Learning to Integrate Reactive and Planning Behaviors for Construction

Gerald Chao, Anand Panangadan and Michael G. Dyer*
Computer Science Department
University of California, Los Angeles
Los Angeles, California 90095
{gerald, anand, dyer}@cs.ucla.edu

Abstract

A connectionist architecture that enables agents to navigate efficiently and learn to construct specified structures within an artificial environment is presented. The agent has both reactive behaviors, used to maintain the viability of the agent, and planning behaviors, which use cognitive maps to internally represent the environment for navigational planning. The action-selection module that mediates between the reactive and planning behaviors can be taught to perform high-level tasks such as construction. To do so the agent must manage multiple goals simultaneously and adapt to changes in the environment. Simulations within a continuous 2-D environment that demonstrate these behaviors and comparisons between two reinforcement learning rules (P/N Hebbian and Q-learning) are also presented.

1. Introduction

In this paper, autonomous agents that maintain viability while learning to construct and repair 2-D structures are presented. The agents adapt to changing situations by reconciling reactive behaviors, needed for survival, with planning behaviors, needed for construction, through action selection. Selecting the right action given its current goals is learned from a teacher and does not rely on *a priori* knowledge. The agents' goals consist of staying alive by attending to hunger, thirst and collision avoidance, while constructing and maintaining structures, such as straight walls and circles, built from materials gathered from a 2-D environment. By investigating the relationships between reactive and high-level behaviors, adaptive agents can be extended to learn and integrate multiple high-level goals without sacrificing the the low-level ones.

Construction is chosen as the high-level goal not only because of its potential applications but also for the interesting challenges it presents. Namely, a construction-agent researcher must address the following issues: 1) the survivability of the agent, 2) the integration of low-level and high-level goals 3) the internal representation of the world and the structures to be built, 4) the ability to internally monitor the construction status, 5) the efficiency of construction, 6) the ability to learn the high-level skills, and 7) the scalability of the system to add other high-level behaviors. Many of these issues, such as integration, representation, and scalability, are not specific to construction but also arise in models that contain both low-level and high-level behaviors. By studying construction, the lesson learned can potentially be applied to other high-level tasks.

In our model, these challenges are addressed by three main techniques: modularity, cognitive maps, and learning. Modularity supports scalability by minimizing interdependencies among behaviors and thus makes the incorporation of new behaviors easier. Cognitive maps are used to represent knowledge of the external world and the target structure, and are indispensable for navigational planning and monitoring of the construction progress. Finally, learning improves the adaptability of our model, enabling the agents to acquire the action sequences of a task based on changing needs. Additionally, the first two techniques resulted in an architecture that facilitates rapid and efficient learning, further improving the survivability of our agents. These techniques are implemented purely with artificial neural networks (ANNs), which offer numerous advantages such as smooth integration, ease of incremental learning, and biological plausibility. The resulting system is one that exhibits the complex behaviors consistent with the construction requirements described above.

2. The Agent and Its Environment

The simulation environment is similar to DiscoTech (Crabbe and Dyer, 1998). The agent is situated in a two dimensional and continuous world, with objects being discs of uniform radius. The discs represent entities that are relevant to the agent, such as food, water and building blocks, and are distinguished by their color: green discs are food, blue discs are water, and red discs are building blocks.

The agent perceives the world around it through 36 distance sensors for each color. These are evenly distributed all around it. Let each sensor i be aligned in a

*This work supported in part by an Intel University Research Program grant to the third author.

direction represented by unit vector $\vec{v_i}$. The sensors have a limited range of 20 units, covering only a small portion of the world. The activation of each sensor, $s_{i,color}$, is inversely proportional to the distance of the nearest disc in its field of vision. The agent also has a compass and this is used to align all sensor readings in one global direction. The agent in this world can continuously move forward or turn within a finite range, through motor commands that consist of the speed and the angle of a turn. However, the motors do not respond instantaneously to them; it takes time to accelerate and decelerate, and the agent can only turn through a small angle per time step.

To remain alive, the agent must eat and drink regularly. *Motivations* are indicators of the health of the agent and consist of *hunger* and *thirst*. When the internal food or water levels go below a threshold, these motivations are activated. To replenish its food and water levels, the agent must eat or drink by touching the appropriate disc. In addition, there is an *avoid* motivation that is always active so that the agent does not collide with objects. Construction in this world involves moving scattered building blocks into designated configurations, by "grabbing" discs if the agent is touching them, which can then "dropped" at another location.

3. Architecture

The architecture of the agent is composed of three modules: 1) Sensory/Motor, 2) Navigational Planning, and 3) Action Selection. These modules and their interconnections are shown in figure 1. The Sensory/Motor module maintains the overall health of the agent by reactively responding to objects within the sensor range. The Navigational Planning module builds and maintains cognitive maps of the environment, learned from the history of sensory inputs. These maps are then used for both construction and self-preservation goals, by planning trajectories to desired locations such as to food discs when the agent is hungry. Both the Sensory/Motor and Navigational Planning modules generate multiple motor actions as outputs, which are fed into the Action Selection module. The Action Selection module then chooses the appropriate action by prioritizing actions responsible for maintaining the agent's health over those required for construction. The Action Selection module is elastic and action priorities can be learnt though reinforcements. The reinforcement signals are produced by an external teacher, who is situated within the same environment and compares the student's external actions with its own.

3.1 Sensory/Motor

The architecture of the reactive module is shown in figure 2. *Behaviors* are motor primitives of the agent and each one takes the sensor data concurrently and outputs a motor activation $\vec{v} =< \theta, r >$, where θ is the angle of the turn and r is the speed of the motor action. For the behaviors MS^c (i.e., Max-Sensor-c), where c is one of the three colors, activation $\vec{v}_{MS^c} = \vec{v}(s_m)$, where $s_m = \max_i (s_{i,c})$. For instance, MS^B specifies the direction of the nearest blue disc (since activation $s_{i,B}$ is inversely proportional to the distance of the sensed object) and by selecting this motor action, the agent would exhibit the behavior of approaching the nearest blue disc. Behaviors SRS^c (i.e., Sum-Reverse-Sensor-c) generate motor activation $\vec{v}_{SRS^c} = -\sum s_{i,c} \cdot \vec{v}_i$. For instance, behavior SRS^B outputs a direction that would move the agent *away* from all the surrounding blue discs , equivalent to avoiding the blue discs. The agent also has *Eat* and *Drink* behaviors that are not shown in figure 2. The connections between motivations and behaviors are used to excite or inhibit behaviors. For instance, *Hunger* motivation excites the MS^G (approach green) behavior and inhibits the SRS^G (avoid green) behavior. This results in the agent approaching food only when it is hungry. Note that there are no second-order links to the MS^R (approach red) behavior. This behavior is needed during construction and hence is regulated by the Action Selection module (described later).

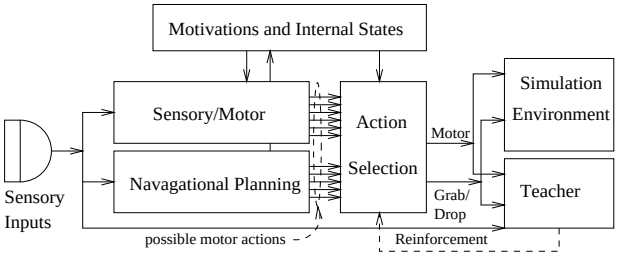

Figure 1: Block diagram of the architecture. The sensory inputs are processed by the Sensory/Motor and the Navigational Planning modules to produce all possible motor actions, one of which is chosen by the Action Selection module to send to the motors. The external action of the agent is then simulated, while the teacher generates reinforcements during training.

The design of the Sensory/Motor module adheres to the principles of constructing autonomous agents put forward by Rodney Brooks (Brooks, 1986). There are no long sequences of modules, with each taking output from the previous module and transforming it into another representation for the next module. Instead, each of the modules accesses the raw sensory data concurrently and produces a motor activation as output. This assures fast response times, which are important since the behaviors are responsible for keeping the agent alive. Currently, the behaviors are innate to the agent as they constitute the minimum required to ensure the survival of the agent. However, (Steels, 1997) shows how these mappings between the sensors and motor actions can be learned.

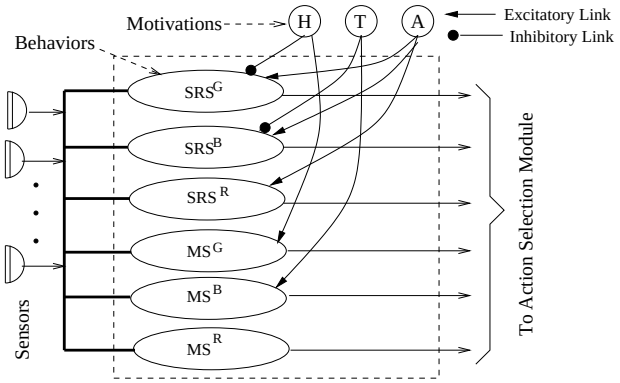

Figure 2: Architecture of the Sensory/Motor module. Sensory inputs are mapped to motor actions by the beahviors, performed using neural networks (shown as ovals). The behaviors are then regulated by the *Hunger* (H), *Thirst* (T) and *Avoid* (A) motivations through the second-order connections.

3.2 Navigational Planning

The Sensory/Motor module responds only to immediate sensory inputs and is therefore completely reactive. To plan a course, while not relying on direct cue stimuli, an internal representation of the world outside of the sensory range is needed. In our model, cognitive maps are used to represent the spatial relationship between the agent and objects in the environment. These maps are then used for planning trajectories to goal locations.

3.2.1 Egocentric Spatial Maps

Our model uses Egocentric Spatial Maps (ESMs) to represent cognitive maps (Chao and Dyer, 1999). ESM is a connectionist architecture that uses neurons arranged in an uniform grid to build and maintain egocentric spatial maps. To construct these cognitive maps, the agent needs to convert the distance and angle readings from the sensors into a spatial map representation. This is achieved by activating the neuron that is spatially correlated with a detected object, determined by intersecting an object's angular position with its distance, both provided by the sensors. The result is an internal representation of a birds-eye view of the world. To maintain these maps, new sensory inputs are continuously integrated to reflect changes in dynamic environments, such as the addition or disappearance of discs. As the agent moves, the neuron activations are passed to neighboring neurons to update the new positions of objects, based on dead-reckoning inputs. This process is illustrated in figure 3, where the agent needs to reach a goal location (star in figure 3) obscured by one of the two rectangles. At time t, the firing neurons (black circles) of the ESM reflect the positions of the rectangles. As the agent moves eastward, the rectangles' new locations are reflected on the ESM by passing neuron activations to the neighboring neurons immediately to the west. As the agent moves north, activations are passed to the neurons immediately south of the currently active neurons. To plan a trajectory, the neuron that corresponds to the goal location initiates a spreading activation on the *Navigation map*, shown at time $t+1$ on the lower right. Simultaneously, the ESM inhibits neurons on the Navigation map, forcing the activation to flow around obstacles. The gradient created by the spreading activation becomes the planned path, navigating the agent around the rectangle to the goal.

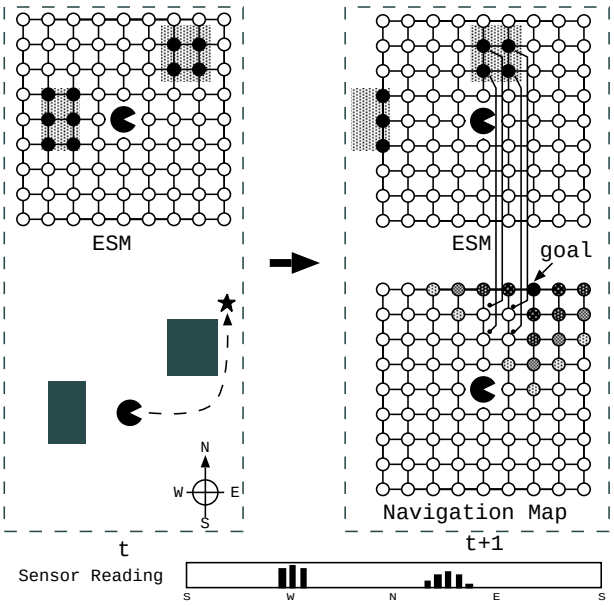

Figure 3: The updating of an ESM and path planning by spreading activation. Sensory reading at time t is shown below, and darkened circles represent firing neurons. As the agent moves to the right, the ESM activations are shifted to the left to reflect the change. The gradient created by the spreading activation on the navigation map, partially inhibited by the ESM, becomes the planned path.

3.2.2 Integrating Multiple ESMs

Using multiple ESMs, our agent separately stores the locations of the different kinds of discs encountered in the environment. The *Food ESM*, *Water ESM*, and the *Building Block ESM* store the locations of the food discs, water discs and building blocks, respectively. Each ESM is a uniform grid of 50 by 50 neurons, which cover a larger area than the sensors but not the entire world. In addition, the structure to be built is also represented in a ESM called the *Configuration ESM*. Like the other ESMs, the activations on this map are also shifted as the agent moves so that the egocentric nature of the map is preserved. Thus, this ESM maintains a "mental image" of the structure to build and repair. Unlike the other ESMs, however, the activations on the Configura-

tion ESM are set *a priori* and are not updated by the sensors.

Each ESM is associated with a Navigation Map that computes spreading activation to plan paths to a particular kind of disc (refer to figure 4), e.g., the Food ESM is associated with the *Food Navigation Map*. Each node in a ESM has an excitatory link to its corresponding node in its Navigation Map and thus provides the goals for path planning to the discs stored in that ESM. All other types of discs are viewed as obstacles and therefore inhibitory links are projected from those ESMs to the corresponding nodes on the Navigation Map. For example, each node in the Food ESM has an excitatory link to the corresponding node in the Food Navigation Map, which also receives inhibitory links from the Water and Building Block ESMs. This relationship can be expressed as the following: Let F_{ij}, W_{ij}, B_{ij}, and C_{ij} represent the activation of node ij on the Food, Water, Building Block, and Configuration ESMs, respectively. The initial activation of node ij on the Food Navigation Map, N^F_{ij}, is then

$$N^F_{ij} = F_{ij} - W_{ij} - B_{ij},$$

and is thresholded to be either 0 or 1. These nodes then initiate spreading activations, propagated throughout the nodes of the Food Navigation Map by:

$$N^F_{ij} \leftarrow \max_{i'j'}(N^F_{ij}, \eta \times N^F_{i'j'}) - W_{ij} - B_{ij}$$

where $i'j'$ are the the four neighboring nodes of node ij and η is the decay of the gradient, $0 < \eta < 1$. The inhibitory connections from the Water and Building Block ESMs prevent N^F_{ij} from firing, resulting in a path planned around obstacles. Paths to water discs are planned similarly by the Water Navigation Map.

The Configuration Navigation Map is used to compute a path to desired locations of building-block discs. Hence, it is activated by the Configuration ESM and inhibited by the activations from the the Food and Water ESMs. The Configuration Navigation Map is also inhibited by the Building Block ESM because if a building block is already present, the agent should not place another disc there. This is how the agent monitors the construction progress and detects missing parts of a structure. Let N^C_{ij} represent the activation of node ij on the Configuration Navigation Map, then the initial activation is

$$N^C_{ij} = C_{ij} - B_{ij} - F_{ij} - W_{ij}.$$

The same idea is extended to generate the activations on the Building Block Navigation Map, which is used to compute a path to a disc that is available for construction. It is excited by the nodes on the Building Block ESM and inhibited by the nodes on the Food and Water ESMs (obstacles during construction). However, there are inhibitory links from the Configuration ESM that deactivates firings from the Building Block ESM to distinguish available building blocks from ones that compose the structure already built. Directly inhibiting the nodes from the Configuration ESM is not desirable since it would always make the whole target structure an obstacle, regardless of whether discs are present or not.

With the initial activations set, the nodes on each Navigation Map spread activations to their neighbors until they reach the center of the map (where the agent is located, due to each map's egocentricity). The gradient of increasing activation from the center of the map (representing the current location of the agent) plots a path to the nearest goal (the activation with the fewest decays). Every Navigation Map outputs a motor action obtained from the gradient of activation at the center, as this indicates the direction in which the agent must move to reach that goal.

Additionally, simple computations of the activations on the Navigation Maps can provide useful information. For example, the sum of all nodes of the Construction Navigation map, denoted by SN^C, can be used to indicate if any part of the structure is missing discs. Similarly, the initial aggregate activity of the Building Block Navigation Map, SN^{BB}, can indicate if there are discs available for construction. The sum of activations from the two other maps can indicate the availability of food and water (but are not currently used). Also, the activations at the centers of the Construction and Building Block Navigation maps, CN^C and CN^{BB}, can indicate if the agent is at the construction goal location or at the available disc location. These four nodes are referred to as Internal State nodes and are used during construction.

Figure 4 illustrates the integration of the ESMs. The Configuration ESM has four activations in the shape of a square (representing the structure to be built). The Building Block ESM has three activations that align with three of the activations on the Configuration ESM. These represent building blocks that already form part of the structure. One of the two mismatches between the ESMs represent the location of a building block that is available for construction, while the other indicates the missing corner of the structure. These "misalignments" activate the corresponding nodes on the Construction Navigation and Building Block Navigation maps. Activations spread from these nodes to the center and give the direction the agent should move to reach the unbuilt part of the square (south-east) and to reach the building block available for construction (west). The activations from the Food and Water Navigation maps give the directions to reach the closest food (south-west) or water disc (north-west). The agent then has to properly choose the right action given the situation.

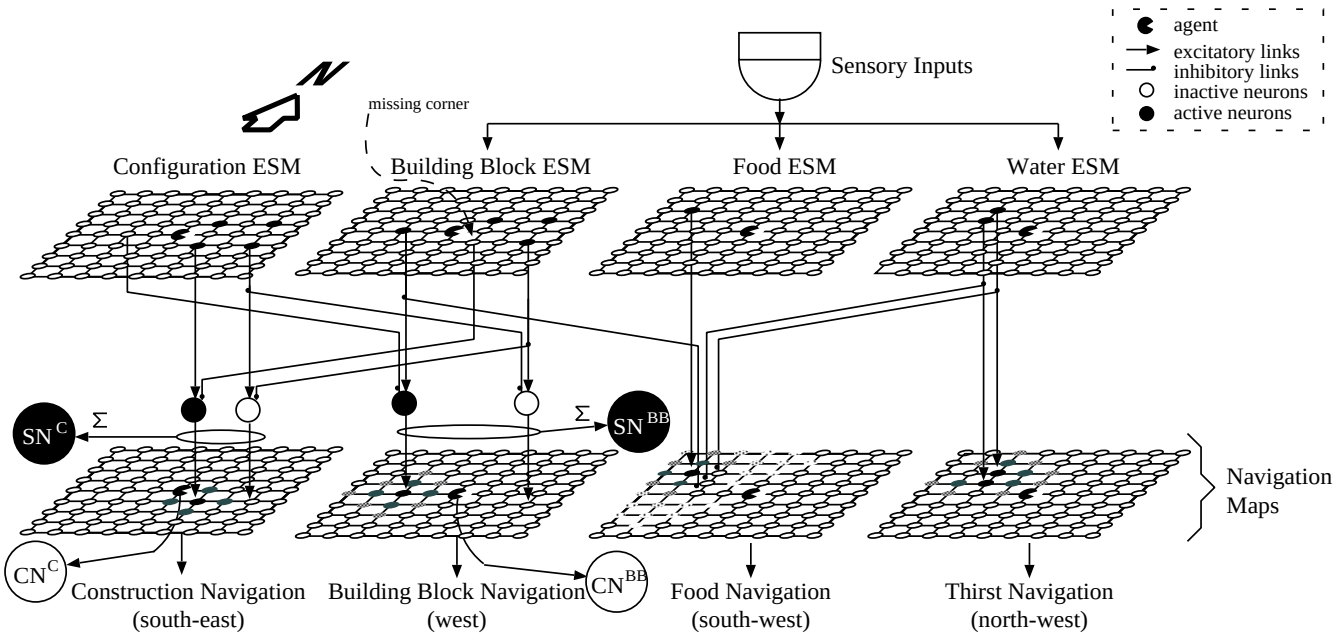

Figure 4: The four ESMs and their corresponding Navigational Maps that compose the Navigational Planning module. Only portions of the interconnections between ESMs are shown.

3.3 Action Selection

The Action Selection module chooses one of the motor actions from the Sensory/Motor and Navigational Planning modules to be sent to the motors, based on the actions' priority. Actions that are crucial to the survival of the agent, eating and drinking, have the highest priority, followed by avoiding obstacles, and finally approaching food and water. For example, if the agent becomes hungry during construction and food is visible, the MS^G (approach green) behavior will be selected. The prioritizing of these actions is implemented by lateral-inhibition links from the high priority actions to all of the lower ones, as shown in figure 5. Therefore, if a high priority action is active, all other lower priority actions are silenced. The lateral-inhibition connections are assumed to be innate and not modified, but the mechanism is general and can be used for any number of behaviors and with different orderings.

If none of the self-preservation goals are active, then the agent can attend to the construction task. This task requires an orchestration of both Sensory/Motor and Navigational Planning actions, summarized in the following action sequence:

⇒ Stay alive at all times

1. Determine if construction has been completed or if repair is needed.
2. Locate an available disc (not a part of the structure):
   - 2a) Choose MS^R (approach red) if close to an available disc.
   - 2b) Activate *Building Block Navigation* if available discs are not directly visible.
   - 2c) If there are no available discs, choose *Explore*.
3. *Grab* the disc when it is within reach.
4. Navigate to the location where the structure is missing a disc by selecting *Construction Navigation* action.
5. Once at the goal location, *Drop* the disc.
6. Go to Step 1.

Based on sensory inputs and Internal State nodes, this sequence can be performed by the Action Selection module through the connections and weights shown in figure 5. Each node in the action selection module may have both excitatory (arrows) and inhibitory (dots) connections from the Internal State nodes. Weights range from 0 to 1 for excitatory and 0 to -1 for inhibitory, and all activations are summed and thresholded, i.e.,

$$A_j = \psi[\sum_i (W_{ij}^{excite} \times A_i + W_{ij}^{inhibit} \times A_i), \theta],$$

where ψ is the step function with θ as the threshold, and A_j is the activation at node j.

The excitatory links activate a node when a set of conditions is met, such as enabling *Grab* when (a) the agent is at a disc ($CN^{BB} > 0$) and (b) the structure is missing discs ($SN^C > 0$). Therefore, 0.5 is assigned to both weights to ensure that *Grab* is active only when both input nodes are firing to surpass the threshold θ of 0.7. However, if the agent already has a disc (*Have-Disc* > 0), then the *Grab* node should be inhibited so the agent does

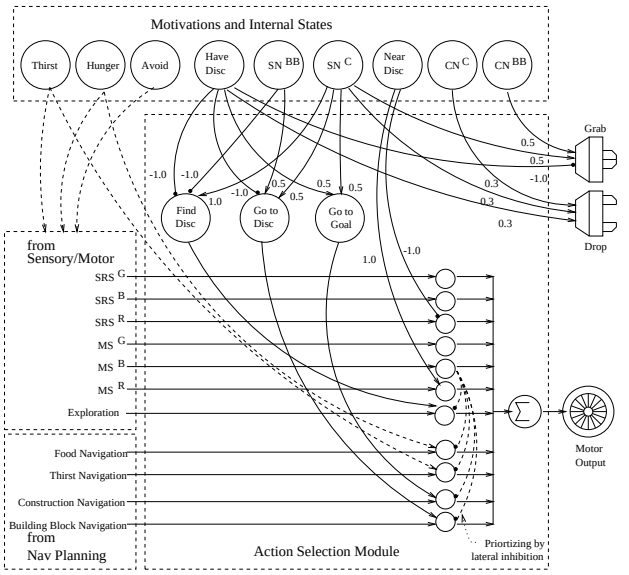

Figure 5: The Action Selection module with ideal connections and weights for construction. The outputs from the Sensory/Motor and Navigational Planning components are fed in from the left and regulated by the Motivation and Internal State nodes. The actions chosen are then sent to the motors on the right. Only one set of the lateral inhibition links for prioritizing is shown.

not try to grab another disc. This is achieved by the inhibitory weight of -1.0, more than sufficient to prevent the activation from surpassing the threshold. Since self-preservation goals have higher priority, the reactive response of avoiding the disc must be turned off when the agent needs to grab a red disc (achieved by the inhibitory link -1.0 from node *Near-Disc* to SRS^R (avoid red) action in figure 5).

By using this combination of excitatory and inhibitory connections, the Action Selection module selects the correct action based on its currently active goals and the construction status. By integrating the reactive motor responses with high-level navigational planning, this module is able to orchestrate the proper sequence of steps to perform construction while preserving viability. The issue of learning this construction skill through weight adjustment is first discussed, followed by the demonstration of this capability in the Results section.

4. Action-Selection Learning

Learning is used to improve the adaptability of our system by enabling the agents to acquire complex behaviors and not rely only on innate architectures. When placed in the environment, a student agent makes no assumption about the high-level goals and how to perform them. This approach provides the flexibility to specialize each agent to changing needs or procedures. For example, instead of construction, the same agent could perform disassembly or move red discs next to food discs simply by learning a different sequence of navigation and grab/drop actions. The ability to survive, however, is innate and not influenced by the teacher.

The teacher agent is situated in the same environment, and its actions are determined by the ideal Action Selection network shown in figure 5. It receives the same sensory inputs as the student, analogous to the teacher following immediately behind or riding on top of the student. The teacher then monitors the student's external actions and generates two positive or negative reinforcement signals, one for the arm action and the other for the motor action of the student. These signals are generated by comparing the student's external actions with the actions the teacher would have taken.

The student begins with a fully connected network, and since the links between any two nodes can be either excitatory or inhibitory, two weights have to be learned simultaneously. Through immediate rewards, the sequence of actions to carry out construction is learned by adjusting the weights to minimize error. The equation for weight adjustments is as follows:

$$W_{ij}^{t+1} = W_{ij}^{t} \times (1 + R_j^t \times W_{ij}^t \times A_i \times e^{-|W_{ij}^t|} \times \rho), \quad (1)$$

where W_{ij}^t is the weight between i and j at time t, A_i is the activation of node i, ρ is the learning rate, and R_j^t is the reinforcement signal at node j. The exponential term is used to restrict the weights to within 1 or -1. If a positive reinforcement of 1.0 is received, the excitatory weights between a pair of firing neurons at time t is strengthened, while the complementary inhibitory weights are weakened. However, a negative reinforcement of -1.0 causes the opposite adjustments to occur, reducing the likelihood of firing given the same inputs. Due to the nature of having both weights adjusted simultaneously, we refer to this learning rule as Positive/Negative (P/N) Hebbian learning. Given sufficient training, the student achieves connections similar to the one shown in figure 5.

To learn the action selection sequence for construction, Positive/Negative Hebbian learning is used on the two sets of weight matrices of 6 inputs (*Have-Disc*, SN^{BB}, SN^C, *Near-Disc*, CN^C, and CN^{BB}) by 7 outputs (*Grab*, *Drop*, *Go to Goal*, *Go to Disc*, *Find Disc*, SRS^R, and MS^R), representing the fully connected network of both excitatory and inhibitory links. The teacher network receives the same inputs as the learning agent and generates the two reinforcement signals.

If the student performs the correct actions, the weights that resulted in these actions are reinforced. If a wrong action is performed, one could simply weaken the weights that produced the erroneous activation. However, this form of "blame assignment" is flawed since the agent could potentially never learn because the correction action is never activated and thus never positively reinforced. The correct node that is supposed to fire should

also be trained.

However, the reinforcement signal does not identify the correct node. Instead of having the student agents randomly guess the correct action, all actions except the one just executed are treated as correct (even though only one is correct). As a result, the incorrect node receives a negative reinforcement while all others receive a positive one. With sufficient training, the correct node will prevail while all the incorrect nodes cease to activate under the same inputs. Through P/N Hebbian learning, the student's weights are adjusted to activate the right nodes in the right sequence, thus resulting in the student performing the construction task as specified by the teacher.

5. Results and Discussion

Both the teacher and the student agent, once trained, are able to construct structures within dynamic environments. In figure 6, screen shots of a sample construction run are shown. This initial world consists of a wall made of food and water discs that acts as an obstruction, while the building block discs are scattered around the environment. The agent's Configuration ESM has node activations set to represent the structure to build, in this case a square consisting of four widely spaced blocks (shown on the lower right map in part E). In part A (and D), a portion of the simulation program is shown, displaying the motivations of the agent and other status on the left side. On the right side of part A, the agent has located the first disc (path in lighter grey) and moved it to the closest corner of the structure (path in darker grey). After properly relocating another disc and en route to place the third, the agent became thirsty and thus suspended construction to reach the water disc (part B). Resuming the high-level task, the agent grabbed the fourth disc on the other side of the wall, when the status of the four ESMs (part E) and the Construction Navigation map (part F) was captured. The ESMs show the agent's belief about disc locations, not completely accurate because the sensors are unreliable at long range. The navigation map is used to plan a path, showing the spreading activation starting at the the last missing corner and flowing around obstacles, which includes the three corners of the square. In part D, the agent's ability to repair and adapt to changes is demonstrated. After the agent successfully constructed the square, the right corner of the square and the middle of the wall (of food and water discs) were destroyed. The agent recognized the need for repair and initially navigated around the left side of the wall to reach the fifth disc because the gap was outside of the sensor range. However, when the agent did detect the gap, it adapted to the new environment and returned to the damaged corner via the shorter route.

Learning to construct a configuration of discs was evaluated by having one student agent trained within the same training scenario, where 5 discs were used to construct a structure. The student repeated the same scenario until it no longer made any mistakes, at which point it was placed into a novel environment to validate that the learning is generalized and adaptive to other construction scenarios. The student agent was given no other *a priori* knowledge except for the structure to be constructed, i.e., the activations on the Configuration ESM. All other ESMs are initially empty and the Action Selection network has all of its weights randomly initialized to be between 0 and 0.5 (excitatory) or 0 and -0.5 (inhibitory).

The two learning parameters, learning rate ρ and decay η of the learning rate between trials, were varied to test their effects. The results from two sets of parameters are shown in figure 7, as well as the results from using Q-learning (Sutton and Barto, 1998) to learn and perform the same task as the Action Selection module. The number of errors (mismatch between the teacher and the student) during training by the student agent versus the trial number is shown, with the error bars showing the standard deviation from 10 independent runs. Each trial consists of 1000 time steps, and between each trial the learning rate ρ is reduced by the decay η for P/N Hebbian learning.

The Q-learning used in this evaluation learns two action-value functions, one for Grab/Drop actions and the other for motor actions. Instead of using function approximators, the two Q-functions are implemented using tables since the state space is small (2^6). The same training and evaluation scenario, teacher-agent, and reinforcement signals are used for Q-learning, repeating the learning trials until no mistakes are made. The ϵ-greedy parameter is halved between trials to improve the rate of convergence.

As shown in figure 7, P/N Hebbian learning makes fewer mistakes and converges faster than Q-learning. For P/N Hebbian, the higher learning and decay rates produced faster learning and convergence. In 7 out of 10 runs the student learned the task in 4 trials or less. With a lower learning and decay rate, the learning is slower and takes longer to converge, as expected. For Q-learning, the student agent makes more mistakes and requires more time to converge. The parameters for Q-learning were chosen to achieve a balance between the number of errors and the convergence rate.

Another benefit of P/N Hebbian learning is that the weight matrices can be easily interpreted for the action sequence learned. As an example, learning of the internal node *Go to Disc* is shown in figure 8. An agent should activate *Go to Disc* when there are discs missing from the structure (SN^C) and there are discs available within the environment (SN^{BB}). However, if the agent already has a disc (*Have-Disc*), this node should be inhibited and the agent should proceed to the location where a disc is

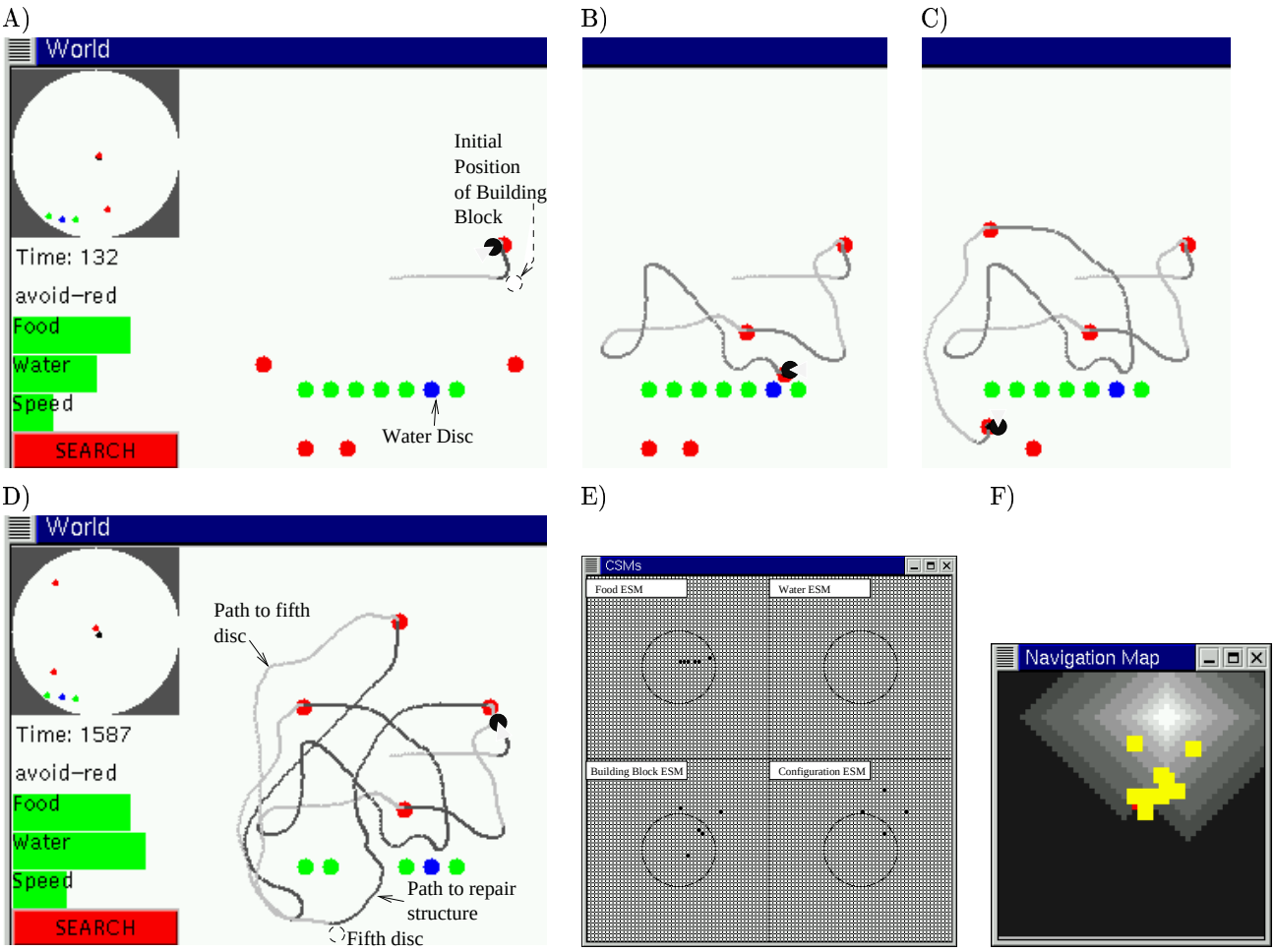

Figure 6: Sample screen shots of a construction run, with part A-D showing different stages of the construction and part E and F showing the ESMs and Navigational Map captured at stage C. The circles in part E indicate the limit of the agent's sensory range. Please refer to the text for more details.

missing from the structure (*Go to Goal*). These three pairs of weights are plotted over time using thick lines in figure 8. For the input SN^C, the excitatory weight is strengthened while the inhibitory one approaches zero, as expected. Similarly, the inhibitory weight of *Have-Disc* increases and the excitatory one decreases with time, also expected. However, we had not anticipated that the weights for CN^C (at goal) and SN^{BB} would be similar. This is due to the following scenario: the *Go to Disc* node is activated immediately after the agent drops off a disc so it would continue the building process. However, the CN^C node is still active from the *last* goal since it has not yet moved away far enough to stop the CN^C node from firing. Therefore, the weight between CN^C and *Go to Disc* tends to be strengthened alongside the weight for SN^{BB}. Nevertheless, the student does learn to properly activate the *Go to Disc* node, even though it is different from the human-engineered teacher network.

One thing to note about P/N Hebbian learning is that its complexity is linear for space and time (for each update), both requiring $O(n_i \times n_o)$, where n_i and n_o are the number of inputs and outputs, respectively. However, Q-learning, when implemented using tables as in our model, is exponential with respect to n_o for time per update and n_i and n_o for space. This is significant for scalability, since if the agent is to learn other complex tasks that require more input and output nodes, using Q-learning with tables can become too cost prohibitive. However, if function approximators are used in place of tables, we believe that they would not perform nearly as well since the Q-functions used for the construction task consist of multiple sharp discontinuities and are thus difficult to learn and approximate accurately.

6. Conclusions and Future Work

The connectionist architecture described here has several features that make it suitable for an environment where agents have to survive autonomously and adapt to changing situations. By using the motivations to gate

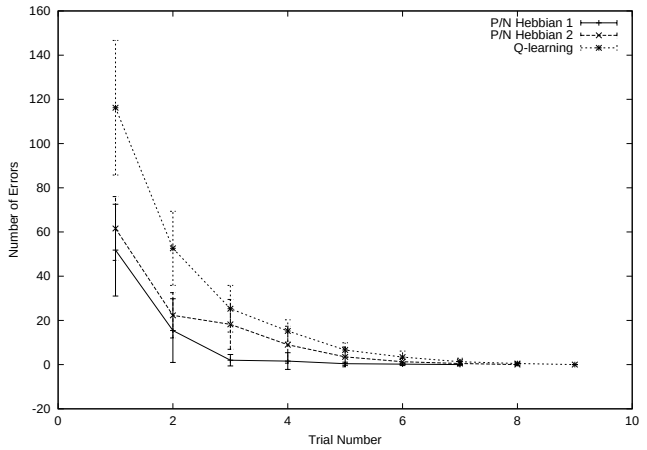

Figure 7: The number of errors made by the student agent versus the learning trial using two different learning algorithms. The two P/N Hebbian learning (parameters: ρ_1=0.8, η_1=0.4, ρ_2=0.6, η_2=0.8) results are compared to Q-learning (parameters: α=0.3, γ=0.5, ϵ=0.05).

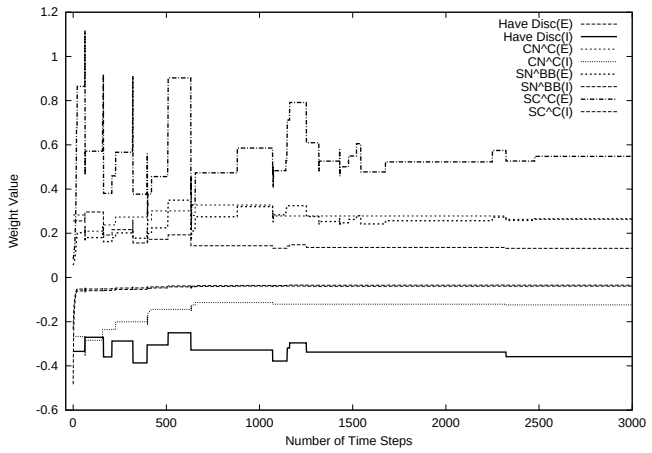

Figure 8: P/N Hebbian learning of the weight pairs between four of the input nodes to the *Go to Disc* node versus the number of time steps.

the behaviors, the agent is able to handle more than one low-level goal simultaneously. Moreover, by selecting the behavior with the highest activation, actions that are most likely to succeed are chosen over others (the activation of a behavior is proportional to how strongly the sensors are activated, which in turn depends on how close the disks are to the agent), i.e, consummatory actions are chosen over appetitive actions, one of the requirements for an action-selection mechanism specified in (Tyrrell, 1993). Though the agent can have any number of behaviors, the number of levels of processing between sensors and motors remains the same. Thus, this approach scales well and the time taken for the agent to react to the sensors remains small.

The Action Selection module chooses the actions specified by the reactive behaviors if they are active over that of the Navigational Planning module. This is a simple yet effective way of integrating low-level behaviors with higher ones. However, high-level tasks can also override these reactive behaviors such as when the agent is close to an available building block (red disc), MS^R (approach red) action is chosen while SRS^R (avoid red) is inhibited.

The spatial maps enable the agent to construct a representation of the environment and use it for path planning. Localization (locating oneself on the map) is easy on a ESM since the center of the map always corresponds to the agent's current location. This is particularly important for construction tasks, since the agent has to match the given target pattern (the Configuration ESM) to the actual locations of the building blocks (the Building Block ESM) to identify what discs are not part of the structure and what parts of the structure are yet to be built. Since the maps are always aligned due to their egocentricity, the aforementioned matching can be done easily.

The Configuration ESM can represent any 2-D structure of arbitrary size and shape, given sufficient neural resources, and can be easily modified at any time since the structure is represented by neuron activations and not fixed weights. Therefore, it is possible for another process to dynamically regulate the Configuration ESM. For instance, to construct structures that have to be built in a particular order (such as concentric circles), the nodes on the Configuration ESM that represent the inner circle can first be activated and only after this is built will the outer circle nodes be activated. The dynamics of the Configuration ESM become even more important when the system is scaled to 3-D environments, where the order of construction is crucial. By adding a new module that regulates the ordering of construction, the existing architecture can scale easily to the third dimension by transforming the ESMs into cubes.

Some of the possible weaknesses of this model can be addressed as follows: The system appears to require separate maps for each disc color. However, neural networks that are trained to classify sensory inputs into one of the n ESM classes can be added. For example, instead of having the disc color directly determine its function, neural network classifiers can learn the mapping from the colors to either food, water or building blocks. This addition would enable the agent to adapt to more complex environments where discs are of arbitrary colors and sizes. Also, structures can then be constructed with more than one disc type (red). Also, the ESMs currently depend on accurate dead-reckoning inputs for updates. In a real world, this information cannot be always reliably obtained from the motors and will introduce inaccuracies in the map. However, spatial maps can themselves be used to help calibrate the motors to improve dead-reckoning accuracy (Yamauchi et al., 1999).

With only limited sensors and minimal prior knowledge, the agent presented in this paper learns to integrate the reactive with the planning modules to exhibit the high-level behavior of construction and repair, while staying viable at all times. The learning by the P/N Hebbian rule is not only fast and accurate, it is also efficient and scalable to integrate other complex behaviors. The modularity of our architecture also enables additions of behaviors without affecting the pre-existing ones. The resulting agent is one that constantly maintains viability, efficiently learns and performs high-level tasks like construction, and adapts to changing needs and dynamic environments.

7. Related Work

Several architectures that maintain spatial maps to solve navigational tasks have been proposed. (Trullier et al., 1997) provide an excellent overview of such biologically based artificial navigation systems. The ESMs used in our system are similar in principle to "evidence grids" developed in (Moravec and Martin, 1996), which were tested on physical robots.

(Mataric, 1992) shows how spatial representation can be integrated into a behavior-based architecture (Brooks, 1986). The internal representation is a graph where nodes represent corridors and walls, and the agent finds a path to the goal by spreading activation from the node representing the goal. Since the maps store only the locations of walls, this system cannot be extended to the construction task. The PerAc (Perception-Action) architecture presented in (Gaussier and Zrehen, 1995) allows learning of sensory-motor associations. The reflex system is innate as in our architecture. In (Revel et al., 1998), a spatial map is added to the PerAc architecture. This spatial map is again in the form of a graph where each node represents a region and the arcs denote the topological relationship between regions. Though this architecture can handle more than one goal, there is no mechanism by which different goals are presented to the system dependent upon the varying requirements of a high-level task like construction. In (Crabbe and Dyer, 1999), autonomous agents perform a construction task without the help of an internal spatial representation. This is possible because the sensors have an infinite range and discs are colored to mark crucial locations.

Unlike the distributed, non-hierarchical action-selection mechanism presented in (Maes, 1990), this model has no links distributing activation between modules. The behaviors influence each other only by affecting the common environment. This is similar to the approach of behavior-based robotics (Arkin, 1998). The architecture presented here shows how spatial maps can be integrated into the action-selection mechanism to adjust to different tasks.

References

Arkin, R. (1998). *Behavior-based Robotics*. MIT Press.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23.

Chao, G. and Dyer, M. G. (1999). Concentric spatial maps for neural network based navigation. In *Proceedings of the Ninth ICANN*, pages 144–149, University of Edinburgh, U.K.

Crabbe, F. L. and Dyer, M. G. (1998). Goal sequence achievement using higher-order neural connections in construction agents. Technical Report UCLA-TR#980018, UCLA.

Crabbe, F. L. and Dyer, M. G. (1999). Second-order networks for wall-building agents. In *Proceedings of IJCNN'99*, Washington D.C.

Gaussier, P. and Zrehen, S. (1995). PerAc: A neural architecture to control artificial animals. *Robotics and Autonomous Systems*, 16:291–230.

Maes, P. (1990). Situated agents can have goals. *Robotics and Autonomous Systems*, 6:49–70.

Mataric, M. J. (1992). Integration of representation into goal-driven behavior-based robots. *IEEE Trans. on Robotics and Automation*, 8:333–344.

Moravec, H. and Martin, M. C. (1996). Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University.

Revel, A., Gaussier, P., Lepretre, S., and Banquet, J. P. (1998). Planification versus sensory-motor conditioning: what are the issues? In *From Animals to Animats 5*.

Steels, L. (1997). A selectionist mechanism for autonomous behavior acquisition. *Robotics and Autonomous Systems*, 20:117–132.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning*. The MIT Press, Cambridge, Massachusetts.

Trullier, O., Wiener, S., Berthoz, A., and Meyer, J.-A. (1997). Biologically based artificial navigation systems: review and prospects. *Progress in Neurobiology*, 51:483–544.

Tyrrell, T. (1993). *Computational Mechanisms for Action Selection*. PhD thesis, University of Edinburgh.

Yamauchi, B., Schultz, A., and Adams, W. (1999). Integrating exploration and localization for mobile robots. *Adaptive Behavior*, 7(2).