Observation and Imitation: Goal Sequence Learning in Neurally Controlled Construction Animats: VI-MAXSON

Frederick L. Crabbe **Michael G. Dyer**
Artificial Intelligence Lab
Computer Science Department
University of California, Los Angeles
4532 Boelter Hall
Los Angeles, CA 90095 USA
{*cardo,dyer*}*@cs.ucla.edu*

Abstract

We report on an under-explored animat learning problem, that of learning sequences of goals, and present an algorithm for solving it using observation and imitation. The presentation introduces the problem as well as a neural agent control architecture we use as a framework in which to use the algorithm. We demonstrate that, by using the algorithm, a learning agent can learn to satisfy a sequence of goals by observing the actions of a teaching agent, and later imitate the teaching agent.

1. Introduction

For an agent to display interesting behavior and accomplish complex tasks, it must be able to select the correct *actions* to achieve multiple *goals* in a sequence. In this paper, actions are the primitive outputs made by an agent, and goals refer to the low-level objectives of approaching or avoiding objects. For example, in order to complete the task of building a structure, an agent might need to satisfy the goals: find raw materials, bring them to the construction site, locate the correct location for the building material, and add it to the structure. The motivation for this work is to create agents that can perform construction tasks while maintaining their survival in a complex and hazardous environment. The agents must be able to balance multiple goals, so that they can pursue survival tasks (such as finding and eating food when hungry) while still meeting sequences of goals required for their construction tasks. Agents that are able to learn to perform these sequences by observing other agents need not be pre-programmed to the task. They can acquire the ability much faster than they would by learning on their own, without observing other agents. This paper presents VI-MAXSON, a model of goal-sequence learning by imitation in a neural architecture.

The construction task in this paper is wall building; the agents must go to a piece of building material, pick it up, bring it to the wall, move to the correct place at the end of the wall, and drop the material. The steps in that sequence are each composed of a single goal that must be met in the correct order, but the sequence of the individual actions that the agent makes to meet these goals are not determined by the task, but rather by the environment and the goals. These actions can occur in relatively free order. For example, the first step in building a wall is always to approach a piece of building material, but the actions needed to meet that goal depend on the locations of the agent and building material. The agents must learn a sequence of goals, not actions.

We use the term *goal sequence learning* because there is a wide range of research that falls under the more general heading of sequence learning. For instance, in data mining, an area of active investigation is to identify sequences of data in databases (Zaki et al., 1999). Similar efforts in animats research investigate agents that learn and detect sequences of perceptual input in order to select actions (Ulbricht, 1996). More commonly, agents learn to perform sequences of outputs. Most of this work involves learning to produce sequences of actions (Morén, 1998, Sun and Peterson, 1998). Learning to achieve sequences of goals differs in that the order of individual actions is left open, but the order of the goals these actions achieve are fixed.

The agents described in this paper begin knowing neither the sequence of goals to be met, nor how to meet those goals, but they use observation and imitation to learn both. In VI-MAXSON, one agent (the 'learner') follows a second agent (the 'teacher') as it performs a

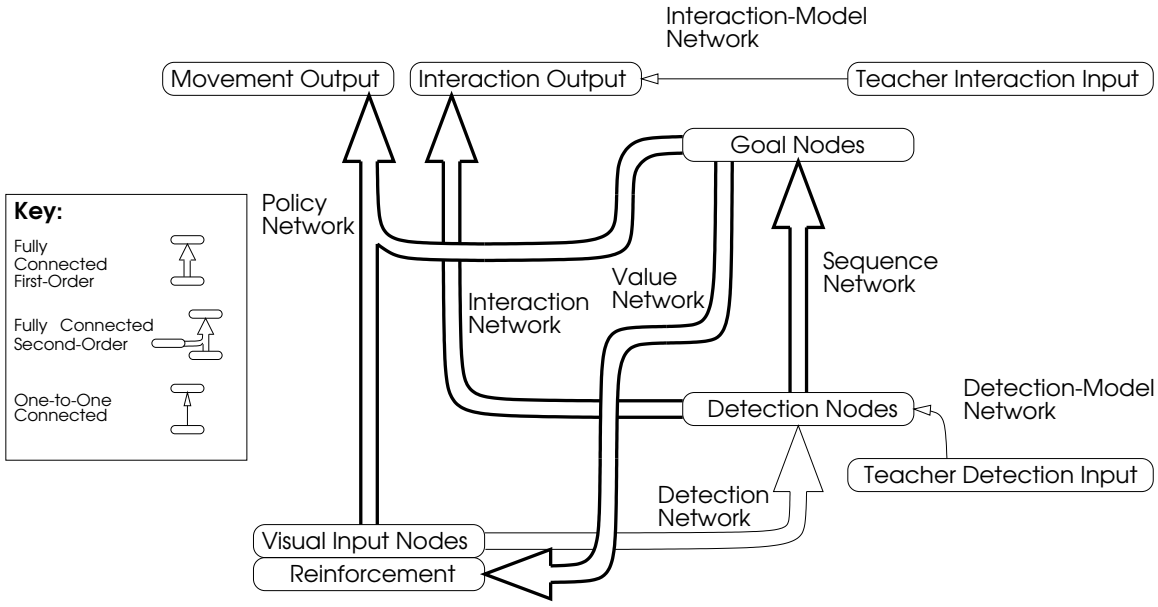


Figure 1: VI-MAXSON agent overview. Ovals represent banks of nodes and arrows represent sets of connections between the banks. The weights in connection sets represented by bold arrows are learned; the weights in the remaining connection sets are innate.

task. As the learner watches the teacher, a portion of its neural network builds a representation of the goals the teacher achieved. Later, when the learner tries to imitate the teacher, it uses a form of reinforcement learning to learn how to achieve each goal.

There are many interesting problems involved in imitation learning among multiple agents, and we intend to address the more abstract problems of identifying and sequencing goals. This will leave a number of areas untouched, such as identifying the actions of other agents. These other problems will need to be solved for complete imitation learning in physical agents, but they are beyond the scope of this paper.

We begin in section 2 by describing the environment the agents are in and the architecture that controls the agents' behaviors. In section 3, we briefly describe how the agents learn to approach or avoid objects such as building materials. In Section 4 we then describe in detail the algorithm for learning to achieve sequences of those goals. Experiments demonstrating the successful operation of the algorithm, where one agent learns to achieve the sequence of goals necessary to build a wall from a second agent, appear in Section 5. This is followed by discussion of advantages and failure conditions of the systems, and finally conclusions.

2. Environment and Architecture

The VI-MAXSON agents reside in the two dimensional continuous environment of DiscoTech (Crabbe and Dyer, 1999a), where all objects are circular (discs). Agents have three kinds of output: movement actions, interactions, and emotive signals. An agent's possible movement actions are: turn left or right (smoothly up to four degrees at a time), and move forward up to four units (where one unit is one twelfth the agent's size). Turn left and turn right both also cause some movement forward. Interactions are atomic actions that an agent performs on a disc. Interactions include: eating food, drinking water, changing the color of a disc, picking up a disc, and dropping a carried disc. The emotive signals are either grins or grimaces, which can convey an agent's pleasure or displeasure. Any or all of an agent's actions may be carried out in parallel. For example, if during one time step, the agent turns right 3.2 degrees, turns left 1.8 degrees, and moves forward 1.6 units, the result is that the agent simultaneously moves forward 2.1 units (some forward motion added by the turns) and turns right 1.4 degrees.

To detect the external environment, the agents have a primitive visual system. An agent can see in a 180 degree field in front of it. Vision is broken up into four input sensors for each type of object the agent can see. The agent is bilateral with two visual sensors on each side for each type of object. These sensors perceive the closest object of that type. For instance, the RDL (*Red_Distance_Left*) sensor measures the distance to the closest red disc on the agent's left side, while the RAL (*Red_Angle_Left*) sensor measures the angle of this red disc relative to the direction the agent is facing. The agent can also perceive the interactions and emotive signals of other agents.

Figure 1 gives an overview of the VI-MAXSON architecture (Crabbe and Dyer, 1999b), consisting of seven

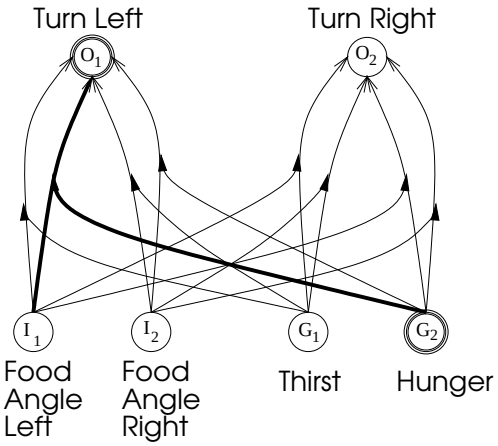


Figure 2: A small portion of a second-order Policy Network. G-nodes (goals) dynamically modulate the influence of I-nodes (sensors) on O-nodes (actions) via second-order connections.

distinct networks. The weights in four of these networks (Policy, Value, Interaction, and Sequence) are learned, while the weights in the remaining three (Detection, Interaction-Model, and Detection-Model) are set when the agent is created. At each time step, actions are selected using the Policy Network, the Sequence Network, and the Interaction Network. The Value, Interaction-Model, and Detection-Model Networks are used as aids to learning. We use this architecture because it has many of the desired action selection properties as listed in (Tyrrell, 1993), including balance between multiple goals, robust sequencing, backtracking, conflict resolution and opportunism.

2.1 Policy Network

The purpose of the Policy Network (figure 2) is to generate movement output for the agent. It combines an agent's visual input with the agent's current goals to determine the agent's movement output at each time step. Connections from visual sensor input nodes go straight to effector nodes, such as *Turn_Left* or *Turn_Right*. These connections are modified by second order connections from a set of goal nodes (Rumelhart and McClelland, 1986, Giles and Maxwell, 1987). Second-order connections multiply the inputs as well as add them, so that the equation for the output nodes is:

$$O_o = \sum_{i \in I, g \in G} I_i G_g W_{i,g,o}, \qquad (1)$$

where I is the activation of the input nodes, G is the activation of the goal nodes, O is the output, and W is the set of weights on the connections. The second-order connections enable the goal nodes to control how much effect the connections from the input sensors to output motors in that sub-network has on the agent's behavior. When the activation of a goal node is 0, it turns off all the connections in its sub-network, eliminating that behavior. An agent's goal nodes include ones that measure the agent's internal states, such as *Hunger*, *Thirst* and *Pain*; ones that control general approach/avoid goals such as *Approach_Red*, or *Avoid_Green*, and one node used for learning from other agents, called the *Vicarious* node.

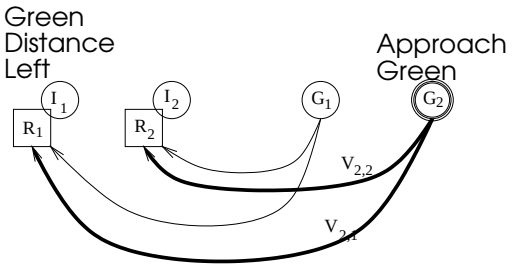


Figure 3: A small portion of a first-order Value Network connecting goal nodes to internal reinforcement. Each R_i node holds the amount of the reinforcement for the associated input sensor node (I_i).

2.2 Value Network

The Value Network (figure 3) stores an agents opinion on the worth of external discs. It is a first-order network that is fully connected from the goal nodes to the reinforcement nodes (one reinforcement node is associated with each visual sensor). The function of the Value Network is to store the intermittent external reinforcement the agent receives. It then distributes external reinforcement to internal reinforcement nodes. The Policy Network can then use this internal reinforcement to efficiently learn to approach and avoid discs in the environment (see section 3).

2.3 Detection Network

The Detection Network takes sensory input and detects when the agent is in one of various states with respect to the environment[1]. In this paper, the weights of the Detection Network are set when an agent is created. A node in the Detection Network becomes active when the agent is in the corresponding state for that node. The states that can be detected (and the nodes that detect them) are: the agent is touching a disc (*Touching_Red*, *Touching_Brown*, etc.); the agent is holding a disc, i.e. has picked up and is carrying a disc (*Holding_Red*, *Holding_Brown*, etc.); the disc the agent is holding has bumped into something (*Carry_Bump_Red*, *Carry_Bump_Brown*, etc.). There is also a default state for when the agent is neither touching, holding, nor carry-bumping into a disc. The node for this default

[1] We will sometimes refer to these as environmental states.

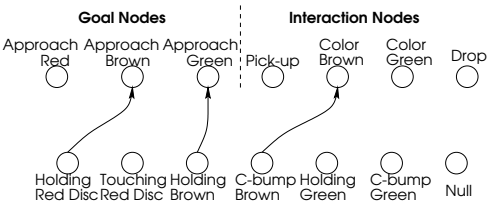


Figure 4: A portion of both the Sequence Network and the Interaction Network, showing how the agents perform a sequence.

state is called *Null*.

2.4 Interaction Network

The Interaction Network connects the Detection Nodes with the Interaction Output Nodes (figure 1). The Interaction Network controls the agent's interaction output when it is in a particular state. For instance, the network might cause the agent to *pickup* when it is touching a red disc. The weights in the Interaction Network are learned as part of the sequence learning (see section 4).

2.5 Sequence Network

The Sequence Network connects the Detection Nodes to the Goal Nodes (figure 1) to control the agent to satisfy a sequence of goals. When an agent is in a particular state, the corresponding detection node is active. This activates one or more goal nodes, thus controlling the agent's behavior. Then when goals are satisfied, the agent's environmental state changes, resulting in a change in the detection nodes, and thus a change in the active goal nodes. The Sequence Network does not explicitly represent where it is in a sequence, but uses external cues (such as the color of the disc it is holding) to keep track of its position in a goal sequence. For example (figure 4), when a red disc holding agent bumps into a brown disc, a different detection node, *Carry_Bump_Brown*, becomes active, which causes the agent to color the disc it is holding brown via the Interaction Network. This activates the *Holding_Brown* detection node, which in turn activates the *Approach_Green* goal node. The agents use color to mark the external world, and thus where they are in a sequence because the architecture has no explicit working memory module to keep track of what has been done before. If detection nodes could include past sensory information, such as the fact that the previous environmental state was a *Carry_Bump_Brown*, then this coloring would be unnecessary.

3. Goal Learning

This section will briefly cover the aspects of goal learning that are necessary for understanding the sequence learning algorithm. Details of the Goal-learning algorithm can be found in (Crabbe and Dyer, 1999a, Crabbe and Dyer, 1999c).

Learning in the Policy Network uses the Value Network (figure 3), to control the learning. The Value Network is fully connected from the goal nodes to reinforcement nodes associated (one-to-one) with the sensors. Weights in the Policy Network represent the agent's attitude toward a particular type of object for a particular goal; a positive weight reflects a positive attitude towards the objects that activate that sensor with respect to that goal; a negative weight reflects a negative attitude.

At each time step, the Value Network generates a reinforcement signal for each sensor based on the change in activation (temporal difference) on that sensor. If the weight on the connection is positive and the sensor activation increases, then the reinforcement is positive, but if the sensor activation dropped or if the weight was negative, the reinforcement is negative. For example, if the weights between the *Approach_Green* goal node and the green disc sensors are 1, and the agent moves toward a green disc so that the *Green_Distance_Left* sensor node goes up in activation, then the reinforcement node associated with the *Green_Distance_Left* gets positive reinforcement. In the Policy Network, the weights on the second-order connections from all sensors with positive reinforcement *and* the maximally responding goal node *to* the maximally responding output node are increased (Bold connections in figure 2). Similarly, weights are decreased on connections from sensors with negative reinforcement and maximally responding goal node to the maximally responding output node. The overall effect is that moving closer to objects toward which the agent has positive weights (or moving away from objects toward which the agent has negative weights) increases the strength of the connections that caused that movement. Moving toward objects with negative weights (or away from objects with positive weights) decreases the strength of the connection that caused that movement. Thus the sensory reinforcements that are learned by the Value Network are used to train the Policy Network.

The weights in the Value Network are set when there are large changes in the activation of a goal node. If the change in the goal node activation coincides with the perception of an *event*, weights are changed on the connections between that goal node and the sensor nodes detecting that event. An event is defined as an interaction performed by an agent on an object. The simplest form of this event-based learning is when the agent senses itself interact with a disc. For example, when an agent eats food, its hunger goal node goes down. At the same time, it visually senses the eating event, and increases the weights on the connections from the hunger goal node to the sensors that sense the food. This way the agents stores in the Value Network the information

that the food satisfies the hunger goal. Conversely, if the agent picks up a sharp object, its pain goal node rises, and the weights from the pain goal node to the sensors for that object are made negative, indicating that the object thwarts the goal to avoid pain.

The ability to sense events involving other agents enables the agents to *vicariously* learn from the effects of actions performed by other agents have on those agents. When an agent performs an act that causes one of its goal nodes to drop in activation, it instinctively grins, and when it performs an act that causes one of its goal nodes to rise in activation, it instinctively grimaces. For instance, whenever an agent (agent A) eats food, its hunger goal node drops and it grins. The observation (by agent B) of grins and grimaces causes changes in a goal node called the *Vicarious* or *Vic-node*. When agent B observes a grin, the activation on the *Vic-node* falls, and when agent B observes a grimace, the activation on the *Vic-node* rises (this operation is hard wired into the agent). This change in the *Vic-node* activation triggers learning in the Value Network, associating a positive (or negative) attitude with the object that agent A interacted with. Agent B then can learn in its Policy Network (via reinforcement) as if it had interacted with the object itself.

In essence, the Policy and Value Networks together function to convert the delayed reinforcement that comes from interacting with an object, to constant distributed reinforcement when observing that object. By converting touch or taste related reinforcement to reinforcement connection with vision, the agent is able to learn to approach or avoid an object whenever it *sees* that object. A mechanism similar to vicarious learning is used in the sequence learning, described below.

4. Goal Sequence Learning

In addition to goal-oriented and vicarious learning, agents must learn to achieve their goals in a specific sequence. For instance, an agent can only color a disc green and drop it at a construction site *after* it has picked up a red disc and moved to the construction site. In this section we address the problem of learning a correct sequence of goals from observation and imitation.

There are a number of hard problems to address in learning sequences of goals by imitation: How does the learning agent detect what goal the teacher is pursuing from the actions that the teacher makes? How does the learning agent associate actions performed by the teaching agent with its own actions at a later time? How will the learner learn the different kinds of knowledge needed (i.e. the declarative knowledge of what the steps are and the procedural knowledge of how to perform the steps)?

4.1 Goal Identification Problem

A difficult problem for learning sequences of goals by imitation is the *goal identification problem*. As the learner observes the movements made by the teacher, how does the learner know what goal the teacher is pursuing? If a teacher is moving toward both a brown disc and a red disc, how does the learner know which disc is the intended target? One way is to guess the teacher's goal, given both the teacher's action and sensory state, but this is not easy. How can the learner have access to the teacher's sensory state without riding on the teacher's back (an approach taken by (Cecconi and Parisi, 1993))? What does the learner do if it isn't clear what goal is being pursued; if the teacher could be approaching either a brown or a red disc, how do we adjust the weights? What if the learner doesn't yet know how to achieve the goal; how will it be able to identify it? We address this problem with a new technique: instead of guessing what the teaching agent is doing as it is doing it, we wait until that part of the task is complete. If a teacher is moving toward a collection of objects, it is unclear what its intention is, but when it picks up one of those objects, then it becomes clear which object the teacher intended to approach. The teacher's act of picking up the green disc is a clear indication that the teacher was approaching the green disc.

4.2 Perspective Problem

Another difficult problem for an agent that imitates another is the *perspective problem*: how does the learning agent correspond sensory input and actions of the teaching agent to its own actions? When the learning agent observes the teaching agent picking up an object, that sensory input may be entirely different from the input the learning agent gets on its own sensors when *it* picks up an object. We solve this problem by introducing one-to-one connections between sensors that detect properties of the teacher, and corresponding sensor and effector nodes in the learner. There are two banks of these connections. First, sensor nodes that detect that the teacher has performed an action, such as grasp or eat, are connected to the learner's motor nodes for those actions (figure 5). Second, detection nodes that detect the environmental state of the teacher are connected to the corresponding detection nodes for the learner (figure 6). This way, sensing the teacher automatically activates the corresponding nodes in the learner. Note that such an explicit mapping is common in other imitation systems (Bakker and Kuniyoshi, 1996).

4.3 Knowledge Problem

As a teacher achieves a sequence of goals, it uses two kinds of knowledge: the order goals should be achieved

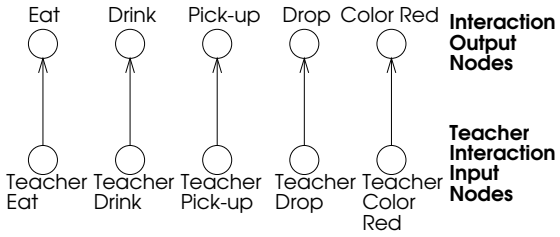


Figure 5: Shown here are some of the one-to-one connections that map the sensor nodes (that detect interactions by a teacher) to the output nodes that (perform the same interactions in the learner).

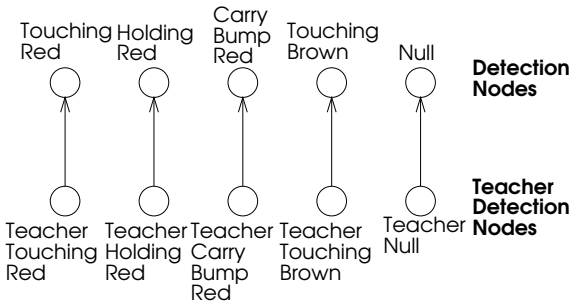


Figure 6: Shown here are some of the one-to-one connections that map the sensor nodes (that detect environmental state of the teacher) to the detection nodes (that detect the learner's environmental state).

Algorithm 1 : Goal Sequence learning

S is the weight matrix for the Sequence Net,
WA is the weight matrix for the Interaction Net,
$\vec{G}^t$ is the goal vector at the current time step,
$\vec{G}^{t-1}$ is the goal vector at the previous time step,
$\vec{D}^t$ is the detection vector at the current time step,
$\vec{D}^{t-1}$ is the detection vector at the previous time step and,
$\vec{A}$ is the interaction output vector at the current time-step.

$g \leftarrow Select\_random\_goal\_index()$ (1)
$G_g^0 \leftarrow 1$
for each event at time t, **do**:
  $G_g^t \leftarrow 0$ (2)
  for each $g \in \vec{G}$ and $d \in \vec{D}$ **do**: (3)
    if $G_g^t - G_g^{t-1} < 0$ and $D_d^t - D_d^{t-1} < 0$ **then:**
      $W_{d,g} \leftarrow G_g^{t-1} - G_g^t$
  for each $a \in \vec{A}$ and $d \in \vec{D}$ **do**: (4)
    if $D_d^t - D_d^{t-1} > 0$ **then:**
      $WA_{d,a} \leftarrow A_a$
  $g \leftarrow Select\_random\_goal\_index()$ (5)
  $G_g^t \leftarrow 1$

in, and the method of achieving these goals. When a learner does not know how to achieve a goal it must learn not only when in the sequence to achieve the goal but also how to achieve the goal. We solve this problem by having agents learn the sequence of goals by observation, but learn how to achieve those goals by physically pursuing them. This is similar to how people learn many physical skills, such as sports. They watch an instructor perform the task and they note the order of the actions, but they only become proficient at the sport after repeatedly performing the sequence themselves. Our learning algorithm sets the weights in the Sequence Network as the learner watches the teacher. It also sets the weights in the Value Network with vicarious learning. Then the Value Network enables the Policy Network to learn to achieve a goal.

4.4 Goal Sequence Learning Algorithm

By combining the architecture and learning structures described in sections 2 and 3, with the ideas listed above, we developed a neural goal sequence learning algorithm (algorithm 1).

Initially the learning agent has a bank of un-used goal nodes. The un-used goal nodes are fully connected into all the networks, but the weights of the connections in and out are 0. A procedural mechanism randomly selects one of these un-used nodes and activates it (step 1). Then, each time the teacher performs an interaction event that is detected by the learner, this detection triggers learning in the learner. The event causes the activation on the currently selected un-used goal node to fall to 0 (step 2). Just as with vicarious learning, this triggers learning in the learner's Value Network, so that the next time the goal node is active, reinforcement will be generated, causing the agent to learn to approach that object. The action performed by the teacher changes the teacher's state, which is sensed by the learner's Teacher Detection Nodes. For example, when the teacher's state changes from *Null* to *Holding_Red*, the activation of the learner's *Teacher_Null* drops and *Teacher_Holding_Red* rises. These nodes in turn change the learner's detection nodes (figure 6). In step 3, the weight on the connection between the detection node whose activation also just fell and the goal node whose activation just fell, is increased. The agent learns that the previous goal should be pursued when in the previous state, as detected by the detection nodes. In step 4, the weight on the connection from the detection node whose activation just rose and the act node for the event, is increased. The learner learns that when it is in the new state, it should perform that interaction. Finally, a new goal node is randomly selected from the pool of remaining un-used goal nodes (step 5).

Imagine the learning agent observing the teaching agent when the teacher is in the null state. In the learner, an unused goal node is active, as well as the sensor nodes that detect that the teacher is in the null state, and therefore the learner's own *Null* detection node is active. Then the teacher bumps into a red disc and picks it up. When the teacher bumps into the red disc, the learning agent's *Teacher_Null* loses activation and the *Teacher_Touching_Red* becomes active because the learner sees that the teacher is now touching the red disc. This changes learner's own detection nodes, dropping the activation in *Null* and raising the activation in *Touching_Red*. When the teacher picks up the red disc, the interaction is detected as an event on the visual input sensors in the learner. Inside the learner, the unused goal node falls, causing the weight between the unused goal node and the *Null* goal node in the learner to increase. The fall of the goal node also causes the weights in the Value Network between that goal node and the sensor nodes for red discs to go up. The increase in the *Touching_Red* detection node causes in the learner an increase in the weight from the *Touching_Red* to *Pick_Up* interaction output node. Later, when the learner is trying to perform the sequence, it begins in the null state. The visual and tactile input makes the *Null* detection node active. This will activate the goal node[2] which will generate reinforcement to cause the Policy Network to learn how to approach a red disc, as discussed in section 3. Once an appropriate policy is learned, the agent will approach a red disc: the visual and tactile input causes the *Null* detection node to become active, which in turn causes the new goal node to become active, which will (via second-order connections) enable visual input of red discs to cause motor output that moves the agent toward the red disc. Then, when the agent touches a red disc, the connection in the Interaction Network causes the agent to pick up the red disc, thus completing the first step of the sequence.

5. Simulations

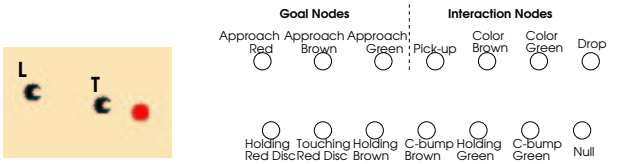
In the simulations, the task is to build a wall out of scavenged materials. The agents do not have an internal map, so the location to build the wall is marked in the environment with its two endpoints, one green and one brown. The agents' sensors are too poor to detect colinearity of objects, so they use a specific technique to keep the wall straight: the agent goes to where one end of where the wall should be (the brown disc), and then turns and moves toward the other end (the green disc), dropping the material when it runs into part of the wall. This ensures that the new material is roughly between the two end points of the wall. As this behavior is repeated, the wall grows from one end toward the other. This technique requires the following sequence for the agents: approach a red disc and pick it up, approach a brown disc (one end of the wall); when it gets there color the red disc brown (changing the active detection node); approach a green disc and when it gets there, color the brown disc green and drop it.

The teaching agent begins with the goal sequence precoded in its networks. The learning agent begins with randomly selected weights less than 0.1 in its Policy Network. The weights in its Sequence, Interaction, and Value Networks are all 0. It also has a separate hard coded network (not shown here) that causes it to follow the teacher in order to observe the teacher's behavior. During learning, the learner observes the activity of the teacher. Then we turn off both the learner's follow-the-teacher network and the sequence learning, and turn on its Policy Network. Finally the learner uses the goal sequence learned to adjust the weights in the Policy Network as it builds the wall alongside the teacher.

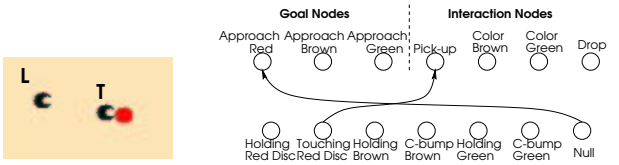
Figure 7 shows a learner following a teacher as it builds a wall. It also shows the weights in the Sequence Network (connections from detection nodes to goal nodes) and Interaction Network (connections from detection nodes to interaction outputs). The weights are increased as expected, creating networks that enable the learner to perform the goal sequence. The one unexpected weight change, which occurred in every simulation, was between the detection node *Carry_Bump_Green* and the goal node that will become *Approach_Green*. This is not necessary for the sequence and was not programmed into the teacher, but does not affect the learner's behavior because it is already at green when the *Carry_Bump_Green* node becomes active.

Figure 8 and figure 9 show the weight values on the twelve Policy Network connections between the goal node that eventually learns to approach brown and the brown visual sensors. Figure 8 shows the learning over the first 250 time steps, when the agent both has that goal node active, and sees a brown disc. The value of the weight on the connection from the sensor *Brown_Angle_Left* to *Turn_Left* (connection #0) and the value of the weight on the connection from the sensor *Brown_Angle_Right* to *Turn_Right* (connection #10) both rise quickly. Figure 9 shows the same weights from time step 256 to time step 1800 (still only when the goal node is active and the agent sees brown). The distance sensor connection weights also rise to 1, but more slowly because turning toward a disc is less well-associated with moving closer to the disc. When an agent turns toward a disc, improvements in the distance are small. We were surprised to discover that the weights on the connections to *Move_Forward* did not rise. Instead, the agent moved toward the brown disc by turning back and forth. This implies that the *Move_Forward* output was not needed.

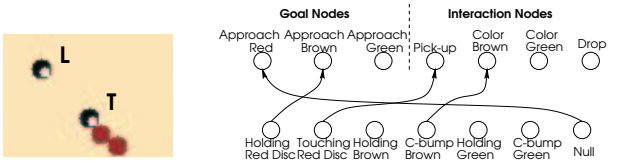
[2] This goal node was un-used up to this point and so has no name. After the policy network learns to approach red, it can be called the *Approach_Red* node.



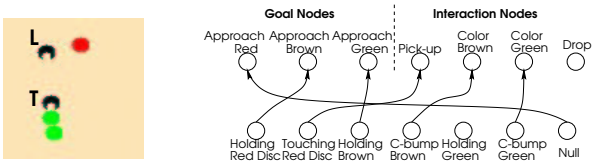
The learner (L) following the teacher (T) has yet to see any events.



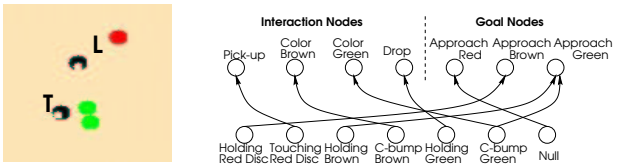
The teacher touches and picks up the red disc. The learner increases the weight on the connections from *null* to the goal node that will later learn to approach red. It also associates touching a red disc with picking it up in the Interaction Network.



The teacher bumps into a brown disc with a disc it is holding. The learner increases the weight on the connection from *Holding_Red_Disc* to the goal node that will eventually learn to approach brown discs. The learner Interaction Network associates the environmental state with coloring the held disc brown.

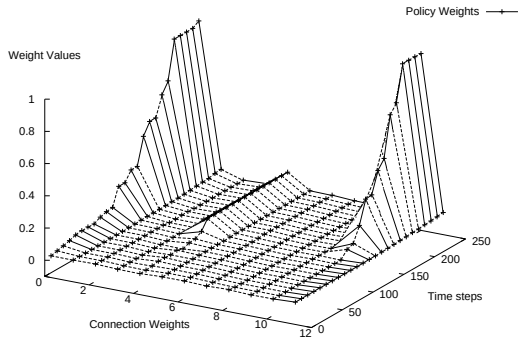


The teacher bumps into a green disc with the disc it is holding, and colors it green. The learner increases the weight on the connection from *Holding_Brown* to the goal node that will later learn to approach green. The learner Interaction Network associates the carry bump green state with coloring the held disc green.



The teacher drops the green disc. The learner associates the state of holding a green disc with dropping the disc.

Figure 7: Screen shots (left) and diagrams of the learner's fully connected Sequence and Interaction Networks (right). Only connections with weights > 0 are shown. The detection nodes receive activation from the visual and touch sensors directly via the detection network. The Interaction nodes are output nodes, causing the agent to perform the listed action, and the goal nodes directly cause the agent to achieve the listed goals by the second order connections in the policy network, as discussed in Section 2.



0: BAL → TL 4: BAL → MF 8: BAL → TR
1: BDL → TL 5: BDL → MF 9: BDL → TR
2: BAR → TL 6: BAR → MF 10: BAR → TR
3: BDR → TL 7: BDR → MF 11: BDR → TR

B: Brown; A: Angle; D: Distance; L: Left; R: Right; M: Move; F:Forward

Figure 8: Plot of the weight values on connections from brown disc sensors and the goal node that eventually controls approaching brown, to the movement output nodes (from 0 to 250 learning time steps).

Figure 10 shows three different walls built by the agents. The left wall (case A) was built by the teacher using the hand-coded network. The middle wall (case B) was built by the teacher and the learner together. After observing the teacher place a disc on the wall, we turned off the follow-the-teacher and the sequence learning in the learner, and turned on it's Policy Network. The learner quickly learned to approach the correct discs and then aided the parent. The right wall (case C) was built entirely by the learner. After the learner's sequence learning was turned off, the teacher was removed and the learner completed the wall on its own. Each case (A, B, and C) was run 10 times and the length of time taken to build the wall was recorded. The average time for case A (teacher only) was 10152.5 time steps. The average time for case B (both) was 6672 time steps and the average time for case C (learner only) was 15969.6 time steps.

6. Discussion and Future Work

The experiments show that a learning agent can learn to perform a task that requires the achievement of sequences of goals. The architecture and algorithm have two major advantages: rapid learning, and any-order learning.

The learning agent is able to learn the order of the steps in a sequence by observing the teaching agent make a *single pass* through the sequence. This is in stark contrast with gradient descent methods, which would require multiple exposures to the input. This is important when the sequence is required for survival or when there

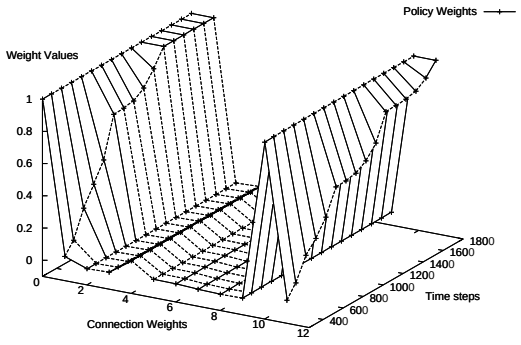


Figure 9: Plot of the weights from brown disc sensors and the goal node that eventually controls approaching brown, to the movement output nodes (from 250 to 1800 learning time steps). The weights are arranged as in figure 8.

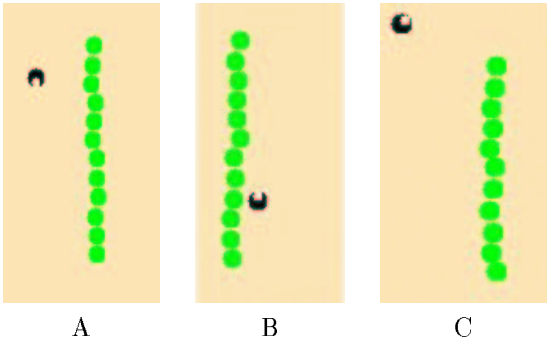


Figure 10: Completed walls built by the teacher alone (A), the teacher and the learner (B), and the learner alone (C).

is little opportunity for teaching.

Another advantage is that the learning agent *does not need to observe the sequence in the order in which it is to be performed.* The learner associates each environmental state with a goal, so it does not need to know what has gone before. This is a distinct advantage in more chaotic environments when the learner may be interrupted while attending to the teacher. This could occur when it becomes hungry or thirsty and stops for food or drink, or when it sees a predator. If the learner misses some part of a sequence, it can pick it up later by observing that part out of order.

One disadvantage of the system is that the learning agents do not have the ability to invent or modify the sequences they observe. A learning agent performs the sequence as it is demonstrated by the teacher without modification, even if the task could be performed more efficiently (faster, or with fewer steps) with some other sequence.

6.1 Failure and Boundary Conditions

The system has a number of fixed parameters, such as the weights of the detection networks. This situation can lead to failures to which the agents are unable to adapt. One example of this arises directly from the limited detection abilities of the detection nodes. The detection nodes were wired to detect particular simple states, such as agent is holding a red disc, but do not represent conjunctions of these states. For example, if, during the wall building sequence, an agent that is carrying a red disc to the brown disc accidentally carry-bumps into a green disc, the Interaction Network will cause the agent to color the red disc green and drop it in the wrong place. This occurs because the detection nodes could not represent that the true condition for dropping the disc should have been that both carry-bump green and holding brown are true. If the detection networks were not fixed, the agent could learn to adjust to detection weights to account to this error.

Another failure condition occurred due to the algorithm used for building the wall. Because the straightness of the wall depends on an agent moving toward a green disc from the brown disc. If the agent is wavers from that line for any reason (e.g. to avoid a collision with another agent), the agent will then end up approaching the green disc from the wrong direction, and dropping the disc it is carrying on the side of the wall rather than the end.

6.2 Future Work

In the future, we intend to both enhance the representational power of the detection nodes so that they can represent conjunctions. and develop mechanisms to automatically learn or evolve the weights in the Detection Network. We also hope to develop additional learning algorithms with which the agents can correctly build the Model Networks on their own by noticing correlations between their own state and that of other agents.

7. Related Work

We know of very little work in the area of learning sequences of goals or other higher level behaviors built on top of actions. A robotic soccer agent (Stone and Veloso, 1998) learned low-level behaviors using actions, and then used those behaviors in learning higher-level behaviors. It differs from our work in that it: a) uses supervised learning rather than imitation, b) learning in each layer is performed as a separate experiment, and c) the low-level behaviors to use were explicitly given to the learning agents.

Imitation learning usually takes the form of imitation of movements or action sequences, but they do not identify and imitate the goals of the teaching agent. (Gaussier et al., 1998, Dautenhahn, 1995) both use imitation to learn trajectories for mobile robots, where what

is learned is to follow a specific path, such as a square or a star. In (Hayes and Demiris, 1994) a robot learns to navigate a maze by associating actions with different locations in the maze, and (Kuniyoshi et al., 1994) describes a robot that learns sequences of actions by imitation of a human performing an assembly task.

(Morén, 1998) and (Sun and Peterson, 1998) both use techniques built on top of a Q-learning system (Sutton and Barto, 1998) to learn sequences of actions. Morén clusters commonly occurring sequences of actions into chunks to speed up learning. Sun and Peterson layer a rule extracting mechanism on top of a Q-learning mechanism, to extract both procedural and declarative knowledge. They extract rules matching states to actions from the Q-learning layer below.

8. Conclusion

In this paper we have presented an algorithm for learning sequences of goals by observation and imitation, as part of a robust animat control architecture. The agents take a wait-and-see approach to solve the goal identification problem to learn the sequence of goals, while using existing goal learning structures in the architecture to efficiently learn how to achieve those goals. Simulations show that the sequence learning in VI-MAXSON enables agents to quickly (in one pass) acquire a skill level rivaling that of their teachers.

Acknowledgements

Thanks to Rebecca Hwa for her comments on several drafts. This work was supported in part by Intel University Research Program grant to the second author.

References

Bakker, P. and Kuniyoshi, Y. (1996). Robot see, robot do: An overview of robot imitation. In *AISB Workshop on Learning in Robots and Animals.*

Cecconi, F. and Parisi, D. (1993). Neural networks with motivational units. In *From animals to animats: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 346–355.

Crabbe, F. L. and Dyer, M. G. (1999a). MAXSON: max-based second-order neural network reinforcement learner for mobile agents in continuous environments. Technical Report CSD-900009, UCLA.

Crabbe, F. L. and Dyer, M. G. (1999b). Second-order networks for wall-building agents. In *Proceedings of the International Joint Conference on Neural Networks*, Washington D.C.

Crabbe, F. L. and Dyer, M. G. (1999c). Vicarious learning in mobile neurally controlled agents: The V-MAXSON architecture. In *Proceedings of the International Conference on Artificial Neural Networks.*

Dautenhahn, K. (1995). Getting to know each other – artificial social intelligence for autonomous robots. *Robotics and autonomous systems*, 16:333–356.

Gaussier, P., Moga, S., and Quoy, M. (1998). From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. *Applied Artificial Intelligence (AAI)*, 12.

Giles, C. L. and Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, 26(23):4972–4978.

Hayes, G. and Demiris, J. (1994). A robot controller using learning by imitation. In *Proceedings of the Second International Symposium on Intelligent Robotic Systems.*

Kuniyoshi, Y., Inaba, M., and Inoue, H. (1994). Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transations on Robotics and Automation*, 10(6):799–822.

Morén, J. (1998). Dynamic action sequences in reinforcement learning. In *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior.*

Rumelhart, D. E. and McClelland, J. L., (Eds.) (1986). *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*, volume I, II. MIT Press, Cambridge, MA.

Stone, P. and Veloso, M. (1998). A layered approch to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence (AAI)*, 12.

Sun, R. and Peterson, T. (1998). A subsymbolic+symbolic model for learning sequenction navigation. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior.*

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning.* MIT Press, Cambridge, MA.

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

Ulbricht, C. (1996). Handling time-warped sequences with neural networks. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.*

Zaki, M. J., Lesh, N., and Ogihara, M. (1999). Planmine: Predicting plan failures using sequence mining. *Artificial Intelligence Review*, 13.