Appendix E

This appendix contains a description of the software used in the project in implementing the evolutionary ANN.

The software allows modules of neurons and input sensors to be added or deleted from the network. Neurons in the network can be selected to be an output neuron. There are options to initialize or modify neuron and sensor parameters, connection status and associated weights. These parameters are subject to training when a new module is added. An Evolutionary strategy is used to evolve these parameters. Finally, the trained module can be retained and saved into a text file. Saved networks can also be reused as the network expands. All the results presented throughout Chapter 6 to 8 are obtained using this simulation software. The results presented are “averages” over several experiments and not “one-offs” test data.

The evolutionary technique was programmed using Borland C++ Builder Version 5. There are 70 functions associated with the software which manage the operation of the simulation. The software is divided into two different main windows. The layout of the first window (which handles the Evolution of Modular Artificial Neural Networks) is shown in Figure E.1.
The software initiates four different types of linked list. These are:

1) System Neuron List (SysNeuLst) – Linked list which stores the neurons in the network
2) System Sensor List (SysSnrLst) – Linked list which stores the sensor inputs to the network
3) Neuron Connection List (NeuCnntLst) – Linked list which stores the neuron connections to be trained
4) Neuron Property List (NeuPropLst) – Linked list which stores the neuron properties to be trained

Described below is the operation of the buttons on the layout (Figure E-1) above.

**New Function** – Assign an ID (N) for different functions added to the system

**Add New Module** – Assign an ID (N) for each new module added to the network

N is an integer from 1 to +∞, ID is the Identity and M is an integer from -1 to -∞
**Add Neuron** – A single neuron can be added at a time. The added neurons are assigned an ID (N) for example 1, 2, 3 etc. Firstly, a neuron structure as shown below is created. The data structure for a new neuron requires function ID, module ID, neuron ID, neuron parameters, training status and two linked lists. Then the system neuron list (**SysSnrLst**) is scanned and the last inserted neuron IDs are obtained. The function and module ID is obtained from the form (Figure E-1) above. The neuron parameters values are initialized to zero. The training status determines whether the neuron parameters, input connections and weights associated with the connection will undergo training. The two linked lists are neuron and sensor input list. These lists contain input information from other neurons and sensors in the network. The neuron and sensor input structure is shown below. Since recurrent connections are allowed in the network, a neuron can be connected to itself. Therefore, as soon as a new neuron is inserted into the network, a neuron input data structure is created and added to the input list. This new neuron will receive and make connections to and from other neurons in the network. The number of input data structures varies and depends on number of neuron in the network. The connections weight and status for the input neurons is initialised to zero. System sensor list (**SysSnrLst**) is also scanned and Input sensor data structures are created. The input value comes from the user while the other two parameters are set to zero. Finally, the neuron structure is added to the system neuron link list (**SysNeuLst**).

```c
struct Neu                      //neuron structure
{
    int fId;                        //function ID
    int mId;                      //module ID
    int id;                         //neuron ID
    double dc;                  //decay constant
    double isp;                 //internal state parameter
    double th;                   //threshold
    int st;                          //training status
    TList *NeuInpLst;     //neuron input list
    TList *SnrInpLst;      //sensor input list
};
```
struct InpNeu     //neuron input structure
{
    int fId;       //function ID
    int mId;       //input neuron module ID
    int id;        //neuron ID
    double lb;     //connection weight
    int st;        //connection status (connected or not connected)
};

struct InpSnr     //sensor input structure
{
    int fId;       //function ID
    int mId;       //input neuron module ID
    int id;        //sensor ID
    double inp;    //sensor input
    double lb;     //connection weight
    int st;        //connection status (connected or not connected)
};

Del Neuron  – Removes the selected neuron from the module. If neurons are not deleted from the list in sequence, a background function will then sort the neuron’s ID in ascending order. This change is updated throughout the network.

Add Sensor  – Add sensor function is very similar to Add Neuron. Firstly, when a new sensor is added to the network, a system sensor structure is created and added to system sensor list (SysSnrLst). Secondly, the sensor input list (SnrInpLst) of each neuron is updated. The data structure for the system sensor is shown below. The reason for having a separate list is to monitor and maintain the growth of the sensors in the network. Sensors can only be added when there is at least one neuron in the network. The button adds a single sensor. Each sensor is assigned an ID (M) i.e -1, -2, etc.
struct Snr  //system sensor structure
{
    int fId;   //function ID
    int mId; //module ID
    int id;    //sensor ID
};

Del Sensor – Removes the selected sensor from the SysSnrLst and SnrInpLst of every neuron. If sensors are not deleted from the list in sequence, a background function will then sort the sensors ID in descending order. This change is updated throughout the network.

Clear Network – Removes all the neurons and sensors for the selected function and module ID.

Neuron Property – List the selected neuron parameters from the System Neuron Listbox. There are options to enable and disable training neuron parameters. Figure E-2 below shows the layout for a selected neuron.

Figure E-2
**Sel Neu for Training** – This option enables the selected neuron’s (of the System Neuron Listbox) inputs (neuron and sensor connections) to undergo training.

**Neu Cnnt for Training** – Display the neuron’s ID whose input connections are selected for training.

**Neu Prop for Training** – Display the neuron’s ID whose neuron properties are selected for training.

**Del Neu Cnnt** and **Del Neu Prop** – Remove the selected neuron.

**Sel Out Neu** – Selects the output neurons from the System Neuron listbox.

**Output Neuron** – Display the selected output neuron.

**Del Out Neu** – Deletes the selected output neuron from Output Neuron combobox.

**Save Network** – The network information (neuron parameters, input connections and associated weights, Evolutionary Strategy parameters) is written to a text file.

**Load Network** – Loads the saved network for evaluation.

**Simulate Network** – This will test run the loaded network for 500 time steps.

**Crt Dcd Lst** – Firstly, the Neuron Properties (**NeuProp**) and Connections (**NeuCnnt**) data structure is created as shown. Secondly, **SysNeuLst** is scanned to determine the training and evaluate whether the status of each neuron is enabled or disabled. Neuron input connections or properties of the enabled neuron will undergo training. The neuron connection data structure is added to neuron connection list (**NeuCnntLst**) and the neuron properties data structure is added to neuron properties list (**NeuPorpLst**). Figure 5-7 of Chapter 5 illustrates how the information is decoded into a chromosome.
struct NeuProp  //neuron property structure
{
  int id;            //neuron ID
  double dc;    //decay constant
  double isp;   //internal state parameter
  double th;    //threshold
};

struct NeuCnnt  //neuron connection structure
{
  int NeuId;   //neuron ID
  int InpId;    //Inp neuron/sensor ID
  double st;   //neuron connections, status 0 = connected, 1 = not connected
  double lb;  //weight
};

Clr Dcd Lst – Clear decode list erases all the information stored on neuron connection and properties linked list.

DistCnntSta – Disable connection status disables the training status of a neuron. This means the selected neuron properties, input neuron and sensor connections will not undergo training.

Neuron Inputs – Enables the user to view the selected neuron inputs. There are two options. First option views all the neuron input connections. The second option shows all the sensor input connections to the neuron. The neuron Inputs form, as shown below, will appear if neuron is selected. Using this form, it is possible to edit neuron input connection weights and the connection status as shown in Figure E-3.
Figure E-3

Sensor Input form shown in Figure E-4 below will appear if sensor option is selected. Using this form it is possible to edit sensor input, connections weights and connection status as shown.
The vision grid shows the evolution of the sensory system through three different stages. In the first stage, the pixel on the centre of the grid is selected. More details about the vision sensor evolution are given in Figure 8-2 of Chapter 8. There are 25 pixels on the grid. The centre grid is selected for single patterns and it has a predefined input value of 1 or –1. Numbers of patterns and inputs (sensors) per pattern have to be specified if the multiple pattern option is selected. The input sensors become unavailable after the patterns are trained. Clicking on the grid changes the input value and pressing the OK button inserts the input pattern.

The layout of the second window is shown in Figure E-5.

**Obj connections** – The Evolutionary Strategy form will appear as shown above, when the create decode list (Crt Decd Lst) button is clicked. The Object connections list shows all the connections that will be trained. For example 1->1 means connection from neuron 1 to neuron 1.
**Crt Neu Cnt Pop** – Creates a population of neuron connection chromosomes. The Number of Chromosomes determines the size of the population. The information for the population is extracted from Neuron connection data structure. Figure 5-7 of Chapter 5 illustrates how the information is decoded into a chromosome.

**Crt Neu Prop Pop** – Creates a population of neuron properties chromosomes. The Number of Chromosomes determines the size of the population. The information for the population is extracted from Neuron properties data structure. Figure 5-7 of Chapter 5 illustrates how the information is decoded into a chromosome.

**Del Neu Cnt Pop** and **Del Neu Prop Pop** – Deletes the created neuron connections and neuron properties population.

**Evaluate Network** – Trains the network and updates neuron properties, neuron connections and its associated weights for the specified number of generations.

The set-up of the Evolutionary Strategy genetic operators is explained in detail in Section 5.3 of Chapter 5.