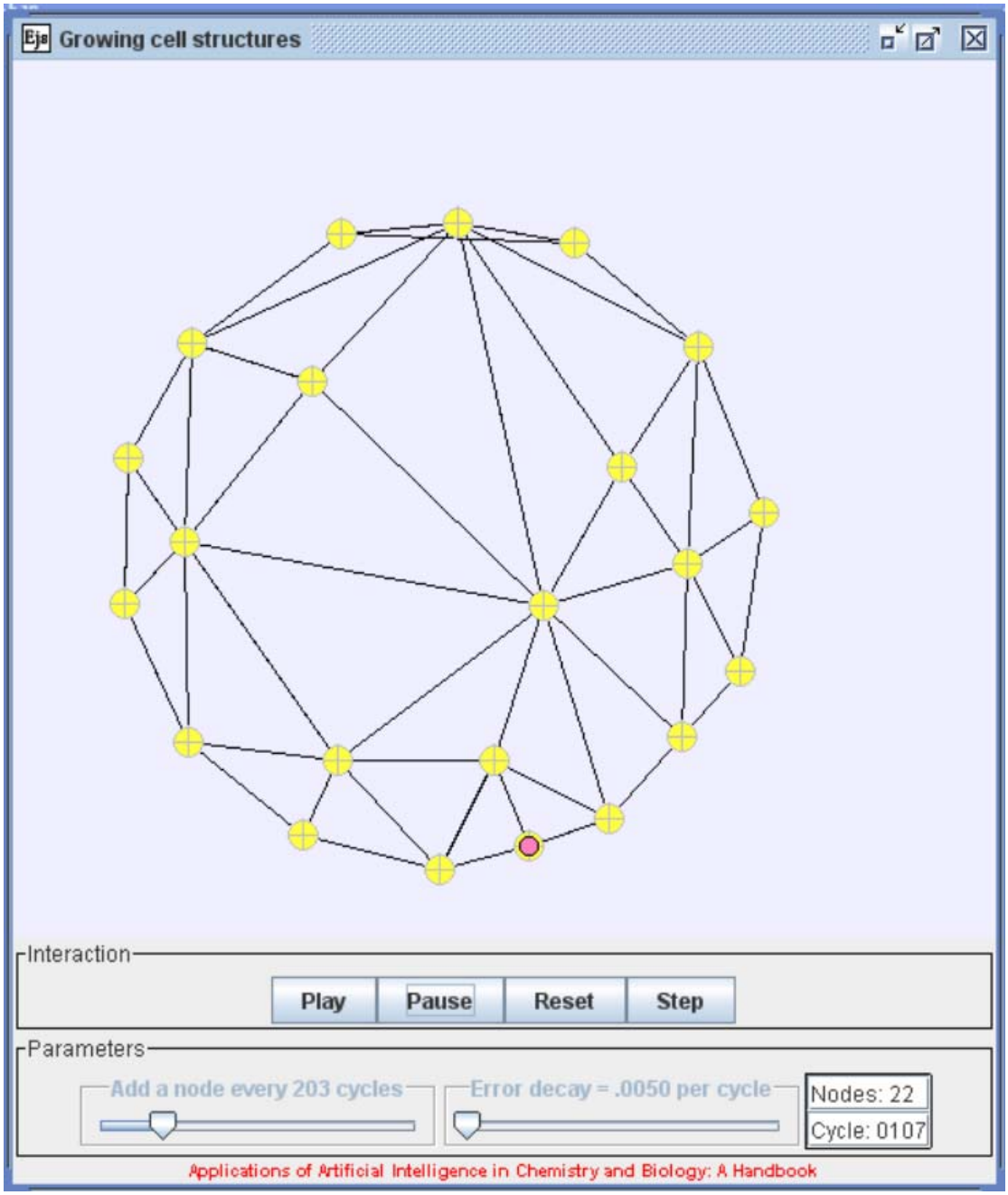
Growing Cell Structures – fitting nodes to a donut-shaped data set

This provides a brief introduction to the Growing Cell Structures program.



The initial network contains just three nodes. A random data point is generated by first choosing an angle at random

```
theta=360.0*Math.random();
```

and then using this angle to calculate x- and y- coordinates, with the addition of some noise:

```
rx=5.0+0.5*Math.random()+4.0*Math.sin(theta);
ry=5.0+0.5*Math.random()+4.0*Math.cos(theta);
```

The data point then provides the input into the GCS. The program compares the values of rx and ry with the weights at each node in turn…

```
for (int i=0; i<nnodes; i++)
{
    diff=(weights[i][0]-rx)*(weights[i][0]-
rx)+(weights[i][1]-ry)*(weights[i][1]-ry);
    if (diff<mindiff)  // found a better match than any
previous ones
    {
        mindiff=diff;
        bestnode=i;    // record the position of the best
node so far
    }
}
```

… in order to find the node whose weights most closely match the values rx and ry.

The goal of the GCS is to arrange its nodes so that they well represent the data. Since each node is plotted at a position determined by its weights, interpreted as x- and y-coordinates, as the map evolves the shape represented by the data points should appear.

The node that was most recently added is highlighted in the display.

BUTTONS

Buttons in the window created by the program have the following functions:

Button	Function
Play	Restart the program if it has previously been paused.
Pause	Temporarily halt execution.
Reset	Restart the calculation from scratch; all parameters are set to their default values.
Step	Execute a single cycle of the program.

SLIDERS

The parameters that can be adjusted by the user are:

Slider	Default	Comment
Add a node	400 cycles	Once the defined number of cycles has passed, a new node is added.
Error decay	0.005	Every cycle the signal counter at each node is multiplied by (1.0-error decay) so that, in the absence of new wins, the counter gradually diminishes.

Investigations & Exercises

1. Default parameters

Run the program using the default parameters and note the gradual emergence of a pattern resembling the input data. Nodes will cluster towards the edge of the donut, but a few nodes will appear in the middle. This version of the program does not include code whose role is to remove redundant nodes, so nodes that lie in the middle, but do not reflect the data accurately, will not be removed.

2. Add a node rate

The behaviour and reliability of a GCS is noticeably dependent upon the rate at which nodes are added. Restart the simulation and try adjusting the **Add a node rate** to a small value. Note that the GCS still attempts to fit

the input data, but that nodes may be added unevenly and the program does a poorer job of representing the data than when nodes are added less frequently.

3. Error decay

Restart the simulation and investigate the effect of increasing the error decay. This parameter determines how quickly the value of the signal counter at each node decays; a value of one leads to no data being retained about the past success of any node. Note the difference between running the simulation with a very high value of the decay and a more “normal” value of 0.01 or less.

4. Node removal

If you are familiar with programming in Java, modify the program, or write one of your own, so that the removal of redundant nodes (those whose signal counters become very small) can be included.