# Chapter 5

# Small Neural Nets : Symbots



In this chapter, we will learn to program a very simple type of neural net with evolutionary algorithms. These neural nets will be control systems for virtual autonomous robots called *symbots*, an artificial life system developed by Kurt vonRoeschlaub, John Walker, and Dan Ashlock. Unlike the neural nets we studied in Chapter 1, these neural nets will have no internal neurons at first, just inputs and outputs.

The symbots consist of two wheels and two sensors which report the strength of a "field" at their position. The sensors can be thought of as eyes, noses, Geiger counters, etc.; the field could be light intensity, chemical concentration, the smell of prey, whatever you want to model. The symbot's neural net takes the sensor output and transforms it into wheel drive strengths. The wheels then cause the symbot to advance (based on the sum of drive strengths) and turn (based on the difference of drive strengths).

In the course of the chapter, we will introduce a new theoretical concept, the *lexical product* of fitness functions, which is used to combine two fitness functions in a fashion that allows one to act as a helper for the other. The lexical product is of particular utility when the fitness function being maximized gives an initial value of zero for almost all creatures.

## 5.1 Basic Symbot Description

Symbots live on the unit square: the square with corners $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$. A basic symbot, shown in Figure 5.1, is defined by a radius $R$, an angular displacement of sensors from the symbot's centerline $\theta$, and 5 real parameters that form the symbot's control net. These parameters are the connection weights, in the sense of neural nets, from the right and left sensors to the right and left wheels and the *idle speed*. The idle speed is the symbot's speed when it is receiving no stimulation from its sensors. We will denote these five real parameters as $ll$, $lr$, $rl$, $rr$, and $s$. (The two letter names have as their first character the

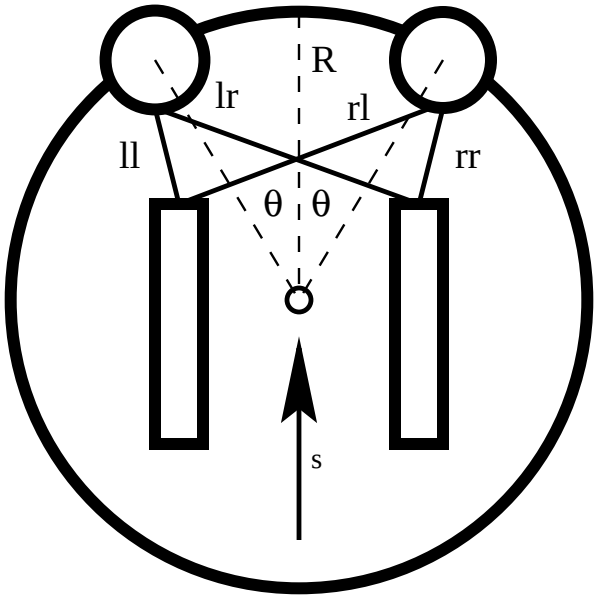


Figure 5.1: Basic symbot layout

sensor ((l)eft or (r)ight) with which they are associated and as the second the wheel with which they are associated.)

The symbot's neural net uses the sensors as the input neurons and the wheels as the output neurons. The sensors report the strength of some field; those sensor strengths are multiplied by the appropriate connection strengths and then summed to find out how hard (and in which direction) the wheels push the symbot. The symbot's motion is simulated by iterating the algorithm given in Figure 5.2 called the *basic symbot motion loop*. The code given in this loop is an Euler's method integration of a kinematic motion model. The step size of the integration is controlled by the constant $C_s$.

The basic symbot motion loop computes, for one time slice, the symbot's response to the inputs felt at its left and right sensors. This response consists of updating the symbot's position $(x, y)$ and heading $\tau$. The loop in Figure 5.2 is completely inertialess and contains a constant $C_s$ that is used to scale the symbot's turning and motion.

The basic intuition is as follows. Compute the current position of both sensors and get the strength of the field at their position with the function $f(x, y)$. Multiply by the connection weights of sensors to wheels, obtaining the drive of each wheel. The forward motion of the symbot is the sum of the wheel drives plus the idle speed. The change in heading is the difference of the drive of the left and right wheel. Both the motion and change in heading are truncated if they are too large. (Notice that the symbot's position $(x, y)$ is the *center* of its circular body.)

The function $f(x, y)$ reports the field strength at position $(x, y)$. This function is a

Begin

| | |
|---|---|
| $x_1 := x + R \cdot \cos(\tau + \theta);$ | //left sensor position |
| $y_1 := y + R \cdot \sin(\tau + \theta);$ | |
| $x_2 := x + R \cdot \cos(\tau - \theta);$ | //right sensor position |
| $y_2 := y + R \cdot \sin(\tau - \theta);$ | |
| $d_l := f(x_1, y_1) \cdot ll + f(x_2, y_2) \cdot rl;$ | //find wheel |
| $d_r := f(x_2, y_2) \cdot rr + f(x_1, y_1) \cdot lr;$ | //drive strengths |
| $ds := C_s \cdot (dl + dr + s \cdot R/2);$ | //change in position |
| If $\|ds\| > R/2$ then $ds := sgn(ds) \cdot R/2;$ | //truncate |
| $d\tau := 1.6 \cdot C_s/R \cdot (dr - dl);$ | //change in heading |
| If $\|d\tau\| > \pi/3$ then $d\tau := sgn(d\tau) \cdot \pi/3;$ | //truncate |
| $x := x + ds \cdot \cos(\tau);$ | //update position |
| $y := y + ds \cdot \sin(\tau);$ | |
| $\tau := \tau + d\tau;$ | //update heading |

end;

Figure 5.2: Basic symbot motion loop for symbot at position $(x, y)$ with heading $\tau$ (The function $f(x, y)$ reports the fields strength; $sgn(x)$ is $-1, 0, 1$ as $x$ is negative, zero, or positive; $C_s$ controls the step size of integration.)

primary part of the description of the symbot's world and the definition of its task. If, for example, the symbot is to be a scavenger, the function $f$ might be a diffusion process, possibly modeled with a cellular automata, which spreads the smell of randomly placed bits of food. If the symbot is a photovore which consumes light sources, the field strength would be computed from the standard inverse square law. If the symbot's task was to follow a line on the floor, the field strength might simply be a binary function returning the color of the floor, 0 for off the line or 1 for on it.

We said that the symbot lives on the unit square. What does this mean? What if it tries to wander off? There are many ways to deal with this problem; here are four suggestions. In a *wall-less world*, the symbot lives in the Cartesian plane and we simply restrict interesting objects to the unit square, e.g., lines to be followed, inverse square law sources, or food. In a *lethal wall world*, we simply end the symbot's fitness evaluation when it wanders off of the unit square. (In a lethal wall world, the fitness function should be a nondecreasing function of the time spent in the world. If this is not the case, evolution may select for hitting the wall.) In a *reflective world*, the walls are perfect reflectors and we simply modify the symbot's heading as appropriate to make angle of incidence equal to angle of reflection. In a *stopping world*, we set the symbot's forward motion $ds$ to 0 for any move which would take it beyond the boundaries of the unit square. This does not necessarily stop the symbot,

since it still updates its heading and can turn away from the wall.

The symbot's world is a highly idealized one and this requires some care be taken. Suppose we are generating the field strength from inverse square law sources. If we had a single source at $(a, b)$ then the pure inverse square law says that the field emitted by that source would be

$$f(x, y) = \frac{C_f}{(x-a)^2 + (y-b)^2}, \tag{5.1}$$

where $C_f$ is a constant that gives the intensity of the source. The problem with this is that a symbot that has a sensor near $(a, b)$ experiences an awesome signal and, as a result, may suddenly shoot off at a great speed or spin through an angle so large, relative to the numerical precision of the machine you are using, that it is essentially a random angle. To avoid this we will assume the inverse square law source is not a point source, but rather has a radius $r_c$ with a constant field strength inside the source equal to the value the inverse square law would give at the boundary of the source. Call such inverse square law sources *truncated inverse square law sources*. The equation for a truncated inverse square law source with radius $r_c$ at position $(a, b)$ is given by:

$$f(x, y) = \begin{cases} \frac{C_f}{(x-a)^2+(y-b)^2} & (x-a)^2 + (y-b)^2 \geq r_c^2 \\ \\ \frac{C_f}{r_c^2} & (x-a)^2 + (y-b)^2 < r_c^2 \end{cases} \tag{5.2}$$

The following experiment implements basic symbot code without imposing the additional complexity of evolution. Later, it will serve as an analysis tool for examining the behavior of evolved symbots. Keeping this in mind, you may wish to pay above average attention to the user interface as you will use this code with symbots you evolve later in the chapter.

**Experiment 5.1** *Write or obtain software to implement the basic symbot motion loop. You should have a data structure for symbots that allows the specification of radius, $R$, angular displacement of sensors from the axis of symmetry, $\theta$, the 4 connection weights: $ll$, $lr$, $rl$, $rr$, and the idle speed, $s$. Use the basic symbot motion loop to study the behavior of a single symbot placed in several locations and orientations on the unit square. Define field strength using a truncated inverse square law source with radius $r_c^2 = 0.001$ at position $(0.5, 0.5)$ with $C_f = 0.1$ and $C_s = 0.001$. Test each of the following symbot parameters and report on their behavior:*

| *Symbot* | $R$ | $\theta$ | $ll$ | $lr$ | $rl$ | $rr$ | *idle* |
|---|---|---|---|---|---|---|---|
| *1* | *0.05* | $\pi/4$ | *-0.5* | *0.7* | *0.7* | *-0.5* | *0.3* |
| *2* | *0.05* | $\pi/4$ | *-0.2* | *1* | *1* | *-0.2* | *0.6* |
| *3* | *0.05* | $\pi/4$ | *-0.5* | *0.5* | *0.7* | *-0.7* | *0.4* |
| *4* | *0.1* | $\pi/4$ | *1* | *0* | *0* | *1* | *0.3* |
| *5* | *0.05* | $\pi/2$ | *-0.5* | *0.7* | *0.7* | *-0.5* | *0.3* |

*Characterize how each of these symbots behaves for at least 4 initial position/orientation pairs. Use a wall-less world. It is a good idea to write your software so that you can read and write symbot descriptions from files, as you will need this capability later.*

With Experiment 5.1 in hand, we can go ahead and evolve symbots. For the rest of this section, we will set the symbots the task of finding truncated inverse square law sources. We say that a symbot has *found* a source if the distance from the source to the symbot's center is less than the symbot's radius. There are the usual laundry list of issues (model of evolution, variation operators, etc.), but the most vexing problem for symbots is the fitness function. It is nice if the fitness function is an nondecreasing function of time; this leaves open the possibility of using lethal walls (see Problem 5.1). We also need the fitness function to drive the symbot toward the desired behavior. In the next few experiments, we will use evolution to train symbots to find a single truncated inverse square law source at $(0.5, 0.5)$. (If you have limited computational capacity, you can reduce population size or number of runs in the following experiments.)

**Experiment 5.2** *Assume in this experiment that we are in the same world as in Experiment 5.1. Fix symbot radius at $R = 0.05$ and sensor angular displacement at $\theta = \pi/4$. Build an evolutionary algorithm with the gene of the symbot being the 5 numbers $ll$, $lr$, $rl$, $rr$, and $s$, treated as indivisible reals. Use tournament selection with tournament size 4, one point crossover, and single point real mutation with mutation size* 0.1*. Evaluate fitness as follows. Generate 3 random initial positions and headings that will be used for all the symbots. For each starting position and heading, run the symbots forward for 1000 iterations of the basic symbot motion loop. The fitness is the sum of $f(x, y)$ across all iterations where $(x, y)$ is the symbot's position. Evolve 30 populations of 60 symbots for 30 generations.*

*Report the average and maximum fitness and the standard deviation of the average fitness. Save the best design for a symbot from the final generation for each of the 30 runs. Characterize the behavior of the most fit symbot in the last generation of each run. (This is not as hard as it sounds, because the behaviors will fall into groups.)*

*Define* finding the source *to be the condition that exists when the distance from the symbot's nominal position $(x, y)$ to the source is at most the symbot's radius $R$. Did the symbots do a good job of finding the source? Did more than one technique of finding the source arise? Do some of the evolved behaviors get a high fitness* without *finding the source?*

*Are some of the behaviors physically implausible, e.g., extremely high speed spin? Explain why the best and average fitness go up and down over generations in spite of our using an elitist model of evolution.*

Some of the behaviors that can arise in Experiment 5.2 do not actually find the source. In Figures 5.3, 5.4, and 5.5 you can see the motion traces of symbots from our version of Experiment 5.2. If we wish to find the source, as opposed to spend lots of time fairly near

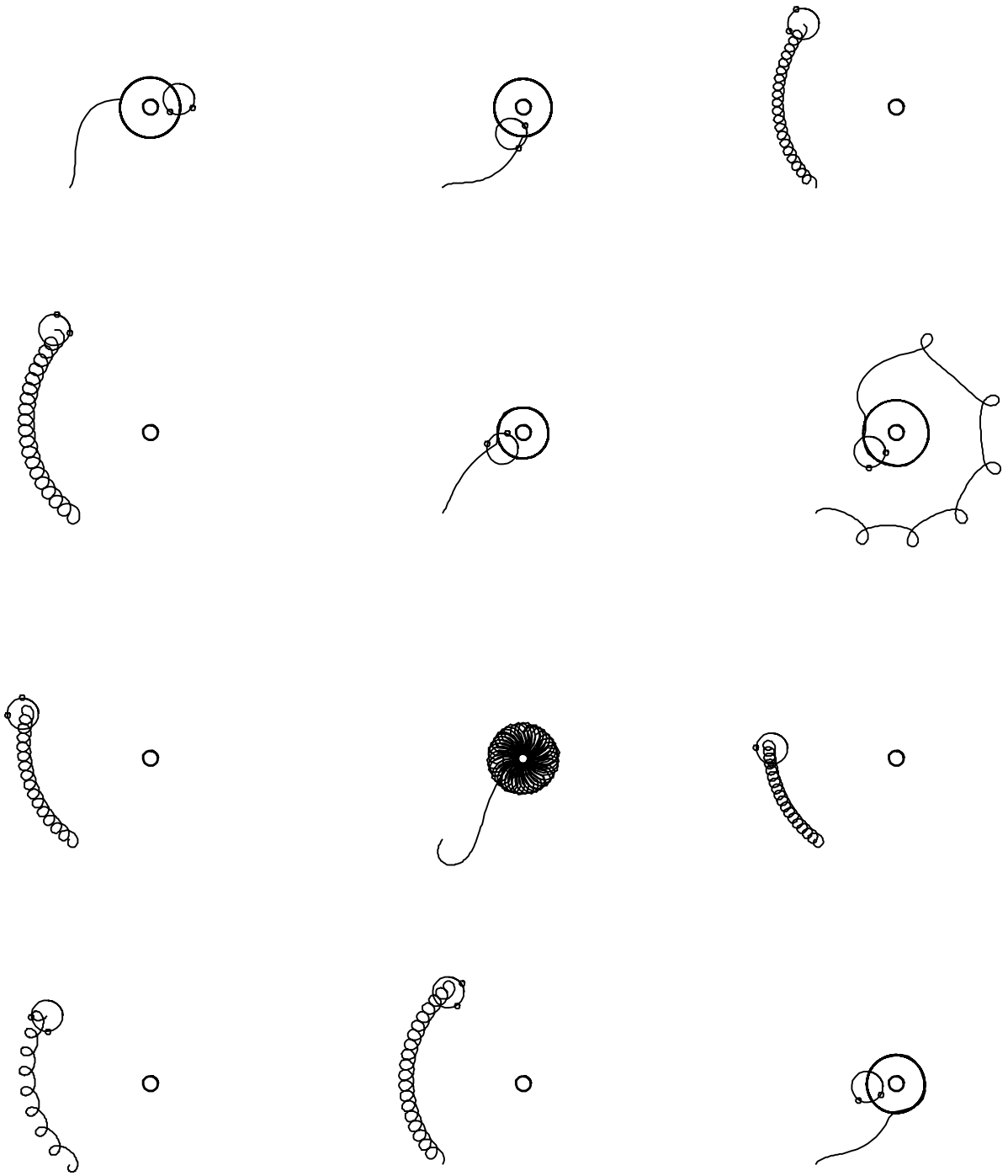

Figure 5.3: Plots for the most fit symbot at the end of the run, runs 1-12

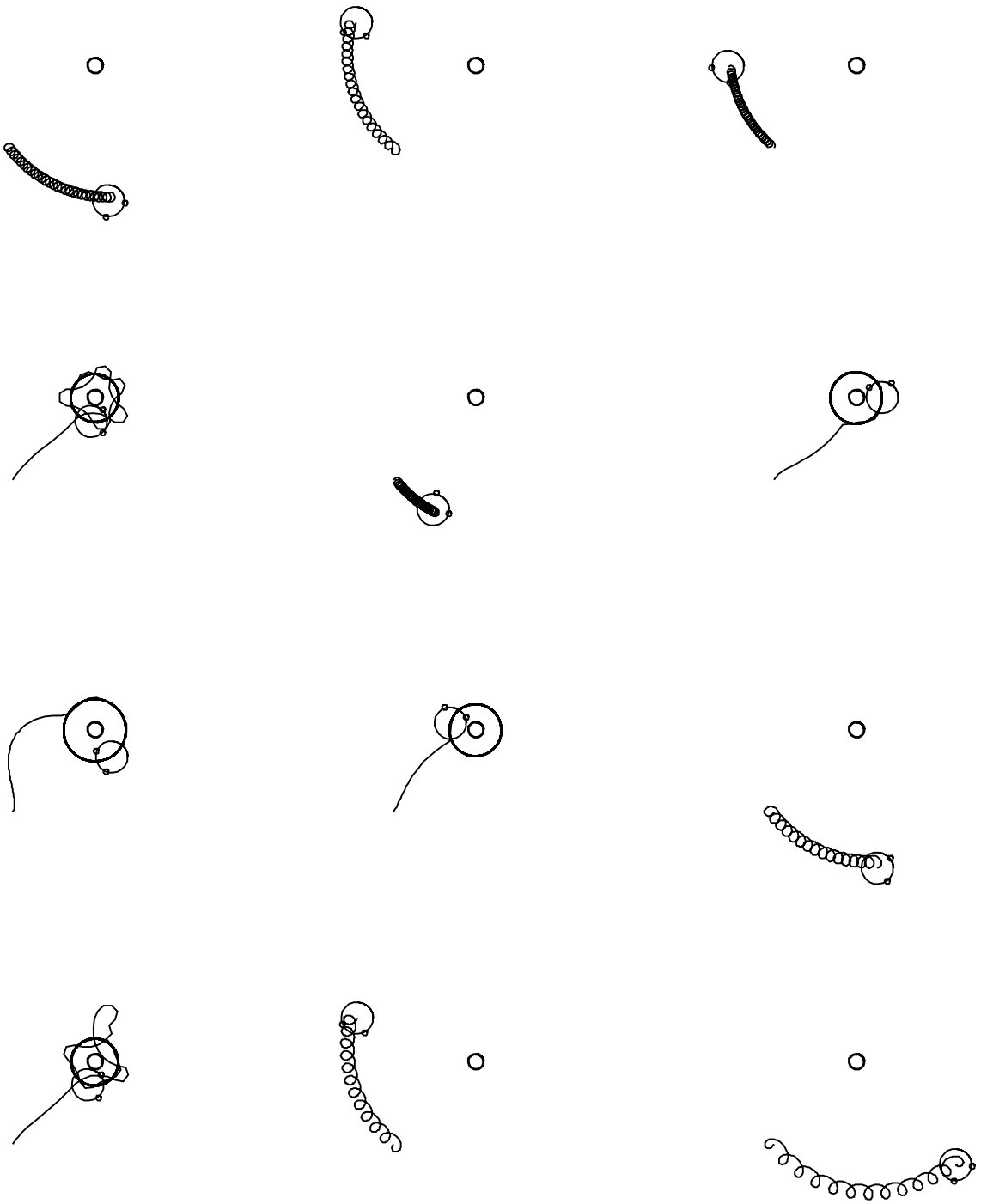

Figure 5.4: Plots for the most fit symbot at the end of the run, runs 13-24

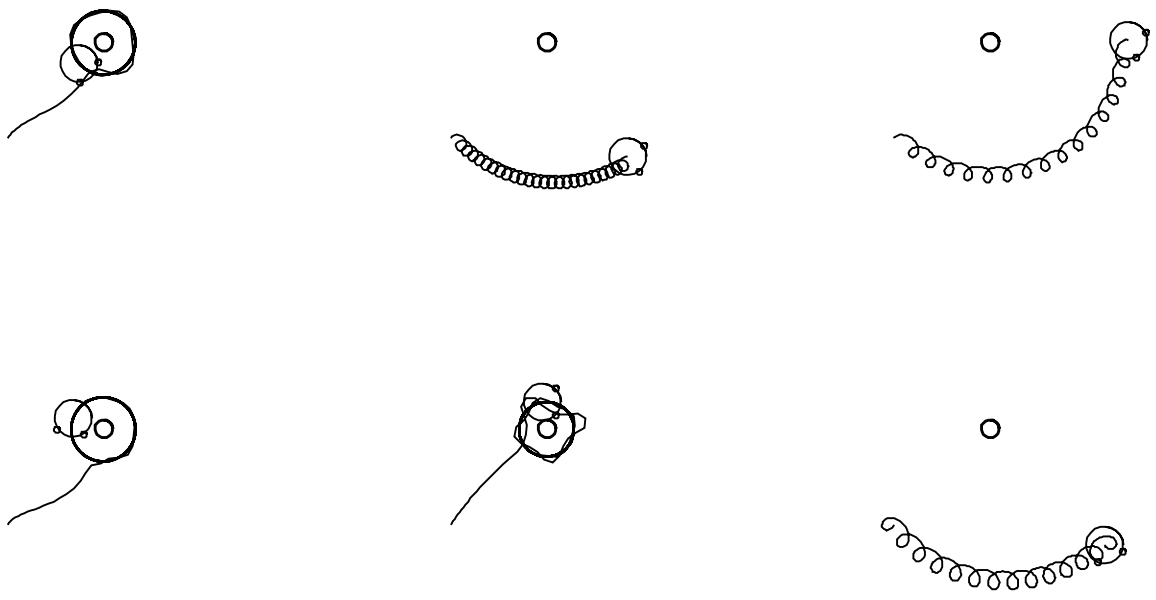

Figure 5.5: Plots for the most fit symbot at the end of the run, runs 25-30

it, it might be good to tweak the fitness function by giving a bonus fitness for finding the source. There are a number of ways to do this.

**Experiment 5.3** *Write a short subroutine that computes when the symbot has found the source at* $(0.5, 0.5)$*, and then modify Experiment 5.2 by replacing the fitness function with a function which counts the number of iterations it took the symbot to find the target for the first time. Minimize this fitness function. Report the same results as in Experiment 5.2.*

*The results may be a bit surprising. Run as many populations as you can and examine the symbot behaviors that appear.*

The fitness function in Experiment 5.3 is the one we really want, if the symbot's mission is to find the source. However, if this function acts in your experiments as it did in ours, there is a serious problem. The mode fitness of a random creature is zero, and unless the population size is extremely large, it is easy to have *all* the fitnesses in the initial population equal to zero for most test cases. How can we fix this? There are a couple of things we can try.

**Experiment 5.4** *Redo Experiment 5.3, but in your initial population generate 3 rather than 5 random numbers per symbot, taking* $ll = rr$ *and* $lr = rl$*. The effect of this is to make the initial symbots symmetric. Do 2 sets of runs:*

*(i) Runs where the condition* $ll = rr$ *and* $lr = rl$ *is maintained under mutation - if one connection weight changes, change the other.*

*(ii) Runs in which the evolutionary operations are allowed to change all 5 parameters independently.*

*Do 100 runs. Does evolution tend to preserve symmetry? Does imposed symmetry help? How often do we actually get a symbot that reliably finds the source?*

The key to Experiment 5.4 is restriction of the space the evolutionary algorithm must search. From other work with symbots, it is known that there are very good solutions to the current symbot task which have symmetric connection weights. More importantly, the probability of a symmetric symbot being a good solution is higher than that probability for an asymmetric symbot. The symmetry restriction makes the problem easier to solve. Keep in mind that Experiment 5.4 doesn't just demonstrate the value of symmetry but also checks the difference between a 3 parameter model (i) and a 5 parameter model with a few restrictions on the initial conditions (ii).

The question remains: can we solve the original 5 parameter problem more efficiently without cooking the initial values? One technique for doing so requires that we introduce a new type of fitness function. The fitness functions we have used until now have been maps from the set of genes to an ordered set like the real numbers.

**Definition 5.1** *The* **lexical product** *of fitness functions,* $f$ *and* $g$*, denoted* $f$ *lex* $g$*, is a fitness function that calls a gene* $x$ *more fit than a gene* $y$ *if* $f(x) > f(y)$ *or* $f(x) = f(y)$ *and* $g(x) > g(y)$*. In essence* $g$ *is used only to break ties in* $f$*. We say that* $f$ *is the* dominant function*. (This terminology helps us remember which function in a lexical product is the tie-breaker.)*

With the notion of lexical product in hand, we can do Experiment 5.2 a different way.

**Experiment 5.5** *Modifying the fitness evaluation techniques used in Experiments 5.2 and 5.3, evolve symbots with a fitness function that is the lexical product of (i) the number of iterations in which a symbot has found the source with (ii) the sum of the field strength at* $(x, y)$ *in all iterations. Let the number of iterations in which the symbot has found the source be the dominant function. Do 30 runs on a population of size 60 for 30 generations and compare to see if using the lexical product gives an improvement on the problem of maximizing the number of iterations in took the symbot to find the target.*

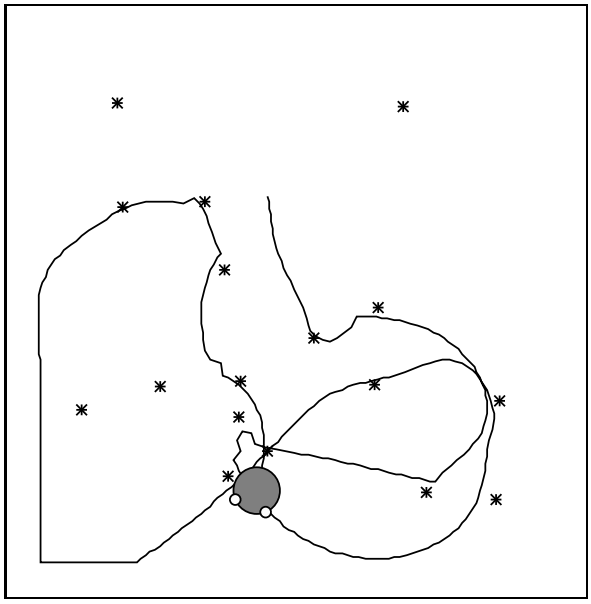

Figure 5.6: A symbot, its path, and sources captured in a $k-5$ run with stopping walls

The motivation for the lexical product of fitness functions is as follows. Imagine a case in which the the fitness function you want to satisfy has a fitness landscape for which almost all random creatures have the same, rotten fitness (so much so that random initial populations tend to be uniformly unfit). When this happens, evolution needs a secondary heuristic or fitness function to be used when the first gives no information.

Maximizing function (ii) from Experiment 5.5, the sum of field strengths over iterations, biases the symbot toward approaching the source. Once the symbots tend to approach the source, the probability that some will actually run it over is much higher, and evolution can proceed to optimize the ability to find the source (function (i)). Notice that the sum-of-field-strength function almost always distinguished between two symbots. With similar symbots, it may do so capriciously, depending on the initial positions and directions selected in a given generation. The quality of being virtually unable to declare two symbots equal makes it an excellent tie breaker. Its capriciousness makes it bad as a sole fitness function, as we saw in Experiment 5.1.

Next, we will change the symbot world. Instead of a single source at a fixed location we will have multiple, randomly placed sources. An example of a symbot trial in such a world is shown in Figure 5.6.

**Experiment 5.6** *Write or obtain software for an evolutionary algorithm with a model of evolution and variation operators as in Experiment 5.2. Assume that we have a world without walls. Implement routines and data structures so that there are k randomly placed sources in the symbot world. When a symbot finds a source, the source should disappear and a new*

*one be placed. In addition, the same random locations for new sources should be used for all the symbots in a given generation to minimize the impact of luck. This will require some nontrivial information management technology. In this experiment let* $k = 5$ *and test two fitness functions, to be maximized:*

*(i) Number of sources found.*

*(ii) Lexical product of the number of sources found with* $\frac{1}{d+1}$ *where* $d$ *is the closest approach the symbot made to a source it did not find.*

*Use populations of 32 symbots for 60 generations, but only do one set of 1500 iterations of the basic symbot motion loop to evaluate fitness (the multiple source environment is less susceptible to effects of capricious initial placement). Run 30 populations with each fitness function. Plot the average and maximum score of each population and the average of these quantities over all the populations for both fitness functions. Did the secondary fitness function help? If you have lots of time, rerun this experiment for other values of* $k$*, especially 1. Be sure to write the software so that it can save the final population of symbots in a generation to a file for later use or examination.*

If possible, it is worth doing graphical displays of the "best" symbots in Experiment 5.6. There are a wide variety of possible behaviors, many of which are amusing and visually appealing: symbots that move forward, symbots that move backward, whirling dervishes, turn-and-advance, random-looking motion, a menagerie of behaviors, etc.

**Experiment 5.7** *Redo Experiment 5.6 with whichever fitness function exhibited superior performance, but replace tournament selection with roulette selection. What effect does this have? Be sure to compare graphs of average and best fitness in each generation.*

## Problems

**Problem 5.1** *When we defined lethal walls, the statement was made that "in a lethal wall world the fitness function should be a nondecreasing function of the time spent in the world." Explain why in a few sentences. Give an example of a fitness function which decreases with time and has evil side effects.*

**Problem 5.2 Essay.** *Explain in colloquial English what is going on in the basic symbot motion loop depicted in Figure 5.2. Be sure to say what each of the local variables does and explain the role of* $C_s$*. What does the constant* 1.6 *say about the placement of the wheels?*

**Problem 5.3** *In a world with reflecting walls, a symbot is supposed to bounce off of the walls so that angle of incidence equals the angle of reflection. Give the formula for updating*

*the symbots heading $\theta$ when it hits a wall parallel to the $x$-axis and when it hits a wall parallel to the $y$-axis. Hint: this is really easy. You may want to give your answer as a modification of the basic symbot motion loop, given in Figure 5.2.*

**Problem 5.4** *Carefully graph the field that results from*
*(i) an inverse square law source at* $(0.5, 0.5)$*,*
*(ii) a truncated inverse square law source at* $(0.5, 0.5)$ *with radius* $0.1$*,*
*(iii) two truncated inverse square law sources with radius* $0.1$ *at* $(0.25, 0.25)$ *and* $(0.75, 0.75)$*.*

**Problem 5.5 Essay.** *Think about a light bulb. Why is there no singularity in the field strength of the light emitted by the bulb? The inverse square law is a good description of the bulb's field at distances much greater than the size of the bulb. Is it a good description close up?*

**Problem 5.6 Essay.** *Suppose we are running an evolutionary algorithm with a lexical fitness function $f$ lex $g$ ($f$ dominant). If $f$ is a real-valued, continuous, nonconstant function, how often will we use $g$? Why is it good, from the perspective of $g$ being useful, if $f$ is a discretely-valued function? An example of a discretely-valued function is the graph crossing number function explored in Section 3.5.*

**Problem 5.7 Essay.** *In Chapter 3, we used niche specialization to keep a population from piling up at any one optimum. Could we use a lexical product of fitness function to do the same thing? Why or why not? More to the point, for which sorts of optimization problems might the technique help and in which would it have little or no effect.*

**Problem 5.8** *Think about what you know about motion from studying physics. Rewrite the basic symbot motion loop so that the symbots have mass, inertia, and rotational inertia. Advanced students should treat the symbot's wheels and give them rotational inertia as well.*

**Problem 5.9** *If you are familiar with differential equations, write out explicitly the differential equations that are numerically solved by the basic symbot motion loop. Discuss them qualitatively.*

**Problem 5.10 Essay.** *In which of the experiments in Section 5.1 are we using a fixed fitness function and in which are we using one that changes? Can the varying fitness functions be viewed as samples from some very complex, fixed fitness function? Why or why not?*

**Problem 5.11 Short Essay.** *Are the fitness functions used to evolve symbots polymodal or unimodal? Justify your answer with examples and logic.*

**Problem 5.12** *Suppose we have two symbots with sensors $\pi/4$ off its symmetry axis, a radius of* $0.05$*, and connection strengths:*

| | | |
|---|---|---|
| *LL* | *1* | *0* |
| *LR* | *0* | *1* |
| *RL* | *0* | *1* |
| *RR* | *1* | *0* |
| *Idle* | *0.5* | *0.5* |

*and a single inverse truncated square law source at* $(0.5, 0.5)$*. Compute the symbot's direction of motion (forward/backward) and current turn direction (left/right) at* $(0.25, 0.25)$*,* $(0.25, 0.75)$*, and* $(0.75, 0.75)$ *assuming it is facing in the positive y direction and then, again, assuming it is facing in the positive x direction. Do we need* $C_s$ *and* $C_f$ *to do this problem?*

## 5.2 Symbot Bodies and Worlds

In this section, we will explore various symbot worlds, free up the parameters that define the symbot's body, and allow evolution to attempt to optimize details of the symbot body plan. At its most extreme this will involve modifying the basic symbot body plan to allow asymmetry and additional sensors.

In Section 5.1, we defined reflecting, lethal, and stopping walls but did not use them. Our first experiment in this section explores these other possible symbot worlds. The experiment asks you to report on what differences resulted from changing the symbot world. Before doing this experiment, you should discuss in class what you *expect* to happen when you change the world. Write down your predictions both before and after the discussion and compare them with the actual outcome of the experiment.

**Experiment 5.8** *Modify the software from Experiment 5.6 so that the walls may be optionally lethal, reflecting, stopping, or nonexistent. Using whichever fitness function from Experiment 5.6 gave the best performance, run 20 ecologies in each sort of world. Do different behaviors arise in the different worlds? How do average scores differ? Do the symbots merely deal with or do they actually exploit the reflective and stopping walls?*

In experiments with lethal walls, it is interesting to note that the symbots can actually learn where the walls are, even though they have no sensors that directly detect them. If you have the time and inclination, it is instructive to recode Experiment 5.2 to work with lethal walls. In Experiment 5.2, the placement of the source gives reliable information about the location of the walls, and hence the symbot can learn more easily where the walls are.

In Section 5.1, we had symbots with sensors that were at an angular displacement of $\pi/4$ from the symbot's axis of symmetry. This choice was an aesthetic one, it makes the symbots look nice. We also know the symbots were able to show a good level of performance with these fixed sensor locations. There is, however, no reason to think that fixing the sensors at $\pi/4$ off the axis of symmetry is an optimal choice and we will now do an experiment to see if, in fact, there are better choices.

**Experiment 5.9** *Modify the evolutionary algorithm used in Experiment 5.6 so that it operates on a gene that contains two additional loci, the displacements off the axis of symmetry of the left and right sensors in radians. Run 30 populations of size 60 for 75 generations with the displacements*

*(i) equal but with opposite sign, and*

*(ii) independent.*

*That is to say, the sensors should be coerced to be symmetric in one set of runs and allowed to float independently in the other. What values for sensor displacements occur? How does the performance of evolution in this experiment compare with that in Experiment 5.6? When the two sensor locations float independently you will need to make a small, obvious modification to the basic symbot motion loop. Include a discussion of this modification in your write up.*

In our version of Experiment 5.9, two common designs were Chameleon (sensors at $\pi/2$ off the axis of symmetry) and Cyclops (both sensors on the axis of symmetry - one in front and one in back). Note, Cyclops can only occur in the second set of runs. When writing up Experiment 5.9, be sure to note any designs that are substantially different from Cyclops or Chameleon.

So far each symbot in our experiments has had a body size of 0.05, 1/20 of the width of the unit square. Making a symbot larger would clearly benefit the symbot; even blundering movement would cover a greater area. A symbot with a radius of 1, for example, would cover all or most of the unit square and hence would "find" things quite efficiently. In addition, if we assume fixed sensor locations, then symbots that are larger have more resolution on their sensors. It is not clear if this is good or bad. The farther apart their two sensors are, the more difference in the field strength they feel. If a symbot is big enough, it is often the case that one sensor is near one source and the other is near another. Such a symbot may have different design imperatives than a symbot that is small.

In the following experiment, we will explore the radius parameter $R$ for symbots. We will use a new technique, called *population seeding*. In population seeding, an evolved population generated in the past is used as the starting population for a new evolution. Sometimes this is done just to continue the evolution, possibly multiple different times, to test for contingency or look for added progress toward some goal. However, it also gives you the opportunity to change the fitness function so as to approach some goal stepwise. If we start with a population of symbots that can already find sources efficiently, then evolution can concentrate on optimizing some other quality, in this case the symbot's radius.

A bit of thought is required to design an experiment to explore the utility of radius to a symbot. The area of a symbot is $\pi R^2$ while the cross section it presents in the direction of motion is $2R$. The symbot's area is the fraction of the unit square it covers but, since it moves, its leading surface might well be the "useful" or "active" part of the symbot. There

is also the role of sensor separation in maneuvering to consider. Symbots that are too small feel almost no difference in their sensor strengths, while symbots that are too large can have inputs from distinct sources dominating each of their sensors. This means that symbot fitness might vary linearly as size, quadratically as size, or vary according to the average distance between sources. The truth is probably some sort of subtle combination of these and other factors. The following experiment can serve to place some bounds and serve as the starting point for designing additional experiments.

**Experiment 5.10** *Modify the software from Experiment 5.6, fitness function (i), setting* $k = 5$ *to provide a source-rich environment. Modify the symbot gene so that radius, set initially to 0.05, is part of the evolving gene. Allow radii in the range* $0.01 \leq R \leq 0.25$ *only. Run 3 sets of 30 populations with population size 32 for 60 generations where the fitness function is*

*(i) unmodified,*

*(ii) divided by the symbots diameter,* $2R$*, and*

*(iii) divided by the symbots area,* $\pi R^2$*.*

*Instead of generating random initial creatures use a population of evolved symbots from Experiment 5.6. Doing this will allow the use of the simpler fitness function - an already evolved population should not need the lexical fitness function boost to its early evolution.*

*For your write up, plot the distribution of radii in the final population of each run. Write a few paragraphs that explain what this experiment has to say about the effect of radius on fitness. Did some sets of runs move immediately to the upper or lower boundary of permitted radius?*

So far, the symbots we have examined have two sensors and, with the exception of Experiment 5.9, bilateral symmetry. This is because they are modeled on biological creatures. The sensors are thought of as two eyes. Maybe three sensors would work better. Let's try it and see.

**Experiment 5.11** *Rewrite the code from Experiment 5.6 so that the symbots have 3 genetic loci that give the angular position of 3 sensors, where 0 is the direction the symbot moves as the result of idle speed alone (the forward direction along its axis of symmetry). You will need to rewrite the basic symbot motion loop to involve 6 sensor/wheel connections, as per Problem 5.16. Run 20 populations of 60 symbots for 75 generations saving average and maximum fitness and the sensor positions of the best symbot in the final generation of each population. What arrangements of sensors occur in your best symbots? How does fitness compare with the fitnesses in Experiments 5.6 and 5.9?*

There are several hundred possible experiments to be done with symbots, just by using the elements of the experiments presented so far in this section. A modest application of imagination can easily drive the total into the thousands. The author urges anyone who

thinks up *and performs* such experiments to contact him. Some additional suggestions: a symbot with a 2 segment body, segments joined by a spring; moving the wheels of the symbot around; adding noise to the symbot's sensors; implementing more realistic underlying physics for the symbots. In this book, our next step will be to give the symbots some modest, additional control mechanisms.

### Problems

**Problem 5.13** *Write out the new version of the basic symbot motion loop, given in Figure 5.2, needed by Experiment 5.9.*

**Problem 5.14** *Often, a lexical product fitness function is used when evolving symbots. Explain why, if we seed a population with evolved symbots and then continue evolution, such a lexical product is not needed.*

**Problem 5.15 Essay.** *Suppose we are running an evolutionary algorithm in which we found a lexical product of two fitness function $f$ and $g$ with $f$ dominant to be helpful. Discuss the pros and cons of using $f$ lex $g$ for only the first few generations and then shifting to $f$ alone as the fitness function. Give examples.*

**Problem 5.16** *Give pseudo-code, as in Figure 5.2, for the basic symbot motion loop of a symbot with 3 sensors at angular positions $\tau_1$,$\tau_2$, and $\tau_3$ counterclockwise from the direction of forward motion.*

**Problem 5.17** *True or False? A symbot with a single sensor could find sources and evolve to higher fitness levels using the setup of Experiment 5.9.*

## 5.3 Symbots with Neurons

The symbots we have studied so far have a feed forward neural net with 2 or 3 input neurons (the sensors), 2 output neurons (the wheels), and no hidden layers or interneurons. The complexity of the symbot's behavior has been the result of environmental interactions: with the field, with the walls, and with the sources. In this section, we will add some neurons to the symbot's control structures.

Recall from Section 11.1 that a neuron has inputs which are multiplied by weights, summed, and then run through a transfer function. The name of a type of neuron is usually the name of its transfer function (hyperbolic tangent, arctangent, or Heaviside for example). The underlying function for the neuron may be modified by vertical and horizontal shifting and stretching. These are represented by 4 parameters so that, with $f(x)$ being our transfer function, in

$$a \cdot f(b \cdot (x - c)) + d \tag{5.3}$$

the parameter $a$ controls the degree of vertical stretching; the parameter $b$ controls the degree of horizontal stretching; the parameter $c$ controls the horizontal shift; and the parameter $d$ controls the vertical shift. To see examples of these sorts of shifts look at Figure 5.7.

In Experiment 5.9, we allowed evolution to explore various fixed locations for a pair of sensors. What if the symbot could change the spacing of its sensors in response to environmental stimuli? Let's try the experiment. We should design it so that it is *possible* for evolution to leave the sensors roughly fixed, in case that solution is superior to moving the sensors. In order to do this, we will take the basic symbot and make the symmetric sensor spacing parameter $\theta$ dynamic, controlled by a single neuron. Since $-\pi/2 \leq \theta \leq \pi/2$ is a natural set of possible sensor positions, we will choose an arctangent neuron. The neuron should use the the sensors as inputs, requiring 2 connection weights, and will have 2 parameters that are allowed to vary, $b$, and $c$ from Equation 5.3 ($a$ and $d$ are set to 1).

**Experiment 5.12** *Modify the software from Experiment 5.6, fitness function (ii), and the basic symbot motion loop to allow the symmetric spacing of the sensors to be dynamically controlled by an arctangent neuron of the form*

$$\arctan(b \cdot (x - c)).$$

*The parameters b and c as well as the connection strength ln and rn of the left and right sensors to the neuron must be added as new loci in the symbot gene. Before iterating the basic symbot motion loop during fitness evaluation initialize $\theta$ to pi/4. Here is the modification of the basic symbot motion loop.*

*Begin*

$x_1 := x + R \cdot \cos(\tau + \theta)$; //*left sensor position*
$y_1 := y + R \cdot \sin(\tau + \theta)$;
$x_2 := x + R \cdot \cos(\tau - \theta)$; //*right sensor position*
$y_2 := y + R \cdot \sin(\tau - \theta)$;
$d_l := f(x_1, y_1) \cdot ll + f(x_2, y_2) \cdot rl$; //*find wheel*
$d_r := f(x_2, y_2) \cdot rr + f(x_1, y_1) \cdot lr$; //*drive strengths*
$\theta = arctan(b \cdot (ln \cdot f(x_1, y_1) + rn \cdot f(x_2, y_2) - c))$ //*new sensor spacing*
$ds := C_s \cdot (dl + dr + s \cdot R/2)$; //*change in position*
*If* $|ds| > R/2$ *then* //*truncate*
    *If* $ds > 0$ *then* $ds := R/2$ *else* $ds := -R/2$;
$d\tau := 1.6 \cdot C_s/R \cdot (dr - dl)$; //*change in heading*
*If* $d\tau > \pi/3$ *then* $d\tau := \pi/3$; //*truncate*
$x := x + ds \cdot \cos(\tau)$; //*update position*
$y := y + ds \cdot \sin(\tau)$;
$\tau := \tau + d\tau$; //*update heading*

*end;*

*The parameters b and c should be initialized to* 1 *when generating the initial population, the connection strengths ln and rn should start in the range* $-1 \leq x \leq 1$. *Do 2 sets of 20 runs on populations of 40 symbots for 75 generations. In the first set of runs, generate all the symbot genetic loci randomly. In the second set of runs get the parameters rr, rl, lr, ll, and idle from an evolved population generated by Experiment 5.6. In addition to the usual fitness data, save the mean and standard deviation of the 4 neuron parameters and devise a test to see if the symbots are using their neurons. (A neuron is said to be* used *if* $\tau$ *varies a bit during the course of a fitness evaluation.)*

Recall that

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{5.4}$$

Now, we will move to a 2 neuron net, one per wheel, in which we just put the neurons between the sensors and the wheels. We will use hyperbolic tangent neurons. Recall the reasons for truncating the inverse square law sources? (Section 5.1) We did not want absurdly large signal inputs when the symbots had a sensor too near an inverse square law source. These neurons represent another solution to this problem. A neuron is *saturated* when no increase in its input will produce a significant change in its output. High signal strengths will tend to saturate the neurons in the modified symbots in the following experiment.

**Experiment 5.13** *Take either Experiment 5.6 or Experiment 5.12 and modify the algorithm so that instead of*

$$d_l := f(x_1, y_1) \cdot ll + f(x_2, y_2) \cdot rl;$$
$$d_r := f(x_2, y_2) \cdot rr + f(x_1, y_1) \cdot lr;$$

*we have*

$$d_l := R/2 \cdot tanh(bl \cdot (f(x_1, y_1) \cdot ll + f(x_2, y_2) \cdot rl) + cl);$$
$$d_r := R/2 \cdot tanh(br \cdot (f(x_2, y_2) \cdot rr + f(x_1, y_1) \cdot lr) + cr);$$

*where* $bl, cl, br, cr$ *are new real parameters added to the symbot's gene. Initialize bl and br to 1 and cl and cr to 0. This will have the effect of having the neurons fairly closely mimic the behavior of the original network for small signal strengths. Seed the values of* $ll, lr, rl, rr$, *and s with those of an evolved population from Experiment 5.6.*

*Run at least 10 populations of 60 symbots for 75 generations. Document changes in the efficiency of evolution and comment on any new behaviors (things that did not happen in the other evolutions).*

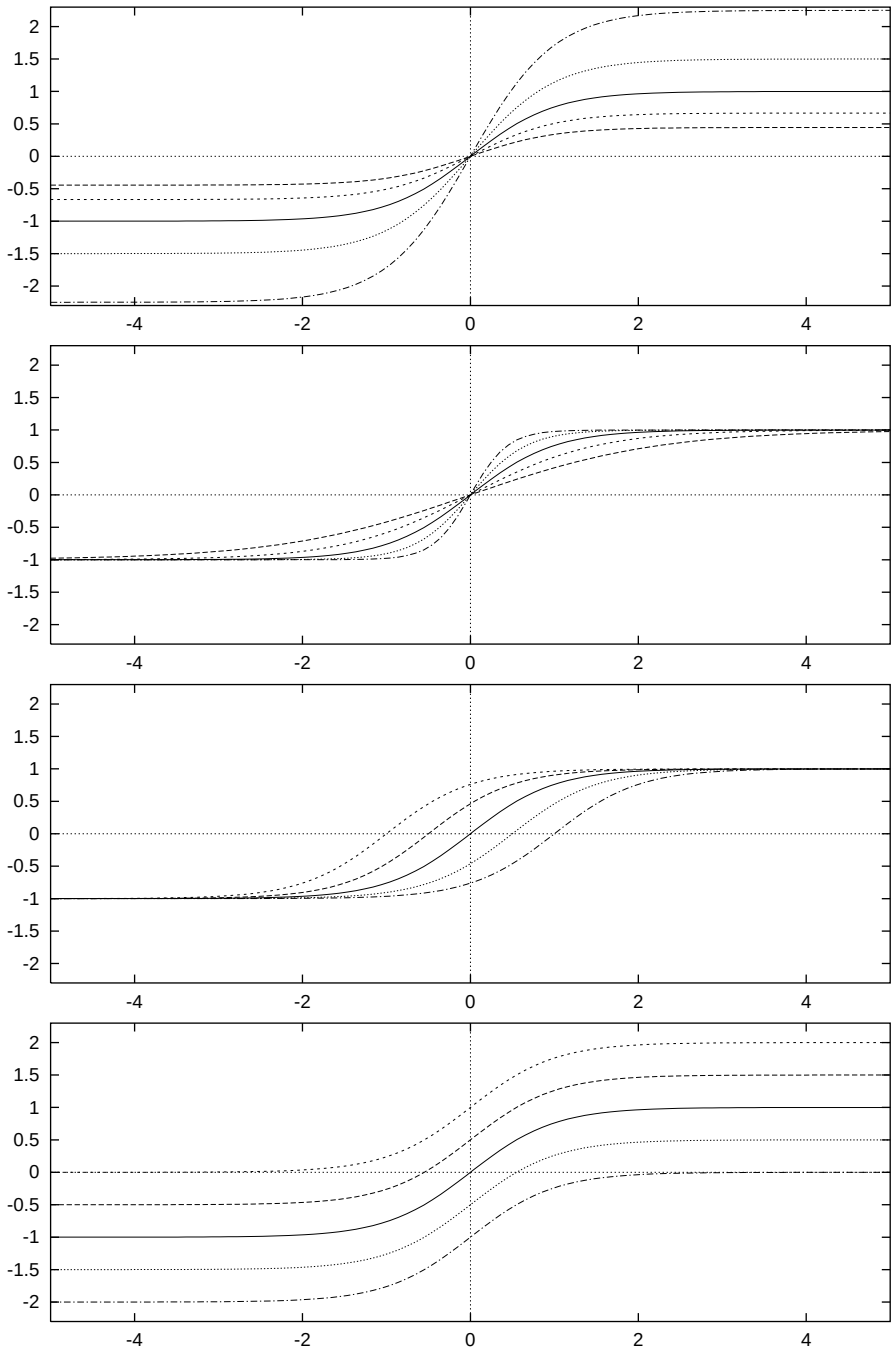


Figure 5.7: Variation of a,b,c, and d for the hyperbolic tangent

The hyperbolic tangent neuron is computationally expensive, so we should see if a cheaper neuron can help. The transfer function

$$f(x) = \begin{cases} -1 & x \leq -1 \\ x & -1 < x < 1 \\ 1 & 1 \leq x \end{cases} \tag{5.5}$$

is much cheaper to compute. Let us do an experiment to test its performance.

**Experiment 5.14** *Repeat Experiment 5.13 replacing the tanh(x) function with Equation 5.5. Compare the performance of the final symbots and the speed with which the populations converge to their final form.*

## Problems

**Problem 5.18** *Can the symbots in Experiment 5.12 set the parameters of their sensor positioning neuron so as to mimic symbots with fixed sensor positions? Give neuron parameters that yield fixed sensors or show why this cannot be done. In the event that only* some *fixed positions are possible, show which these are.*

**Problem 5.19** *Assume that you are working in a computer language that does not have hyperbolic tangents as primitive functions. Unless you are using quite advanced hardware, computing exponentials is more expensive than multiplication and division which is in turn more expensive than addition and subtraction. Assume you have the function $e^x$ available (it is often called exp(x)). Find a way to compute tanh(x) (Equation 5.4) using only one evaluation of $e^x$ and two divisions. You may use as many additions and subtractions as you wish.*

**Problem 5.20** *Describe a way to* efficiently *substitute a lookup table with 20 entries for the function tanh(x) in Experiment 5.13. Give pseudo-code. A lookup table is an array of real numbers together with a procedure for deciding which one to use for a given x. In order to be efficient, it must not use too many multiplications or divisions and only a moderate amount of addition and subtraction. Graph tanh(x) and the function your lookup table procedure produces on the same set of axes. Advanced students should also augment the lookup table with linear interpolation.*

**Problem 5.21 Essay.** *Examine the graph of $\tanh(x^3)$ as compared to tanh(x). Discuss the qualitative advantages and disadvantages of the two functions as neuron transfer functions. What about the shape of the first function is different and when might that difference be significant?*

**Problem 5.22 Essay.** *If we use hyperbolic tangent neurons as in Experiment 5.13, then large signal strengths are ignored by saturated neurons. Using experimental data, compare the minimal symbots that rely on truncating (Experiment 5.6) with the ones that have saturation available. Are the neuron-using symbots superior in terms of performance, "realism," or stability?*

**Problem 5.23 Essay.** *Explain the choices of a and d made in Experiment 5.12. Why might vertical shift and stretch be bad? How would you expect the symbots to behave if these parameters were allowed to vary?*

## 5.4 Pack Symbots

In this section, we will examine the potential for coevolving symbots to work together. We will also try to pose somewhat more realistic tasks for the symbots.

To this end, we define the *Clear-the-Board* fitness function. Start with a large number of sources and place no new sources during the course of the fitness evaluation. Fitness is the lexical product of the number of sources found with $\frac{1}{d+1}$ where $d$ is the closest approach the symbot made to a source it did not find (compare with Experiment 5.6).

We will distribute the large number of sources using one of three algorithms: uniform, bivariate normal, and univariate normal off of a line running through the fitness space. Think of the sources as spilled objects. The uniform distribution simulates a small segment of a wide area spill. The bivariate normal distribution is the scatter of particles from a single accident at a particular point. The univariate normal off of a line represents something like peanuts spilling off of a moving truck.

**Experiment 5.15** *Modify the software in Experiment 5.6, fitness function (ii), to work with a Clear-the-Board fitness function. If two symbots both clear the board, then the amount of time taken is used to break the tie (less is better). Change the symbots radius to 0.01 and have* $k = 30$ *sources. Run 20 populations of 60 symbots for 50 generations on each of the 3 possible distributions:*

*(i) Uniform,*

*(ii) Bivariate normal with mean (0.5,0.5) and variance 0.2, and*

*(iii) Univariate normal with variance 0.1 off of a line.*

*See Problem 5.25 for details of problem (iii). Seed the populations with evolved symbots from Experiment 5.6. When the normal distribution produces points not inside the unit square, simply ignore those points and generate new ones until you get enough points. Report mean and best fitness and say which distributions allowed the symbots to learn to clear the board most often. If it appears that the symbots could clear the board given a little more time you might try increasing the number of iterations of the symbot motion loop allowed. You should certainly terminate fitness evaluation early if the board is cleared.*

Experiment 5.15 is intended to give you practice with the new fitness function and the new patterns of source distribution. With these in hand, we will move on to *pack symbots*. Pack symbots are symbots which learn to work together in groups. There are two ways to approach pack symbots: specify a set of symbots with a single gene, or evolve several populations of symbots whose fitness is evaluated in concert. For both approaches, there are many symbots present in the unit square simultaneously. It may be that the various symbots will learn to coevolve to do different tasks. One would hope, for example, that, in the experiments with a bivariate normal source distribution, several symbots would intensively scour the center of the region while others swept the outer fringes.

A new problem that appears in multiple symbot environments is that of symbot collision. Symbots realized in hardware might well not care too much if they bumped into one another occasionally, but it is not desirable that we evolve control strategies in which symbots pass through one another. On the other hand, realistic collisions are difficult to simulate. Aside from mentioning it, we will, for the present, ignore this problem of symbot collisions.

**Experiment 5.16** *Modify the software from Experiment 5.15 so that a gene contains the description of $k$ symbots. The resulting object is called a polysymbot. All $m$ symbots are run at the same time with independent positions and headings. The fitness of a polysymbot gene is the sum of the individual fitnesses of the symbots specified by the gene. Run 20 populations of 60 polysymbots for 100 generations on one of the 3 possible distributions of sources for $m = 2$ and $m = 5$. Use $k = 30$ sources on the board.*

*In addition to documenting the degree to which the symbots clean up the sources and avoid colliding with each other, try to document, by observing the motion tracks of the best cohort in the final generation of each run, the degree to which the symbots have specialized. Do a few members of the group carry the load or do all members contribute?*

In the next experiment, we will try to coevolve distinct populations instead of gene fragments.

**Experiment 5.17** *Modify the software from Experiment 5.16, with $m = 5$ symbots per pack, so that instead of a gene containing 5 symbots, the algorithm contains 5 populations of genes that describe a single symbot. For each fitness evaluation the populations should be shuffled and cohorts of five symbots, one from each population, tested together. Each symbot is assigned to a new group of five in each generation. The fitness of a symbot is the fitness that its cohort, as a whole, gets. Do the same data acquisition runs as in Experiment 5.16 and compare the two techniques. Which was better at producing coevolved symbots that specialize their tasks?*

## Problems

**Problem 5.24** *Is the fitness function specified in Experiment 5.15 a lexical product or not? Check the definition of lexical products very carefully and justify your answer.*

**Problem 5.25** *In the experiments in this section, we use a new fitness function in which the symbots attempt to clear the board of sources. To generate uniformly distributed sources, you generate the $x$ and $y$ coordinates as uniform random numbers in the range $0 \leq x, y \leq 1$. The bivariate normal distribution requires that you generate two Gaussian coordinates from the random numbers (the transformation from uniform to Gaussian variables is given in Equation 3.1). In this problem, you will work out the details of the Gaussian distribution of sources about a line.*

*(i) Give a method for generating a line uniformly selected from those that have at least a segment of length 1 inside the unit square.*

*(ii) Given a line of the type generated in (i), give a method for distributing sources uniformly along its length but with a Gaussian scatter about the line (with the line as the mean). Hint: use a vector orthogonal to the line.*

**Problem 5.26** *Imagine an accident which would scatter toxic particles so that the particles would have a density distribution that was a Gaussian scatter away from a circle. Give a method for generating a field of sources with this sort of density distribution.*

**Problem 5.27** *Give a method for automatically detecting specialization of symbots for different tasks as one would hope would happen in Experiments 5.16 and 5.17. Logically justify your method. Advanced students should experimentally test the method by incorporating it into software.*

**Problem 5.28 Essay.** *Describe a baseline experiment that could be used to tell if a polysymbot from either Experiment 5.16 or 5.17 was more effective at finding sources than a group of symbots snagged from Experiment 5.6.*