Goal Sequence Achievement Using Higher-Order Neural Connections in Construction Agents

Frederick L. Crabbe[⋆]**, Michael G. Dyer**[⋆⋆]

[⋆] Artificial Intelligence Laboratory
Computer Science Department
UCLA, Los Angeles, CA 90095
cardo@cs.ucla.edu
[⋆⋆] dyer@cs.ucla.edu

Abstract

This paper describes a neural architecture for selecting and achieving sequences of goals. The architecture is tested on a wall-construction task in a two dimensional environment containing circular objects. Scavenger agents bring building materials to a site while builder agents use the material to construct a wall. The agents are robust, that is, they stop to eat and drink when hungry or thirsty and they are able to recover from failed conditions and incomplete steps.

1. Problem

We are developing agents that can build structures out of building materials found in a two dimensional environment consisting of circular objects called discs. Primarily we focus on nests, which are structures that can protect the agents from predators or other dangers like weather. Nests include briar patches and enclosures. Briar patches are collections of material packed close together, preventing passage of predators but not close enough to prevent passage of the agents, and enclosures are surrounding walls with one or more openings. This paper focuses on a part of the enclosure construction task, namely, building a wall. The agents, controlled by artificial neural networks, must complete the task while performing other tasks necessary to their survival. As the agents build the wall, they must still be able satisfy their more immediate goals, such as eating, or fleeing predators.

1.1 Task

A first step in creating agents that can build an enclosure is to have them build a wall. In our world, we define a wall as any set of contiguous discs stretching out in a general direction. A wall does not need to be perfectly straight or have uniform thickness. There are many possible classes of algorithms a set of agents could use to organize discs to form a wall, each requiring different abilities in the agents. For instance, one class of algorithms depends on blueprints. Building a wall using a blueprint would require an agent to have a world map, as well as the ability to decode the wall encoded in the blueprint. Another class is based on leaving markers in the environment. Marker-based algorithms include: the *one-end-out*, the *two-ends-in*, and the *fill-in*. In the one-end-out case, the agents build the wall from a point in the environment toward a marked point. The two-ends-in strategy starts with both ends of the wall marked and the agents add to those marks on either side, growing the wall toward the middle. The fill-in approach is like the two-ends-in, except the agents place the building material anywhere between the two end points.

The particular algorithm the agents in this paper use is the one-end-out. M agents construct a pile by scavenging for building materials while N additional agents build a wall starting from a single point. The wall is built from the starting point towards the pile of scavenged discs. The agents use the locations of specially colored discs, which are placed by the researchers[1], to locate the pile and the wall. Figure 1 shows how one-end-out works. The scavengers look for red-colored discs to use as building materials. Once a scavenger finds a red disc, it carries it to a single brown disc. The brown disc is used to mark the place where the scavengers are gathering the building materials. Once a scavenger has brought a red disc to the brown disc, it drops the red disc and colors it orange. The orange color signals the wall-building agents that the material is ready for use in construction. If the discs were left red, then the building agents would have to build with red discs, and would not use only the scavenged discs. If the red discs were colored brown like the pile marker, then the building agents would have to build with brown discs and might inadvertently pick-up the brown pile marker and use it to form part of the wall. When a wall-builder sees an orange disc, it picks it up and looks for a purple disc. A single purple disc is placed in the environment at the start of the wall. Each builder takes an orange disc to a purple disc and then colors the orange one purple. Coloring a disc purple keeps other builders from inadvertently

[1] We envision that in the future, other agents will set the locations of these initial discs.

disassembling parts of the wall while building the wall. The coloring of an orange disc to purple as the disc is added to the wall also gives other wall-building agents a new wall end toward which to carry orange discs. During this construction, the agents must still stay alive. In this paper these survival-based tasks are eating and drinking.

1.2 Challenges

In order to perform the one-end-out wall-construction task properly, it is important that our agents' neural control mechanisms exhibit the following:

- *Robust Sequencing*. An agent must be able to perform long sequences of actions, even in the absence of changes in the environment. For instance, when scavenging, an agent must look for and find an object to scavenge, go to it, grasp it, look for and find the location to take it to, drop it, and change its color.
- *Backtracking and Conflict Resolution*. An agent needs to detect problems in a sequence, including when the agent fails to complete some action or when some other prior state of a sequence becomes undone. When these problems are detected, the agent needs to return to an earlier step in the sequence to correct the problem. For example, if a scavenging agent is carrying a disc back to the scavenge-site when it stops to drink, it must drop the disc it is carrying in order to drink. The agent has to know that it's no longer carrying a disc and must go find another one before returning to the piling site.
- *Obstacle Avoidance*. Agents must be able to get to their destinations in an environment strewn with obstacles. They must avoid collisions with those obstacles as well as with other agents.

2. Experimental Environment

The experiments are run in DiscoTech, a continuous two-dimensional world typically 1000x1000 units in size. Everything in the environment (both objects and agents) are circular discs. Agents are 12 units in diameter. Objects in the environment range from 2 units in diameter to 72 units in diameter. Each disc is a single color: agents are black, while the passive discs consist of green-colored food discs, blue-colored water discs and building discs of a variety of other colors. Food and water discs can be consumed by the agents. Food discs disappear when eaten and water discs shrink as agents drink from them.

An agent can visually sense other discs in a 120 degree-wide field in front of it. There is a set of visual receptor nodes for each color in the environment, e.g. a set for blue (water) discs, a set for red discs and so on. Each set consists of three sensory nodes, one for the left portion of the visual field, one for the right portion, and one for a 4 degree-wide center portion. So with seven colors in the environment, there are 21 visual receptor nodes for each agent. Activation on a particular receptor is $1 - \frac{distanceToObject}{maximumVisualDistance}$. If a disc spans more than one visual field, then only the neuron that represents the visual field that contains the closest point on that disc is activated.

Agent effectors consist of: turning left or right with some movement forward (i.e. an agent cannot spin in place); moving forward, left, right or backwards; eating; drinking; painting discs (i.e. changing the color of a non-food or non-water disc); and grasping. Grasping must be done continuously over a period of time for an agent to hold onto a disc for that period. Dropping is done as a cessation of grasping. Agents can only grasp one disc at a time; agents can only move when grasping a disc if the agent's strength is larger than the weight of the disc. Agents can see past (over) discs they are grasping.

Time is divided into discrete time-steps. At each time-step, every agent is given in parallel an opportunity to act. Each agent returns a set of actions, all of which are carried out in parallel, e.g. an agent can simultaneously execute a grasp, move, and turn. The size of a time-step in the world is roughly analogous to a fraction of a second. The farthest an agent can move in a single time-step is 4 units, or one third of its diameter. Typically, agents move around 1 unit per time-step. For these experiments, there were no predators and no other agents not involved in the wall-construction task.

As seen in figure 2, the environment is very sparse. It takes around 1000 time-steps for an agent to travel from the start of the wall to the water. Because of this, the agents' hunger and thirst is set so that each needs to drink at least every 10,000 time-steps and eat every 20,000 time-steps without dying. This ensures that the agents will need to perform these actions while still giving them time to complete the wall.

3. Neural Architecture

The control mechanisms in our agents are neurally inspired. The networks are currently hand designed by the researchers.

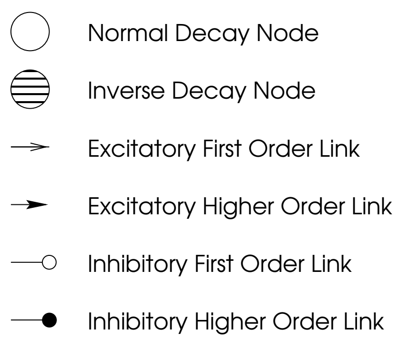

Figure 3 Key for node and link types.

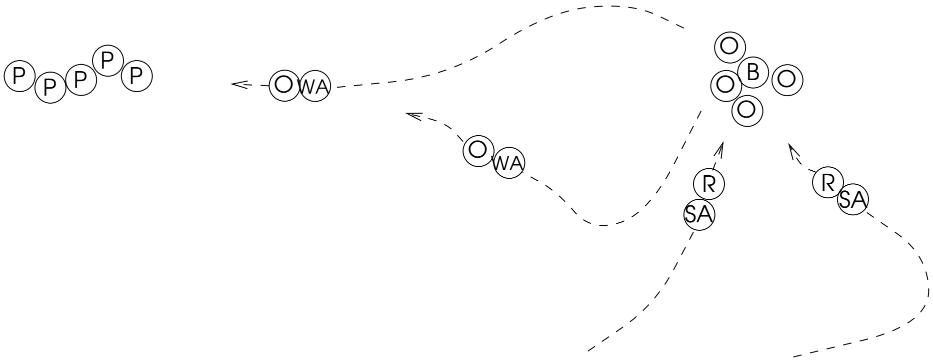

Figure 1 Schematic of the *one-end-out* wall-building algorithm in action. While colors are used in the model, here purple discs are marked with Ps, orange discs with Os, and red discs with Rs. Scavenging agents (SA) are bringing red discs to the pile marked with a brown disc. Wall-building agents (WA) are taking the the orange discs to the closest purple disc. Agent motion is indicated by the arrows.

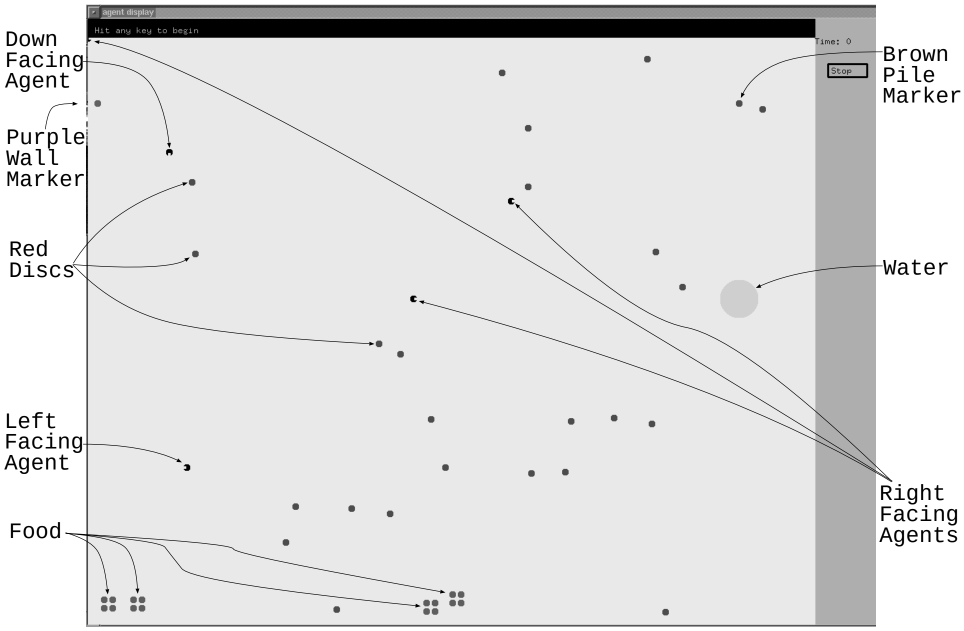

Figure 2 Screen dump of the starting position for the one-end-out task. The large disc on the right is the water, and the food is in four clusters along the bottom. The wall marker (purple) is in the upper left and the scavenge pile marker (brown) is in the upper right.

3.1 Neuron/Connection Types and Dynamics

Agents' neural networks are made up of four types of links and four types of nodes. Figure 3 contains the key for link and node types used in this paper. Nodes spread activation to links which calculate their own activation level before passing it on to other nodes, which sum the activation of all incoming links. Both nodes and links have an activation associated with them, as well as an activation function that is applied to the activation. Having activation functions on links allows us to remove nodes whose sole purpose is to provide an activation function. The equations shown here are used to calculate the activation on nodes and links and are explained below:

$$A_n^{t+1} = f(A_n^t \eta + \beta + \sum_{i=1}^{n} L_i^t), \quad (1)$$

$$A_n^{t+1} = f((1 - \eta(1 - A_n^t)) + \beta + \sum_{i=1}^{m} L_i^t), \quad (2)$$

$$L_n^t = f(\beta + A_n^t \omega \prod_{i=1}^{m} H_i^{t-1}). \quad (3)$$

Nodes Nodes hold some of their activation over time instead of becoming zero before each time-step. Associated with each node is a decay factor between 0 and 1, controlling how much of the activation is held from time-step to time-step. The four node types are: Normal decay, Inverse Decay, Sensor and Effector. They behave as follows:

- All nodes have activation values $0 < A < 1$.
- Normal Decay nodes sum their input from: (a) other nodes, (b) a bias and (c) the node's previous activation multiplied by a decay rate. Then an activation function is applied. Equation 1 is used to calculate the activation of a normal decay node, where A_n^t is the activation of node n at time t, $f(x)$ is the activation function, β is the bias, η is the decay parameter, and L_i^t is the activation of incoming link i at time t. The activation function can be any function, but is typically a sigmoid, step, or linear function.
- Inverse decay nodes behave like regular nodes, except they decay upward towards 1 (see equation 2). An example of the use of inverse decay nodes is seen with Avoid nodes, discussed in section 3.5
- Sensor nodes take input from the external environment rather than from other nodes via links.
- Effector nodes do not output through links, but rather output to the simulator to carry out the set of actions that an agent takes.

Links The links are made up of two classes: *First-Order Links*, and *Higher-Order Links*. In each class, a link can either be excitatory or inhibitory. Higher-order links, like nodes, have activation values $0 < A < 1$, but first-order links can have the value of any floating point number as their activation. First-order links are links whose destination is a node; the activation of the first-order links are summed at a node. Higher-order links modify the strength of either first-order or other higher-order links. Excitatory higher-order links multiply the activation of the higher-order link by the activation of the lower order link. If the value of an excitatory higher-order link is zero, it effectively turns off the link that it modifies.

Inhibitory higher-order links multiply the activation of the modified link by one minus the activation of the higher-order link. If the activation of the inhibitory higher-order link is one, it turns off the lower-order link; if the activation of the inhibitory higher-order link is zero, then the lower-order link operates as if no modification occurred. Higher-order links act as secondary weights that modulate other links according to activation elsewhere in the network.

All links calculate their activation with equation 3 where L_n^t is the activation of link n at time t; H_i^t is the activation of higher-order link i at time t (if the higher-order link is inhibitory; H is one minus the activation of the link); $f(x)$ is the activation function; β is the bias; A_n^t is the activation of the source node at time t; and ω is the weight (weights can be any floating point number). If the link in question is itself a higher-order link, its value will be used to calculate the activation of another link. If the link is a first-order link, then its activation is used to calculate the activation of its target node. If a first-order link has a negative weight, then it is a inhibitory link, otherwise it is an excitatory link.

Example Figure 4 shows a sample network. At each time-step, first the activation of the links is calculated using equation 3. For a particular link, the activations (at the previous time-step) of all the higher-order links that modulate that link are multiplied together and then multiplied by the activation of the source node and the weight. The bias is added and the activation function is applied.

For example, the activation of link **a** at time t_4 is equal to the activation of node 1 at time t_4, times the weight of link **a** times the activation of links **d** and **e** at time t_3, added to the bias and passed through the activation function. Higher-order links are initialized to 0, so that at time t_0, excitatory links do not excite the link they modulate and inhibitory links do not inhibit the links they modulate.

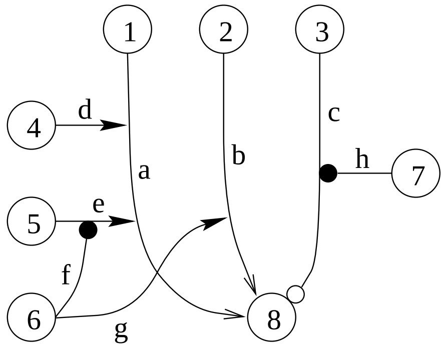

Figure 4 An example network. Numbers 1-7 are nodes, and letters a-h are links.

In the second half of the time-step, the activation of the nodes is calculated using equations 1 and 2. For example, the activation of node 8 at time t_5 is equal to the sum of the activations of links **a**, **b**, and **c** at time t_4, plus the activation of node 8 at time t_4 times node 8's decay factor, plus the bias, and all passed through node 8's activation function. The activation of inverse decay node 3 is based only on its decay because it has no inputs. If the bias of node 3 were 0, the decay factor were 0.5, the activation function were $y = x$ and the activation were 0.75 at time t_4, then at time t_5, its activation would be: $1 - 0.5(1 - 0.75) = 0.875$. If, at time t_4, link **h** had an activation of 0.3, and link **c** had a weight of 1, a bias of 0 and an activation function of $y = x$, then the activation of link **c** at time t_5 would be: $0.875 \times 1 \times (1-0.3) = 0.6125$.

3.2 Basic Level Neural Circuits

In the agents' neural networks, different nodes have different uses. Figure 5 is the key for all of the node types in all of the further diagrams in this paper. Goal nodes (squares) initiate neurocircuits that achieve lower-level goals. Examples goals include: color a disc orange, grasp a disc, approach a disc, and avoid a disc. Detection nodes (diamonds) use input from sensors and become active (or inactive) when variusconditions are true (or false). For instance, they detect whether an agent is holding or touching a particular type of disc. Sequence nodes (downward pointing triangles) are used to ensure that a set of actions occurs in a specific order. Motivation nodes (pentagons) send activation to goal nodes, as controlled by sequence nodes, to achieve a set of goals in an appropriate order.

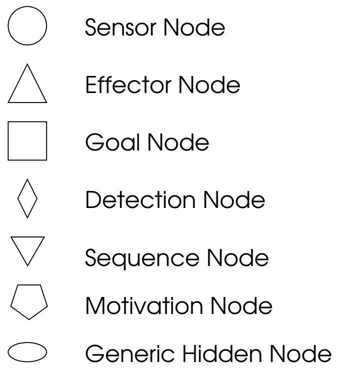

Figure 5 Key for node uses. The shape is a mnemonic device that indicates the use of the node in the network.

Figure 6 shows a reactive network with higher-order connections. These reactive networks form the basic building blocks of the more complicated networks. Here, the sensory nodes have links directly to the effector nodes, causing the agent to move toward red discs. When the agent sees a red disc to the left, the direct connection from Red Disc Left to Turn Left causes the agent to turn left. When there are multiple red discs activation might reach both turn left and turn right, and both actions are taken in parallel. The turns cancel each other out, so the agent ends up moving forward with a slight turn to whichever side has the stronger activation. At the next time-step, one of the red discs would be slightly closer, so the agent turns even more toward that side, and eventually heads toward the disc in a curved path. While this curved path is less efficient than moving straight toward the closest disc, it does have the advantage of delayed commitment. If the closer disc is picked up by another agent (as often happens) the agent is in a better position to move to the other, farther, disc.

Notice that in figure 6, the links from sensors to effectors are modified by higher-order links from the Approach Red Disc Goal node. The Approach Red Disc node represents the agent's goal of approaching a red disc. When it has activation, it makes the links (from the red sensor nodes to the effector nodes) become stronger. So if the agent sees a red disc, the agent is more likely to move toward red. When the Approach Red Disc Goal node is not active, this reduces the strength of the links from the red sensory nodes to the effectors, making it less likely that the agent will act on red input.

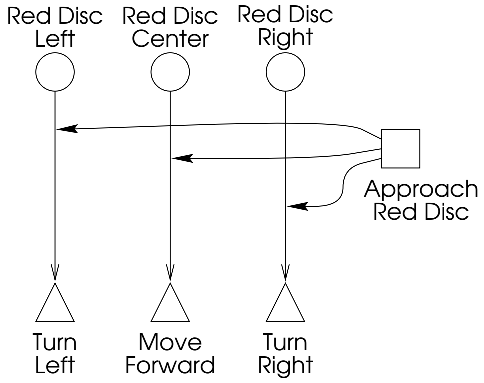

Figure 6 Gated reactive Approach Red Disc network.

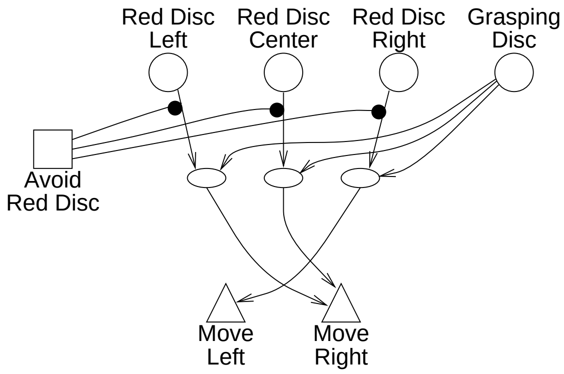

Figure 7 Gated reactive Avoid Red Disc network

In any agent's neural network, there are many of these Goal neural circuits, including: ones to approach each color of object as well as to approach food and water. There are also avoid circuits for avoiding collisions with objects or other agents (Figure 7). Another type of basic circuit is the detection circuit. In these circuits, there is a detection node that becomes active to indicate that a particular external world state is true. Figure 8 shows the Touching Red Disc node and its circuit. Touching Red Disc becomes active when the agent is touching a red disc. These circuits (goal and detection)are used as subcircuits in other larger circuits to generate more complex behavior.

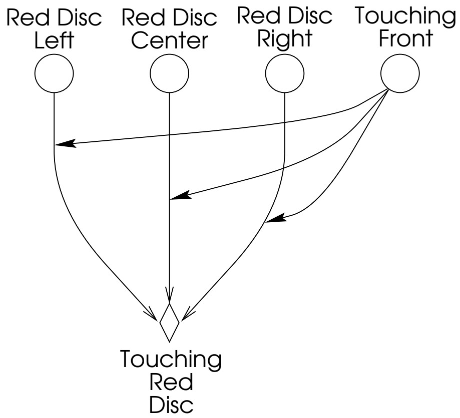

Figure 8 Touching red disc detector circuit. The higher-order connections from Touching Front ensure that the agent is both near a red disc and touching a disc.

3.3 Sequencing

Figure 9 shows a sequence circuit for scavenging. The agent must find and approach something to scavenge, pick it up, carry it back to the pile, drop it, and then repeat the process. In the scavenge circuit, the *motivation node* (pentagon) feeds activation to the goal nodes (squares) as modulated by the *sequence nodes* (triangles). *Detection nodes* (diamonds) control when one sequence node is turned off and the next is turned on. The process is as follows:

1. The Scavenge node begins with some activation, indicating a motivation to scavenge. None of the sequence nodes have any activation, so the links from the motivation node to the goal nodes do not pass activation.
2. The link from the Scavenge node to the first sequence node (S1) has a threshold activation function. If the activation on the Scavenge node is over the threshold on the link, S1 becomes active. Sequence nodes do not decay and so remain active until explicitly inhibited.
3. S1 has a higher-order connection to the link from Scavenge to Approach Red Disc, so when the sequence node becomes active, it allows activation to pass from Scavenge to Approach Red Disc. The Approach Red Disc node controls the agent to move to a red disc (see figure 6).
4. The link from S1 to the second sequence node (S2) is controlled by the Touching Red Disc detection node. When the agent is finally touching a red disc, the detection node becomes active, turning on the link to S2 via a higher-order connection. S2 immediately inhibits S1. It also has an inhibitory higher-order connection to the link from the Scavenge node to S1. This prevents the scavenge sequence from restarting before it is completed.
5. The higher-order connections from S2 turn on the links to the Hold Disc and Approach Brown Disc goal nodes. Because the goal nodes decay to zero immediately, the Approach Red Disc node becomes zero, while the Hold Disc and Approach Brown Disc nodes get activation from the Scavenge node. These goal nodes excite the neurocircuitry (not shown) that causes the agent to approach the brown disc while simultaneously holding on to the red disc.
6. When the agent arrives at the brown disc, it will detect that the red disc it is holding is touching the brown disc. The Disc Held Touching Brown Disc node will become active, thus modulating the connection from the S2 to the third sequence node (S3). S3 turns off S2 and allows activation to pass to the Color Disc Orange goal node.
7. As soon as the disc has been colored orange, the Touching Orange Disc detection node becomes active, turning on the link from S3 which inhibits S3 itself. Now, none of the sequence nodes have activation, and the sequence can begin again, as long as the activation of Scavenge remains above the threshold.

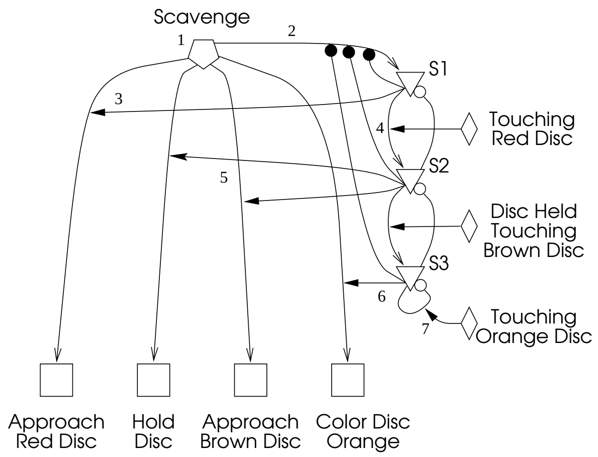

Figure 9 A basic sequence circuit: numbers correspond to the steps in §3.3

3.4 Backtracking and Conflict Resolution

In complex and dynamic worlds, states that an agent detects do not always hold for the duration of a step in a sequence. This can happen because one of an agent's actions failed (e.g. the agent tried to grasp an object too far away), because of the actions of other agents (e.g. an agent moves another agent's previously placed disc), or because of the actions made by the agent to

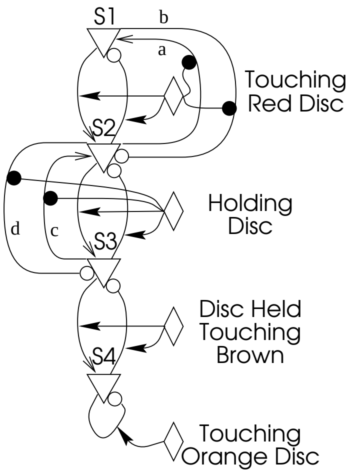

Figure 10 An extension of the sequence network in figure 9. The backtrack links are labeled a-d. The goal nodes and motivation node are omitted, as are the higher-order links from the sequence nodes. A fourth sequence node is added to account for a possible failure between the time an agent is touching a red disc and when it actually is holding the disc.

meet other goals (e.g. dropping a disc in order to eat or drink). When this happens, the agent needs to go back in the sequence to re-establish the necessary preconditions. Figure 10 shows an extension to the network shown in figure 9. This extension allows the agent to move back to earlier steps if it ever needs to. During normal operation of the Scavenge sequence circuit, detection nodes turn on links that move the sequence one step forward. In figure 10, they also turn *off* links that would move the sequence one or more steps backwards. If a state that needs to be true ever becomes false, the detection node will no longer be active, so it will stop inhibiting these backward links, and thus the agent will return to an earlier step in the sequence. For example:

1. An agent at step two of the sequence (i.e. the second sequence node (S2) is on), has detected that it is touching a red disc and is about to pick it up. Let us assume that at that very moment, another agent picks up the very same disc and carries it away.
2. As a result, the Touching Red Disc detection node turns off. The links (labeled a and b in figure 10) that the node was inhibiting now become active; thus the first sequence node (S1) becomes active, turning off S2. The agent then approaches a different red disc (caused by S1), grasps it (caused by S2) and moves to step three, approaching the brown disc.
3. At this point, let us assume the agent becomes thirsty and approaches the water while still carrying the red disc. Because the agent cannot both hold a disc and drink at the same time, the quench thirst circuit (not shown) inhibits the Hold Disc goal node once the agent is close to the water. At this point the Holding Disc detection node turns off, and the backtrack links (c and d in figure 10) turn on, thus activating S2 and inhibiting S3. Because the agent is not touching a red disc, links a and b activate S1 and inhibit S2. Thus an error causes activation to *cascade* backwards through the sequence until an appropriate recovery step is encountered. When the agent has finished drinking, it now begins the scavenge sequence at the first step, namely, searching for red discs.

3.5 Use of Inverse Decay

Agents need to avoid collisions with each other and with stationary discs (so they avoid getting stuck near other discs or in crevices between discs). To this end, the agents possess avoidance circuits, as shown in figure 7. Because these neurocircuits should be on as a default, their goal nodes are inverse decay nodes, decaying upwards towards an activation of one. If an agent needs to approach an object or other agent (e.g. to mate), then the corresponding avoid goal node needs to be inhibited. Inhibition comes from the corresponding goal node. For instance, the Approach Red Disc goal node inhibits the Avoid Red Disc goal node. Later, when that avoidance circuit needs to be turned back on, inhibition is removed and the node automatically decays back up to one. In this way, other circuits do not have to explicitly turn on all the avoid circuits for all of the possible things to avoid.

3.6 Putting It All Together

The neural networks for the agents have hand-built node/link structure and weights. The scavenging agents have three basic sequence neurocircuits: a scavenge circuit, a quench thirst circuit, and a satisfy hunger circuit. The scavenge circuit has the agents seek out red discs, bring them back to a brown disc, and color them orange. The quench thirst and satisfy hunger sequences have the agent seek out water or food and then drink or eat it, respectively. In addition each agent has avoidance circuits that are selectively inhibited as needed.

The wall-building agents have no explicit idea of building the wall. Instead, the wall is an emergent phenomenon that arises due to the interactions among the scavenger and wall-builder agents. The wall-building agents have three sequence neurocircuits as well. The satisfy hunger and quench thirst sequences are exactly the same as in the scavenging agents. The wall-building sequence is also the same as the scavenging sequence with the exception of the colors of the target discs. This sequence circuit has the agent seek out orange discs, bring them to a purple disc, and color them purple. Because the orange discs are all piled in a small area away from the purple disc, as the wall-building agents bring the orange discs to the closest purple discs, they become strung out in a line toward the orange pile, thus creating a wall.

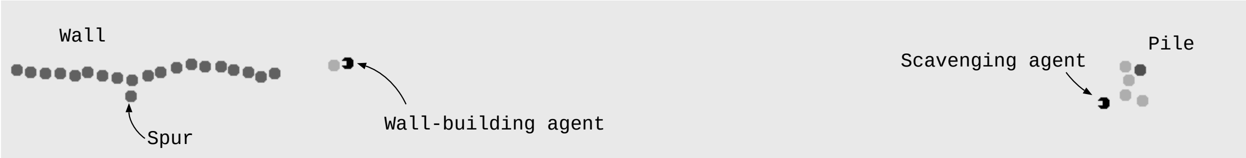

Figure 11 Screen dump of the wall being built. A scavenger is looking for a red disc, while a wall-builder is bringing an orange disc to add to the wall. The other three agents are off the portion of the screen being shown here.

4. Simulation and Discussion

We ran the simulation with 5 agents (2 scavengers and 3 wall-builders), 28 red discs to be used as building material, 1 brown disc to mark where the material should be gathered, 1 purple disc to mark the start of the wall, 1 water disc and 16 food discs on a 1080 by 900 world for 30,000 time-steps. In multiple runs a wall emerges each time with a maximum variation (from straight line) within 10 degrees and an average variation within 5 degrees. The foraging agents proceed to collect the red discs and drop them haphazardly around the brown disc (figure 11, right). The other three agents build a wall by extending it from the initial purple disc toward the brown disc (figure 11, left). The wall is uneven because of the variation in the locations of the orange discs. The wall-building agents bring each orange disc straight to the closest purple disc, so deviations in where the agents start cause deviations in the wall. The spur on the wall was created when an agent became hungry while carrying an orange disc. When the agent got to the food, it dropped the orange disc to eat, and another agent picked it back up again and brought it to the wall. But because the line from the food to the wall was perpendicular to the line from the wall to the brown disc, the agent placed its disc as a spur rather than as a linear addition to the wall.

During the gathering and building, the agents maintained sufficient levels of food and water. Each agent eats once, eating three food discs per meal and each agent drinks three times across an entire wall construction run. There were some cases when agents did die of thirst. This occurred when a wall-building agent became trapped into the pile by scavenging agents. A wall-building agent can become surrounded by orange discs and unable to move; as a result it will eventually die of thirst.

The approach of resolving different goals (via addition of activation at the action level) eliminates the need of having to choose a particular goal or action via a winner-take-all network (WTA). WTA networks lose information when some nodes are suppressed (Tyrrell, 1993b). When implemented in neural networks, WTA networks reduce the modularity of the system, as well as taking time to settle— time that is badly needed for acting in the dynamic environment.

When backtracking on the sequence circuits, cascading backwards over multiple steps enables the circuit to move back to any arbitrary step while limiting the number of explicit backward links. To move directly back to any arbitrary sequence node would require N^2 sets of links for the entire sequence. Using the cascading mechanism reduces this requirement to N sets of links.

Although our system is a neural network, its structure (of nodes gating links to nodes that gate other links) is hierarchical in nature. At the lowest level, each goal node controls an agent's reaction to input. Sequence nodes control which sets of goals are active, and motivation nodes control both the sequence nodes and the goal nodes directly.

5. Related Work

The reactive portions of the circuits described here were inspired by Braitenberg(Braitenberg, 1984), who proposed them as primitive networks for his vehicles. Werner(Werner, 1994) showed that adding higher-order connections to Braitenberg networks could improve the behavior of creatures when finding food and water and avoiding predators. Much of our model was inspired by Beer(Beer, 1990; Beer and Chiel, 1991), who built neural networks by hand to control a simulated cockroach. Beer also uses higher-order connections (which he calls compound synapses) as well as node decay.

Maes's(Maes, 1991) Predecessor and Successor links between behaviors are similar in function to our sequence nodes. However, Maes's architecture is very different in many respects: her behavior nodes specify behavior directly as a set of processes; activation is spread forward from sensors, and backward from motivations, and an external non-neural system picks the maximally responding behavior to execute (a winner-take-all approach).

Higher-order connections are equivalent to multiplicative nodes (nodes that multiply their input rather than sum it). Multiplicative nodes are also used for action selection by Rosenblatt & Payton(Rosenblatt and Payton, 1993) and Tyrrell(Tyrrell, 1993a). A good survey of hierarchical action selection structures can be found in (Tyrrell, 1993a).

Therauluz & Bonabeau(Therauluz and Banabeau, 1995) use collections of simple rule-based wasp-like agents to build nests. Their wasps could build complicated structures using only local rules. The wasps communicated by observing and changing their local environment, similar to our agents observing and changing the colors of the discs.

6. Future Work

We have both short-term and long-term areas of future work. Short-term future work includes: (1) Creating mechanisms for searching for input when there is none. If an agent has activation on goal nodes but senses no objects for satisfaction of the goals, it is not motivated to seek them out. We will introduce mechanisms to cause the agent to seek stimuli in their absence. (2) Investigation into other wall-building approaches, such as the two-ends-in, and the fill-in. (3) Construction while avoiding predators. (4) Construction of complex structures, such as multi-wall enclosures or corridors with intersections. (5) The creation of a foreman who, via signaling, conscripts and directs teams of agents to perform different tasks. (6) Developing agents able tp repair already built structures that have become damaged.

Long-term future work includes: (1) Reenforcement learning. We plan to adapt reenforcement learning techniques to the neural networks and the environment in order to adjust the weights of pre-wired neural networks. (2) Critical period learning. We will investigate a period of time right after the birth of an agent when the agent's neural network structure is easily changed through experience. As the agent ages, this critical period is passed, and the network structure becomes relatively fixed. (3) Imitation learning. Here, agents will learn by imitating the actions of of other agents in the environment. (4) Parenting. Parent agents should protect and provide for child agents, as well as act as role models for the child agents' imitative and critical period learning. Similarly, foremen could train new workers on a job site. (5) Evolution of the network weights and structure. We plan to evolve portions of the neurocircuitry at various levels of detail, including individual weights and connections between subcircuits, and the order of goals in sequences.

7. Conclusion

The neural network shown here can control agents to achieve a sequence of goals. The agents reprioritize according to both internal needs and environment, taking advantage of chance occurrences in the environment while executing multiple actions in parallel and acheiving goals in their proper order. While the behavior of the agents is not optimal (the wall is squiggly and the agents do not always take the shortest route to a destination), what emerges is naturalistic and robust and is completed by multiple agents working concurrently on distinct subtasks.

Acknowledgements

This work is supported in part by an Intel University Research Program grant to the second author. The model is implemented in GNU Common Lisp and runs under Linux on Pentium II based computers, also donated by the Intel Corporation. Software was donated by Microsoft.

References

Beer, R. (1990). *Intelligence as Adaptive Behavior. An Experiment in Computational Neuroethology.* Academic Press, San Diego, MA, USA.

Beer, R. D. and Chiel, H. J. (1991). The neural basis of behavioral choice in an artifical insect. In Meyer, J.-A. and Wilson, S. W., editors, *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 247–254.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge, MA, USA.

Maes, P. (1991). A bottom-up mechanism for behaviour selection in an artificial creature. In Meyer, J.-A. and Wilson, S. W., editors, *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 238–246.

Rosenblatt, K. J. and Payton, D. W. (1993). A fine-grained alternative to the subsumption architecture for mobile robot control. In *International Joint Conference on Neural Networks.*

Therauluz, G. and Banabeau, E. (1995). Coordination in distributed building. *Science*, 269:686–688.

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

Tyrrell, T. (1993b). The use of hierarchies for action selection. In Meyer, J.-A. and Wilson, S. W., editors, *From animals to animats2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 138–147.

Werner, G. M. (1994). Using second order neural connection for motivation of behavioral choices. In David Cliff, e. a., editor, *From Animals to Animats3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 154–161.