Sample Iteration

Let's now look at a sample action-selection iteration of an evolved agent. In this example, we'll look at an herbivore that has evolved during the simulation. While not perfect, this particular agent survived in the environment for over 300 time steps. It was able to do this by avoiding predators (carnivores) as well as finding and eating plants. Figure 7.6 shows the neural network for this agent.

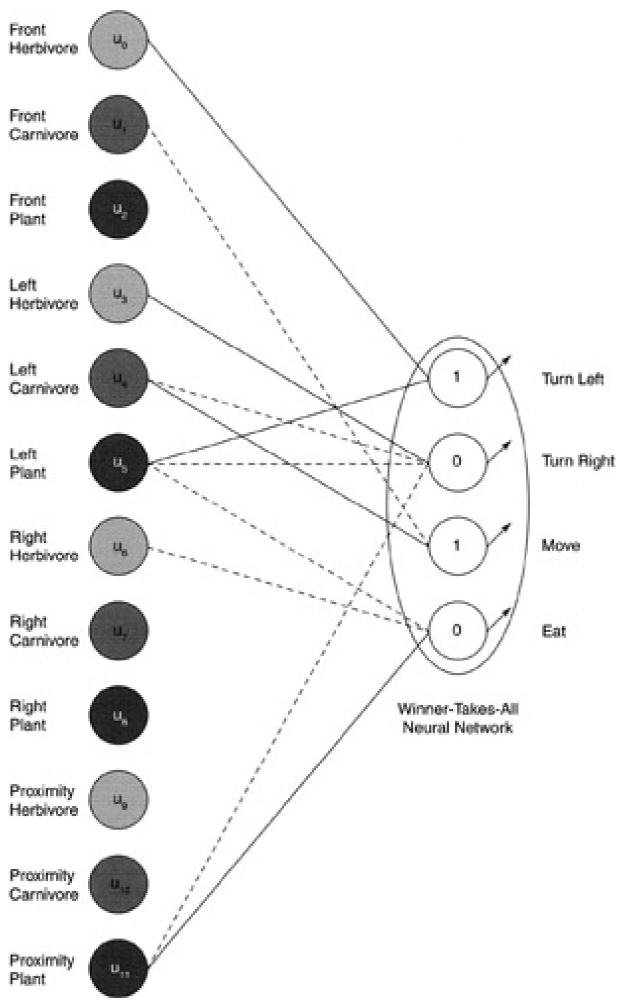

Figure 7.6: Neural network for evolved herbivore.

Solid lines in the neural network are excitory connections, while dashed lines are inhibitory. Within the output cells are the biases that are applied to each output cell as it is activated. From Figure 7.6, we can see that an excitory connection exists for the *eat* action when a plant is in proximity (a plant can only be eaten when it is in proximity of the agent). Also of interest is the inhibitory connection for the *move* action when a carnivore is in front. This is another beneficial action for the survival of the herbivore.

An agent's actions are not formed by a single connection. Instead, the action with the largest value is permitted to fire based upon the combination of the sensor inputs. Let's look at a couple of iterations of the herbivore described by the neural network in Figure 7.6.

Recall from Equation 7.1 that we multiply the weights vector (for the particular action) by the inputs vector and then add the bias.

In our first epoch, our herbivore is presented with the scene as shown in Figure 7.7. The different zones are shaded to illustrate them (recall Figure 7.4). In this scene, the 'X' denotes the location of the herbivore (its reference point to the scene). A plant is located within *proximity* and a carnivore is located in *front.*

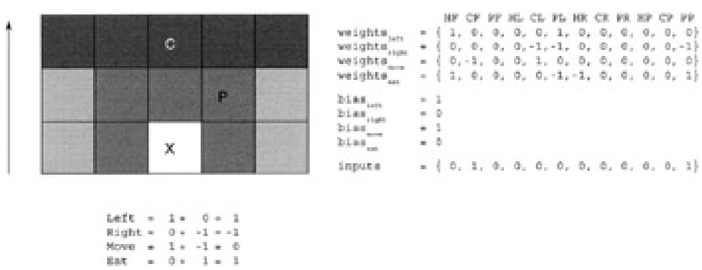

Figure 7.7: Herbivore at time t_0.

The first step is assessing the scene. We count the number of objects of each type in each of the four zones. As shown in Figure 7.7, the weights and inputs are labeled with type/zone (HF is herbivore-*front*, CF is carnivore-*front,* PP is *plant-proximity,* etc.). In this example, our input's vector has nonzero values in only two of the vector elements. As shown in the scene, these are a carnivore in the *front* zone and a plant in *proximity.*

To evaluate the function to take, we multiply the input vector by the weights vector for the defined action and then add in the respective bias. This is shown below in Figure 7.7. Per our action-selection algorithm, we take the action with the largest resulting value. In the case of a tie, as is shown in Figure 7.7, we take the largest value that appeared last. In this case, the *eat* action is performed (a desirable action given the current scene).

Once the plant is consumed, it disappears from the scene. The herbivore is then left with the scene as shown in Figure 7.8.

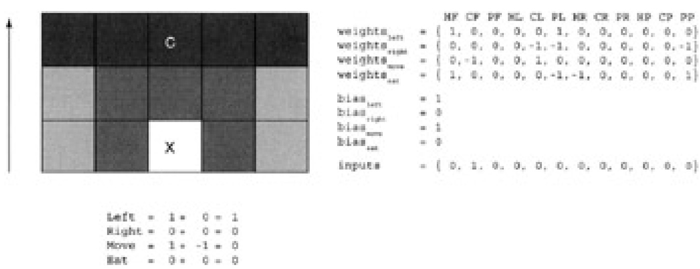

Figure 7.8: Herbivore at time T_1.

The scene is assessed once again, and as shown, the plant no longer exists, but the carnivore remains. This is illustrated by the inputs (changes are shown in bold from the prior iteration). We again compute the output cells of the neural network by multiplying the input vector times the respective weights vector. In this case, the largest value is associated with the *left* action. Given the current scene, this is a reasonable action to take.

Finally, in Figure 7.9, we see the final iteration of the herbivore. Note that the agent's view has changed because it changed direction in the last iteration. With the change in the scene, the inputs have now changed as well. Instead of a carnivore in the *front* zone, it has moved to the *right* zone due to our new view of the scene.

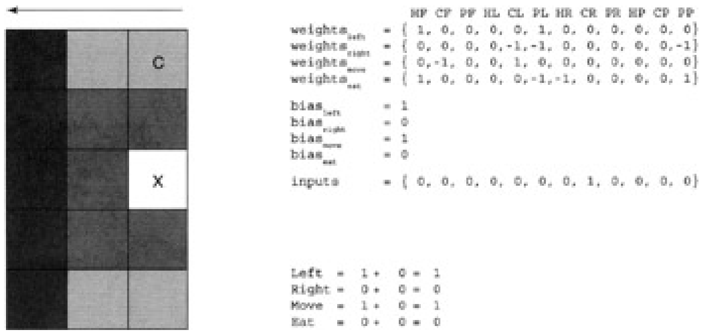

Figure 7.9: Herbivore at time T_2.

Computing the output cells once again results in the move action being the largest, but more importantly, last in this case. The herbivore's behavior allowed it to find food and eat it, and then avoid a carnivore within its field of view. From this short demonstration of the herbivore's neural network, it is not surprising that it was able to navigate and survive in the environment for a long period of time.

On the CD The source for the Alife simulation can be found on the accompanying CD-ROM at ./software/ch7.

Sample Results

We'll now look at a few summaries of the simulation in action. The operation of the simulation will also be discussed including the available command-line options that are available.

The simulation can be run by simply executing the application with no options, such as:

```
./sim
```

This will run the simulation given the parameters defined in the header file `common.h`. The defaults within `common.h` include 36 agents (18 herbivores and 18 carnivores) and 35 plants within a 30x30 grid. The maximum number of simulation steps is set to 1 million. Figure 7.10 shows the maximum age reached for each of the two species.

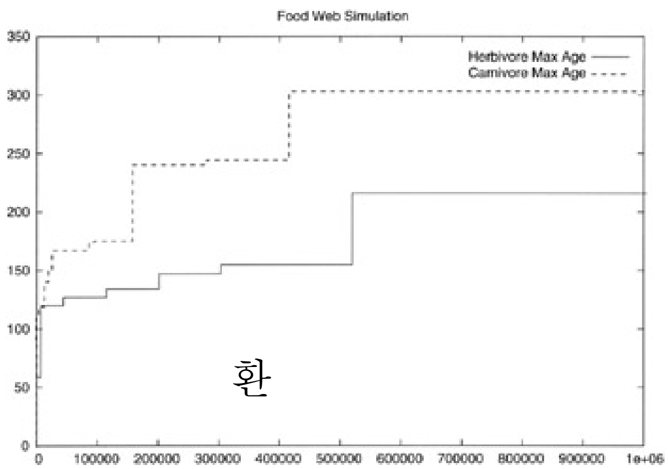

Figure 7.10: Age progression in a sample simulation.

It's interesting to note the trend of increasing age in the agents. When the carnivore species finds an interesting strategy that increases its longevity, shortly thereafter, herbivores find another strategy that gives them the ability to live a little longer. In some ways, the species compete with one another. When a species evolves an interesting strategy, the other species must evolve to counteract this new strategy.

Once a run is complete, the two best agents of the species are saved off into a file called `agents.dat`. These agents can then be pitted against one another in a simulation called *playback.* This mode doesn't seed the population with random members, but instead starts with the best members from the last run. This can be performed using the following command:

```
./sim -prn
```

The 'p' argument defines that we want to run in playback mode. The 'r' specifies that we wish to save off the run-time trend information, and 'n' defines that no reproduction is permitted. The trend data stored in *playback* mode includes agent birth and death counts (for both species).

Using agents evolved in the prior run, Figure 7.11 shows a plot of the runtime trend data that was created.

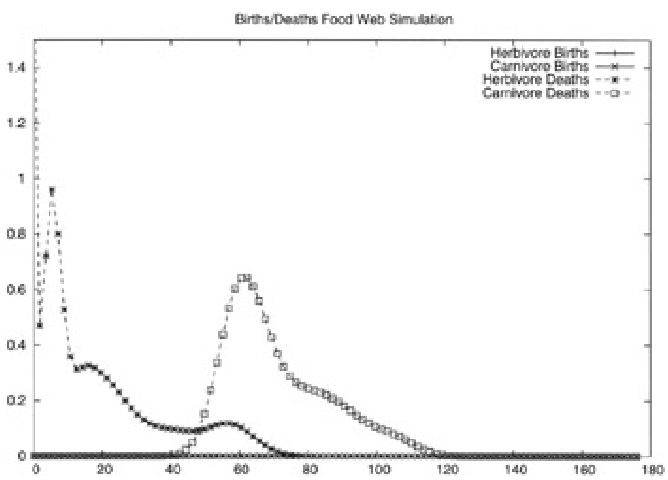

Figure 7.11: Run-time trend data from a playback simulation. Herbivore and carnivore births, while represented in the graph, are not visible due to the graph scaling and frequency of the herbivore and carnivore deaths.

As we've defined that no reproduction may occur, this simulation shows no births occurring, only deaths. When the playback simulation has begun, the landscape is initialized with herbivores and carnivores. It's clear from the plot that the carnivores have a field day with the abundance of herbivores that are available to consume. Once this winds down, carnivore deaths begin a steep increase as the landscape has been stripped of prey and starvation sets in. With the loss of food for the carnivores, their demise is certain.

The simulation provides a number of options. These are shown in Table 7.1.

Table 7.1: Command-line Options for the Simulation.

Option	***Description***
-h	Command-line help
-p	Playback mode (read agents.dat)
-r	Save run-time trend data (runtime.dat)
-g	Don't regrow plants
-c	Convert carnivores to plants when they die
-n	No reproduction is permitted
-s	Manual step (carriage-return required)

One interesting scenario is provided by the following parameters:

```
./sim -prncg
```

This provides a circular food web in which carnivores hunt and eat herbivores, but conversely, once carnivores die, they become food available to herbivores.

Interesting Strategies

While this simulation is simple and agents are provided with a minimal number of input sensors and available actions, very interesting behaviors can result.

One interesting herbivore strategy entailed herding. An herbivore would follow another herbivore if it were in front. The analogy could be strength in numbers—as long as you're not the herbivore in front. Carnivores found numerous interesting strategies one of which was finding plants and then waiting for herbivores to wander by. This strategy was successful but short-lived as herbivores finally evolved the ability to avoid carnivores, even if plants were in the vicinity.