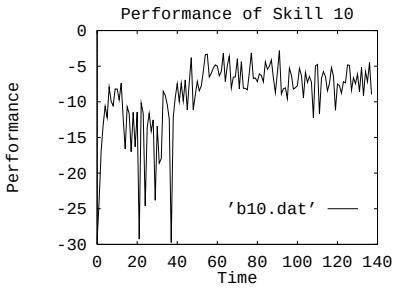


(a) Peformance for action Q^0, (up).

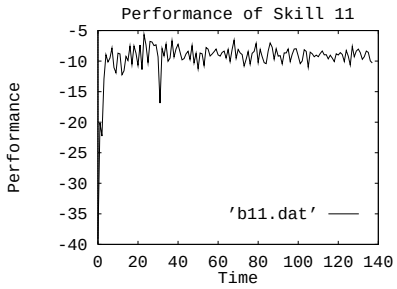


(b) Peformance of skill Q^5, (no light).

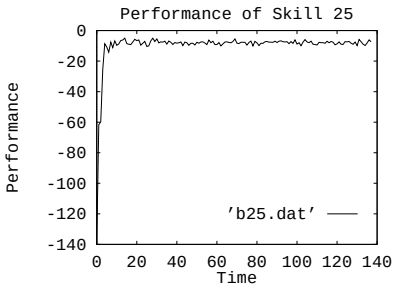
(c) Peformance of skill Q^9, (blue panel).



(d) Peformance of skill Q^{10}, (green panel).



(e) Peformance of skill Q^{11}, (spatial 0).



(f) Peformance of skill Q^{25}, (spatial 14).

Figure 6: Detailed performance plots for selected skills.

and for the blue signal it corresponded to Q^9, or *find the blue panel*. These bottom-up responses were shown to respond high for the skill that was capable of fulfilling the task that would result in favorable reinforcement signals. In these simulations, the green light triggered skill Q^{10} and the blue light triggered Q^9. These high level skill then invoked other skills and eventually primitive actions in a top-down manner to fulfil the tasks.

The control strategies invoked by the bottom-up reactions of Q^9 and Q^{10} are shown in Table 1 and Table 2. The schematic shown in Figure 8(a) shows the distinct two level architecture that emerged from the simulation. In Tables 1 and 2, the starting location is taken to be the upper left corner of the world at spatial position 10 as indicated in Figure 2(c). Tables 1 and 2 correspond to the configuration of the world as $World_1$. When the world is configured as $World_2$, Q^9 and Q^{10} are represented by Tables 3 and 4, respectively. Figure 8(b) shows the movements of the animat for the skill, Q^9 for both world configurations. It is clear that a hierarchical control system emerged which, when invoked by a bottom-up opportunistic drive, began to search for the locations of the correctly colored panels. It should be noted that as this search required two way movement through the state space and the continual disruption of a single monolithic evaluation function, learning this problem in a non-hierarchical architecture would be difficult. That is, for the current task, some states in a monolithic evaluation function would have a constantly changing correct action. Such a moving target would make convergence impossible. By decomposing the task into a hierarchical assembly of skills, the evaluation functions that represented the skills had a constant target to converge to.

Table 1: Skill Q^{10}: Find green floor panel (for $World_1$ with starting point as indicated in Figure 2(c)). Note: the length of the arrows indicates the depth of the command signal into the hierarchy. For example, $->$ indicates that the command signal is at the top of the hierarchy, while $--->$ indicates that the control signal is down two levels into the hierarchy.

Level Down	Command	Comments
1	$-> Q^{10}$	Find green panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^{1}$	Down
3	$---> Q^{1}$	Down and at goal

3.2 Generation of Novel Hierarchies

Often as the structure emerged without direct input from the designer, many unexpected but successful structures emerged. For example, in a related simulation, the agent was required to learn routes to spatial positions 4 and 14, represented by skills Q^{15} and Q^{25}, respectively. For some reason, skill Q^{20}, the skill to spatial position 9, which lies directly between positions 4 and 14, converged faster that the other skills. Skills Q^{15} and Q^{25} then learned to use Q^{20} to arrive close to their goal and then

Table 2: Skill Q^9: Find blue floor panel (for $World_1$ and starting point as indicated in Figure 2(c)).

Level Down	Command	Comments
1	$-> Q^{9}$	Find blue panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^{1}$	Down
3	$---> Q^{1}$	Down
2	$--> Q^{25}$	Spatial location 14
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{0}$	Up
3	$---> Q^{0}$	Up and at goal

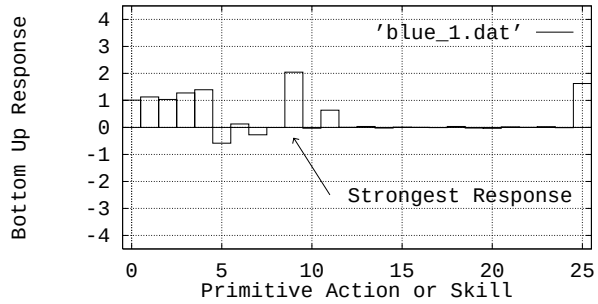
Table 3: Skill Q^{10}: Find green floor panel (for $World_2$ and starting point as indicated in Figure 2(c)).

Level Down	Command	Comments
1	$-> Q^{10}$	Find green panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^{1}$	Down
3	$---> Q^{1}$	Down
2	$--> Q^{25}$	Spatial location 14
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{0}$	Up
3	$---> Q^{0}$	Up and at goal

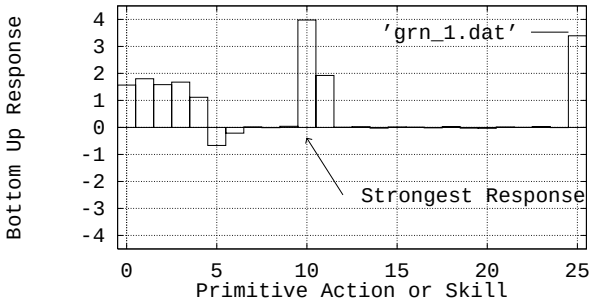
it was a single step to reach the desired location. This hierarchical control structure is illustrated in Figure 9(a) with the animats actions in Figure 9(b). Throughout the course of these simulations many other examples of interesting multi-level hierarchies emerged, each with the agent developing some unique and innovative method of solving the control problem.

3.3 Information Transfer Between Tasks

In this simulation the world was again set to alternating between its two possible states. Initially, the desired task, as defined by Equation 17, was to find the blue floor panel whenever the blue light was on. A hierarchy using the two skills Q^{11} and Q^{25} emerged to ultimately perform Q^9 (find blue panel) as shown in Figure 10(a). Next, a new task was requested as per the external reinforcement signal of Equation 18. The new task was to find the green panel whenever the green light was on. With most of the control strategy already learned from the previous task, the animat now had to learn how to utilize these existing skills to perform the new task. This was accomplished by learning skill Q^{10} (find green panel), extending the hierarchy to that pictured in Figure 10(b).



(a) Bottom up reactive response for the blue signal light on. Note the strength of response for skill Q^9.



(b) Bottom up reactive response for the blue signal light on. Note the strength of response for skill Q^{10}.

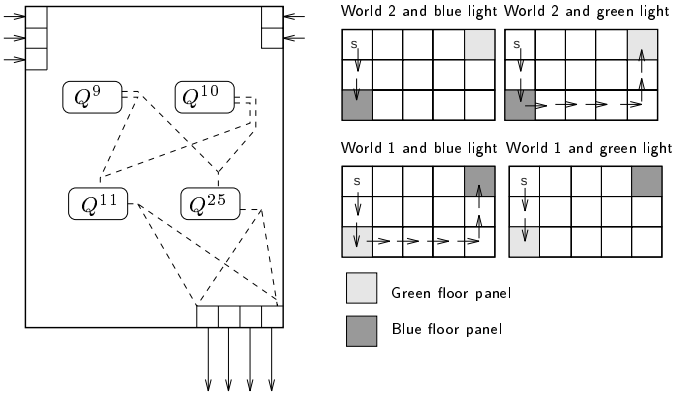
Figure 7: Bottom up reactive responses for all skills and primitive actions for sensory conditions of (a) blue signal light on and (b) green signal light on.

Table 4: Skill Q^9: Find blue floor panel (for $World_2$ and starting point as indicated in Figure 2(c)).

Level Down	Command	Comments
1	$-> Q^9$	Find blue panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^1$	Down
3	$---> Q^1$	Down and at goal

$$r_{EXT} = \begin{cases} +4 & \text{if light is blue and floor is blue.} \\ -1 & \text{otherwise.} \end{cases} \quad (17)$$

$$r_{EXT} = \begin{cases} +4 & \text{if light is blue and floor is blue.} \\ +4 & \text{if light is green and floor is green.} \\ -1 & \text{otherwise.} \end{cases} \quad (18)$$



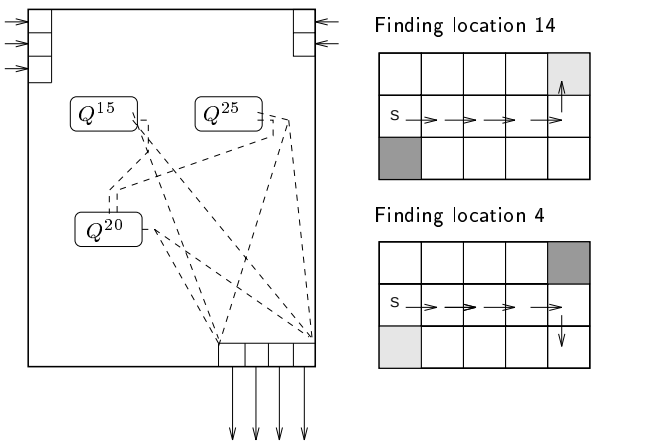
(a) Emergent structure. (b) Actions taken by the robot.

Figure 8: Generation of a hierarchical control system. (a) the structure that emerged with two bottom-up driven skills, Q^9 and Q^{10}, at the top, (b) actions taken by the animat.

4 Discussion

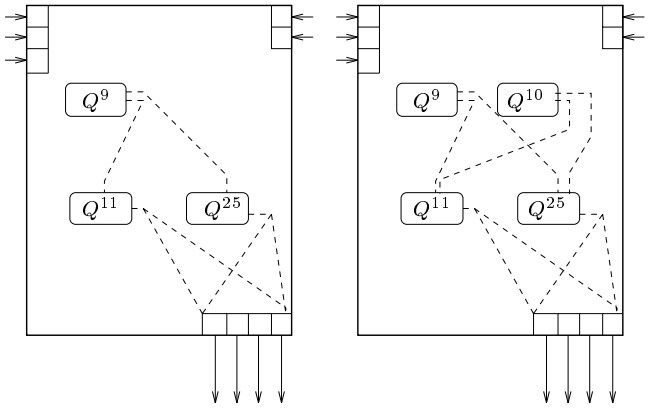
The hierarchical structures outlined emerged as sensory-motor relationships and converged from the bottom (closest to the actuators) upward. Those sensor-motor skills that required only primitive actuator movements were discovered and learned first. Higher level skills could then be built upon lower level skills. Once the low level spatial skills (Q^{11} and Q^{25}) converged, or at least had begun to be performed reliably, then the higher level skills (Q^9 and Q^{10}) could be learned. The learning of these stratified skills required both the learning of temporal sequences of primitive actions and/or skills as well as which sensor information was relevant or irrelevant at each emerging level of behavior. In these simulations the agent learned that the skills involved with finding spatial locations were independent of the color of the signal light and floor panels was irrelevant and also that the agent needed only primitive actions to perform reliably. This was not always necessarily true. Although only primitive actions were needed, this did not prevent the emergence of convenient hierarchical structures. An example of the learning system finding novel convenient methods of solving its problems was presented in Section 3.2 in which an intermediate location was used to find adjacent spatial locations.

Although initially necessary, the exploration component later contributes to poorer performance. Within such an evolving control system, it might be desired that skills crystallize, that is, limit and eventually cease the exploration component of the skill, once exploration begins to prove fruitless. This would result in skills becoming stratified from the actuators upward, freeing limited computation resources for discovering and learning yet higher skills. Should changes or malfunctions occur, the exploration component would be reactivated in the effected section of the control hierarchy. As the skills converged, many sensor inputs proved irrelevant to the evaluation and reactive functions and could be removed. For example, the skills necessary to find spatial locations were independent of signal light color and floor panel color and relied only upon a sense of location. Current



(a) Emergent structure. (b) Actions taken by the robot.

Figure 9: The hierarchical control system that emerged for skills, Q^{15} and Q^{25}, utilizing the intermediate skill Q^{20}. (a) the structure that emerged, (b) actions taken by the animat.



(a) First skills learned Q^9 Q^{11} and Q^{25}. (b) Second skill learned, Q^{10}.

Figure 10: Transfer of skills between tasks: (a) control structure after the first task, fulfilled by skill Q^9, is learned, and (b) after the second task, fulfilled by skill Q^{10}, is learned. Note: Q^{10} learns to use previously acquired skills.

work will allow the agent to prune sensory and action connections from skills when they are discovered irrelevant. Again, this would further crystallize the skills into encapsulated and efficient sensor-motor control systems. Another extension that becomes evident is that the abstraction of features must be allowed. Just as the control strategies are built from abstractions of primitive actions and lower skills, features must be allowed to construct hierarchies of abstraction.

5 Conclusions

The nested Q-learning technique developed in this paper generated hierarchical control systems for the control of a simple simulated animat. These structures often utilized unique and not obvious control strategies to solve the control problems confronting the animat. The section of the control structure that was active was determined by a combination of evolving bottom-up reactive drives and top-down goal seeking drives. As a bottom-up reactive/opportunistic drive was triggered, it was fulfilled by a cascade of top-down invoked skills. The nested Q-learning algorithm effectively partitioned the control problem into many small evaluations functions to be combined and recombined into different solutions rather than having solutions locked into a single monolithic evaluation function.

References

[1] Barto, A.G., Sutton, R.S. and Watkins C.H. (1989), Learning and Sequential Decision Making, *COINS Technical Report.*

[2] Digney B. L. (1994), A Distributed Adaptive Control System for a Quadruped Mobile Robot, *From animals to animats 3: SAB 94*, Brighton, UK, August 1994, pp. 344-354, MIT Press-Bradford Books, Massachussets.

[3] Albus, J.S. (1991), Outline of a Theory of Intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, 21, 3, pp. 473-509.

[4] Digney, B.L. (1996), Shaping Emergent Control Structures with Scaffolding Actions and Staged Learning, *Artificial Life V*, May, 1996, Nara-Ken, Japan. (to appear)

[5] Tyrrel, T., (1992), The Use of Hierarchies for Action Selection, *From animals to animats 2: SAB 92*, pp. 138-148, MIT Press-Bradford Books, Massachussets.

[6] Meyer, J.A. and Guillot, A. (1994), From SAB90 to SAB94, *From animals to animats 3: SAB 94*, Brighton UK, August 1994, pp. 2-11, MIT Press-Bradford Books, Massachussets.

[7] Maes, P. and Brooks, R.A., (1990), Learning to coordinate behaviors, *Eighth National Conference on Artificial Intelligence* pp. 796-802, 1990.

[8] Mahadevan, S. and Connell, J. (1991), Automatic programming of behavior based robots using reinforcement learning, *Nineth National Conference on Artificial Intelligence*, Anaheim, CA, 1991.

[9] Long-Ji, L. (1993), Hierarchical learning of robot skills, *IEEE International Conference on Neural Networks*, San Francisco, 1993, pp. 181-186.

[10] Dayan, P. and Hinton, G.E., Feudal reinforcement learning, *Advances in Neural Information Processing Systems 5*, San Mateco, CA, Morgan Kaufman, 1993.

[11] Singh, P.S., (1992), Transfer of learning by composing solutions of elemental sequential tasks, *Machine Learning, 8(3/4)*, pp. 323-339, 1992.

[12] Kaelbling, L.P., (1993), Hierarchical learning in stochastic domains: Preliminary results, *Tenth International Conference on Machine Learning*, Amherst, MA, 1993.

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