Technical Report OUCS-2006-12

Rehearsal and stability profiles in Hopfield type networks

Authors:

Anthony Robins, Chris Gillum

Department of Computer Science, University of Otago, New Zealand

Rehearsal and stability profiles in Hopfield type networks

Anthony Robins, Chris Gillum

February 16, 2006

This document describes a preliminary investigation of rehearsal and stability profiles in Hopfield type networks. Section 1 contains basic descriptions of the network itself followed by Section 2 which shows the methods used to both create a new network and to remove it. Section 3 describes the training and cycling of the networks, Section 4 describes the utility functions associated with the rehearsal functions, while Section 5 details the four rehearsal methods employed and the results produced.

1 Network Basics

The network actually consists of three weight independent networks. A Hebbian trained symmetric network, a Delta trained symmetric network and a Delta trained asymmetric network. Although all three networks utilise the same nodes for training and cycling, they are all trained separately and are capable of being independently cycled.

For a network to be constructed, a parameter file must be supplied containing the following information in a space separated format with each parameter being contained on a single line, eg [parameter] [value]. Some parameters are particular to individual methods while others must be supplied at all times. These are shown by references to the rehearsal methods.

- $Nodes$ [int]: The total number of nodes in the network. 5.1, 5.2, 5.3, 5.4.
- $Loops$ [int]: The number of times to repeat a given rehearsal mechanism for the purposes of averaging. 5.1, 5.2, 5.4.
- $Initial$ [int]: The number of patterns to initially train the network on. 5.1, 5.2, 5.3, 5.4
- $Extras$ [int]: The number of patterns to be introduced as intervening items. 5.1, 5.2, 5.3.
- $Pseudo$ [int]: The number of pseudo patterns required. 5.1, 5.2, 5.3, 5.4.
- $IncludeTraining$ [int]: True(1) / False(0) values indicating whether or not to include training patterns in the pseudo patterns sets. 5.1, 5.2, 5.3, 5.4.
- $HebbianLC$ [double]: The Hebbian symmetric network learning constant. 5.1, 5.2, 5.3, 5.4.
- $SymmetricLC$ [double]: The Delta symmetric learning constant. 5.1, 5.2, 5.3, 5.4.
- $AsymmetricLC$ [double]: The Delta asymmetric learning constant. 5.1, 5.2, 5.3, 5.4.
- $PatternRatio$ [double]: The ratio of on / off activations in a randomly generated pattern. 5.1, 5.2, 5.3.

- $NodeOn$ [int]: Value indicating the output state of a node when its weighted input is greater than 0. 5.1, 5.2, 5.3, 5.4.
- $NodeOff$ [int]: Value indicating the value of a node when its weighted input is less than or equal to 0. 5.1, 5.2, 5.3, 5.4.
- $RatioCut$ [double]: Used for determining if a found spurious pattern should be included in the pseudo pattern set. Can be positive (only include patterns with energy profile ratios greater than this value) or negative (only include patterns with energy profile ratios less than this value). 5.1, 5.2, 5.3, 5.4.
- $TrainingNoise$ [double]: The amount of noise to use when training the delta networks (both symmetric and asymmetric). 5.1, 5.2, 5.3, 5.4.
- $InputFile$ [string]: The name of a supplied pattern file. The number of patterns in this file should be consistent with both the number of nodes in the network, and the sum of the initial and extra patterns required. 5.1, 5.2.
- $BufferSize$ [int]: The number of patterns to be passed along with the intervening item as used in rehearsal function 1. 5.1.
- $Mode$ [int]: The mode which should be used when using rehearsal function 1. The options are 0 (full), 1 (random) and 2 (sweep). 5.1.
- $Depth$ [int]: The 'depth' to which the network should be probed. 5.3.
- $ProtoRatio$ [double]: The On / Off node activation ratio for prototype patterns. E.g. a ratio of 0.2 indicates that 20% of the nodes should be in one state. 5.4.
- Min [double]: The starting value to be used for prototyping. 5.4.
- Max [double]: The last value to be used for prototyping. 5.4.
- $Steps$ [int]: The number of steps including and between Min and Max. This has a minimum value of 2. 5.4.
- $Prototypes$ [int]: The number of prototypes to test. This value is correlated with the ProtoRatio value. This value multiplied by the ProtoRatio cannot exceed 100% as there are not enough nodes in the network to give all prototypes the same on / off ratio. 5.4.

2 Network Startup / Shutdown

Each rehearsal methods creates its own network, which is then trained and has operations performed on it relating to the rehearsal method. A Network consists of a number of structures, which are described as

Constants Holds values associated with the parameter file and other values which do not change over the life of the network.

Patterns Holds the training, intervening and pseudo patterns required by the network.

Weights Holds the weights associated with each network type.

Results Holds a number of variables associated with the various rehearsal mechanisms.

The startup function simply allocates all the memory required whilst reading in the parameter file and setting all static values. The shutdown function frees all the associated memory and shuts the program down.

3 Network Training and Cycling

3.1 Training Method

Training is performed with two separate functions. A Hebbian learning function implements this single shot training on a symmetrically connected network. A Delta learning function is capable of training on both symmetrically and asymmetrically connected networks. The Delta learning function incorporates the thermal perceptron rule where the temperature is initially set to 128 and the learning constant is also gradually decremented. The learning constants are set in the parameter file.

The Delta training function follows the following steps:

1. While the error is greater than 1 and less than 500 epochs have taken place:
    (a) For each pattern in the supplied buffer (in random order):
        i. Load the pattern into the nodes and record their current state in a 'before' buffer.
        ii. Randomly flip the activation of nodes with the probability as given in the parameter file (noise).
        iii. For each node in the pattern (in random order):
            A. Calculate the summed weighted input to this node.
            B. Set the nodes activation and record this in an 'after' buffer.
        iv. Compare the 'before' and 'after' buffer to check for changes.
        v. Calculate the 'thermal' value for this pattern.
        vi. If the training function is being used for one of the rehearsal mechanisms, the current pattern is in position 0 (this is where the intervening pattern is placed in the buffer) and the current pattern has an error of 0, then return.
        vii. Else, adjust the weights of the network using the thermal perceptron learning rule.
        viii. Make a small decrement to the temperature value and the learning constant so that after the last loop (max 500) the temperature will be 1 and the learning constant will be 0.
2. Return the network.

3.2 Cycling Method

Cycling is performed using two functions. The cycle function repeatedly calls the one cycle function until a pattern has stabilised or 500 individual cycles have taken place. These functions require that the network to be cycled is passed to them so they can use the correct weight set.

The one cycle function uses the following steps:

1. For each node in the network (in random order):
    (a) Sum the weighted inputs to this node.

   (b) If Asynchronous mode is selected, update the activation of this node, otherwise store the new activation.

2. If Synchronous mode is selected, update the activations of all the nodes in the network from the stored list.

4 Network Utility Functions

4.1 Generating Stable Patterns

This is the function used to generate all the pseudo patterns required by the rehearsal mechanisms. It is designed to find stable attractors in the weight space and can be set to deliberately exclude training patterns, or to exclude patterns above or below a given energy ratio. While it can be used for all three network types, it is only used for the Delta trained networks at this time.

It uses the following steps:

1. While the number of stable states found is less than the number required (parameter file $Pseudo$):
   (a) Generate a random pattern and cycle the network until stable.
   (b) If the current pattern's energy is outside the given limit, throw it away and try again.
   (c) If the current pattern already exists in the pseudo set, throw it away and try again.
   (d) If the current pattern matches a training pattern and training patterns have been excluded, throw it away and try again.
   (e) If all three conditions above fail, then load the current pattern as a stable attractor.

4.2 Hamming Distance

The Hamming distance is the difference between two patterns. A distance of 1 indicates that one node's activation differs between the two patterns. This method is used to calculate the average distance that a number of training patterns will settle away from their original trained state. It is mainly used in rehearsal mechanisms where the training of intervening items is damaging the weight space surrounding the originally trained patterns.

The function uses the following steps:

1. For each Delta trained network:
   (a) For each trained pattern:
      i. Load each training pattern.
      ii. Cycle the network to a stable state.
      iii. Compare the original and settled patterns and record their differences.
   (b) Average the distances.

5 Rehearsal Methods

5.1 Comparing Rehearsal Methods

This method compares six different rehearsal mechanisms. The function commences with the construction of a new network as described in Section 3. If a pattern set is not supplied, one is created which has a number of patterns equal to the sum of the number of initial and intervening patterns supplied from the parameter file. These patterns (supplied or created) are randomly assigned to be either training or intervening patterns. A number of pseudo patterns (defined in parameter file) are also generated.

For each iteration (loop) required, the following steps are processed:

1. All network weights are reset from the previous iteration.

2. All the networks are trained on the training patterns to criterion or for 500 epochs.

3. The current weights of all the networks are stored.

4. For each of the six rehearsal mechanisms, repeat:

   (a) Load the original network weights and record the number of trained patterns that are stable in each network.

   (b) For each intervening item, repeat:

      i. Train the networks on the intervening item, passing a buffer containing a number of patterns based on the rehearsal mechanism being used. Only the intervening item has an effect on stopping training. All patterns in the buffer also have an effect on the network weights but have no effect on stopping training. Once the intervening item is learnt, training stops.

      ii. Record the number of trained patterns that are stable in each network.

The buffer which is passed along with each rehearsal mechanism is described as:

1. No Rehearsal. No other patterns are based with the intervening item.

2. Full Rehearsal. All the training patterns are passed with the intervening item.

3. Random Rehearsal. Five training patterns are selected at random and are passed with the intervening item.

4. Sweep Rehearsal. For each training epoch, five training patterns are selected at random and are passed with the intervening item.

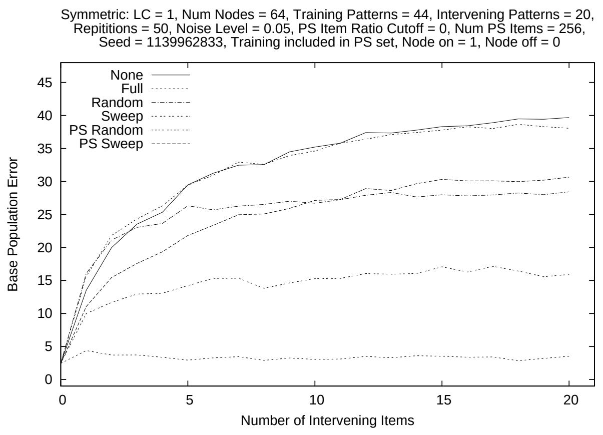
5. Random Rehearsal. Five pseudo patterns are selected at random and are passed with the intervening item.

6. Sweep Rehearsal, For each training epoch, five pseudo patterns are selected at random and are passed with the intervening item.

This function is affected by the following parameters. [Nodes], [Loops], [Initial], [Extras], [Pseudo], [Include-Training], [HebbianLC], [SymmetricLC], [AsymmetricLC], [PatternRatio], [NodeOn], [NodeOff], [RatioCut], [TrainingNoise], [InputFile].

Figure 1 shows the number of originally trained patterns that have beenforgotten after each intervening item has been trained. Figure 2 shows the average Hamming distance from the original pattern to a settled patterns after each intervening item has been trained. As can be seen, full rehearsal is the best method while no rehearsal is the worst. The other methods fit in between with sweep rehearsal using the original training patterns being next best, followed by sweep rehearsal using pseudo patterns, random rehearsal using original patterns and random rehearsal using pseudo patterns. This is especially obvious when looking at the asymmetrically connected network (right).

The fact that sweep rehearsal using pseudo patterns performs better than random rehearsal using the original training patterns, shows that since spurious attractors were created by the process of creating the trained attractors, using these spurious patterns as training patterns themselves has the effect of preserving the originally trained patterns.



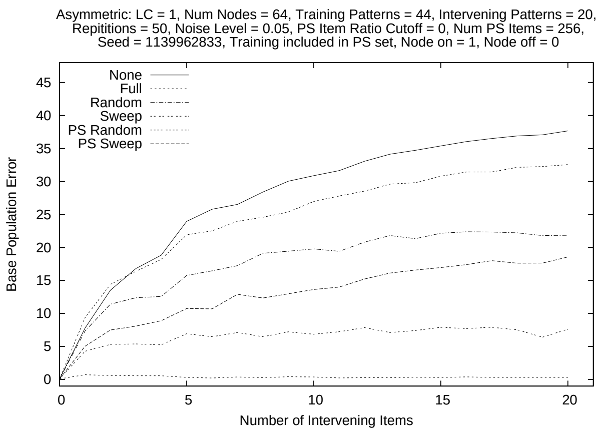
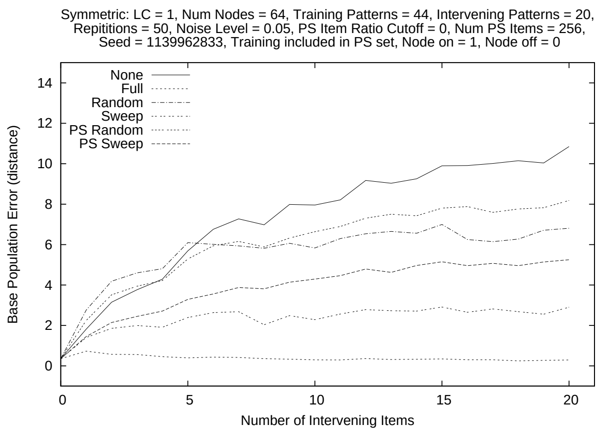


Figure 1: Showing the Number of original training patterns forgotten after each iterating item is introduced. Symmetrically connected (left) and Asymmetrically connected (right).



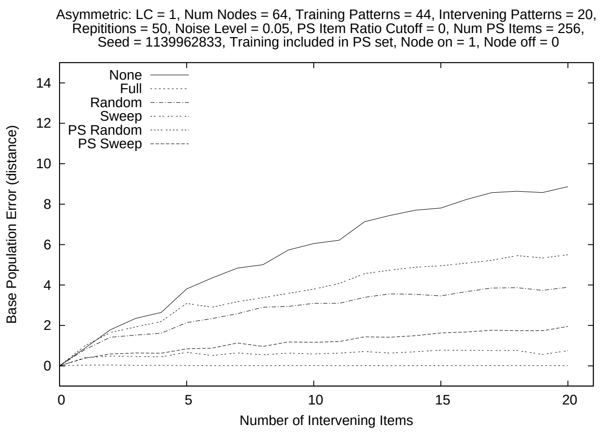


Figure 2: Showing the average Hamming distance that training patterns settle into after each iterating item is introduced. Symmetrically connected (left) and Asymmetrically connected (right).

5.2 Reproducing Previous Results

This method reproduces some of the results reported in Robins & McCallum(1998). The function commences with the construction of a new network as described in Section 3. If a pattern set is not supplied, one is created which has a number of patterns equal to the sum of the number of initial and intervening patterns supplied from the parameter file. These patterns (supplied or created) are randomly assigned to be either training or intervening patterns. A number of pseudo patterns are also generated. There are three such sets for both of the Delta trained networks. The sets used in the last two rehearsal mechanisms are created when each of the intervening item are passed to the training function.

For each iteration required, the following steps are processed:

1. All network weights are reset from the previous iteration.
2. All the networks are trained on the training patterns to criterion or for 500 epochs.
3. The current weights of all the networks are stored.
4. For each of the ten rehearsal mechanisms, repeat:
   (a) Load the original network weights and record the number of trained patterns that are stable in each network.
   (b) For each intervening item, repeat:
      i. Create new pseudo training set if required.
      ii. Train the networks on the intervening item, passing a buffer containing a number of patterns based on the rehearsal mechanism being used. Only the intervening item has an effect on stopping training. All patterns in the buffer also have an effect on the network weights but have no effect on stopping training. Once the intervening item is learnt, training stops.
      iii. Record the number of trained patterns that are stable in each network.

The buffer which is passed along with each rehearsal mechanism is described as:

1. No Rehearsal. No other patterns are passed along with the intervening item.
2. 25% Rehearsal. 25% of the original training patterns are selected at random and passed with each intervening item.
3. 50% Rehearsal. 50% of the original training patterns are selected at random and passed with each intervening item.
4. 75% Rehearsal. 75% of the original training patterns are selected at random and passed with each intervening item.
5. Full Rehearsal. All of the original training patterns are passed with each intervening item.
6. Full Rehearsal. A pseudo set of 128 stable patterns which may or may not include any of the original training patterns is passed with each intervening item
7. Full Rehearsal. A pseudo set of 256 stable patterns which may or may not include any of the original training patterns is passed with each intervening item

8. Full Rehearsal. A pseudo set of 256 stable patterns which do not include any of the original training patterns is passed with each intervening item

9. Full Rehearsal. A pseudo set of 128 stable patterns which may or may not include any of the original training patterns is generated and passed with each intervening item

10. Full Rehearsal. A pseudo set of 256 stable patterns which may or may not include any of the original training patterns is generated and passed with each intervening item

This function is affected by the following parameters. [Nodes], [Loops], [Initial], [Extras], [Pseudo], [Include-Training], [HebbianLC], [SymmetricLC], [AsymmetricLC], [PatternRatio], [NodeOn]. [NodeOff], [RatioCut], [TrainingNoise], [InputFile].

The original (1998) function when using the pseudo data sets, passed all of the available patterns along with the intervening item (methods 6 - 10 inclusive). This function however can operate in three different modes when using pseudo items. We can pass all the items (as they did), or a random rehearsal variant which selects the same number of patterns from the pseudo set that the network was originally trained on for training with each intervening item, or a sweep rehearsal where for every epoch of training we select the same number of patterns from the pseudo set that the network was trained on.

By not passing all the available pseudo items, the process of retraining and rehearsal is much faster. Using the sweep rehearsal mechanism delivers results which are achieved around 20% less time than passing all the items and yet show that pattern retention is as good as that with full training on the pseudo items (Figures 4 and 6).

The layout of the following graphs is split between a symmetrically connected delta trained naetwork on the left and an asymmetrically connected delta trained network on the right. In addition, the three forms of pseudo rehearsal are shown with full rehearsal at the top, random rehearsal in the middle and sweep rehearsal at the bottom. These different methods only make a difference in Figures 4 and 6.

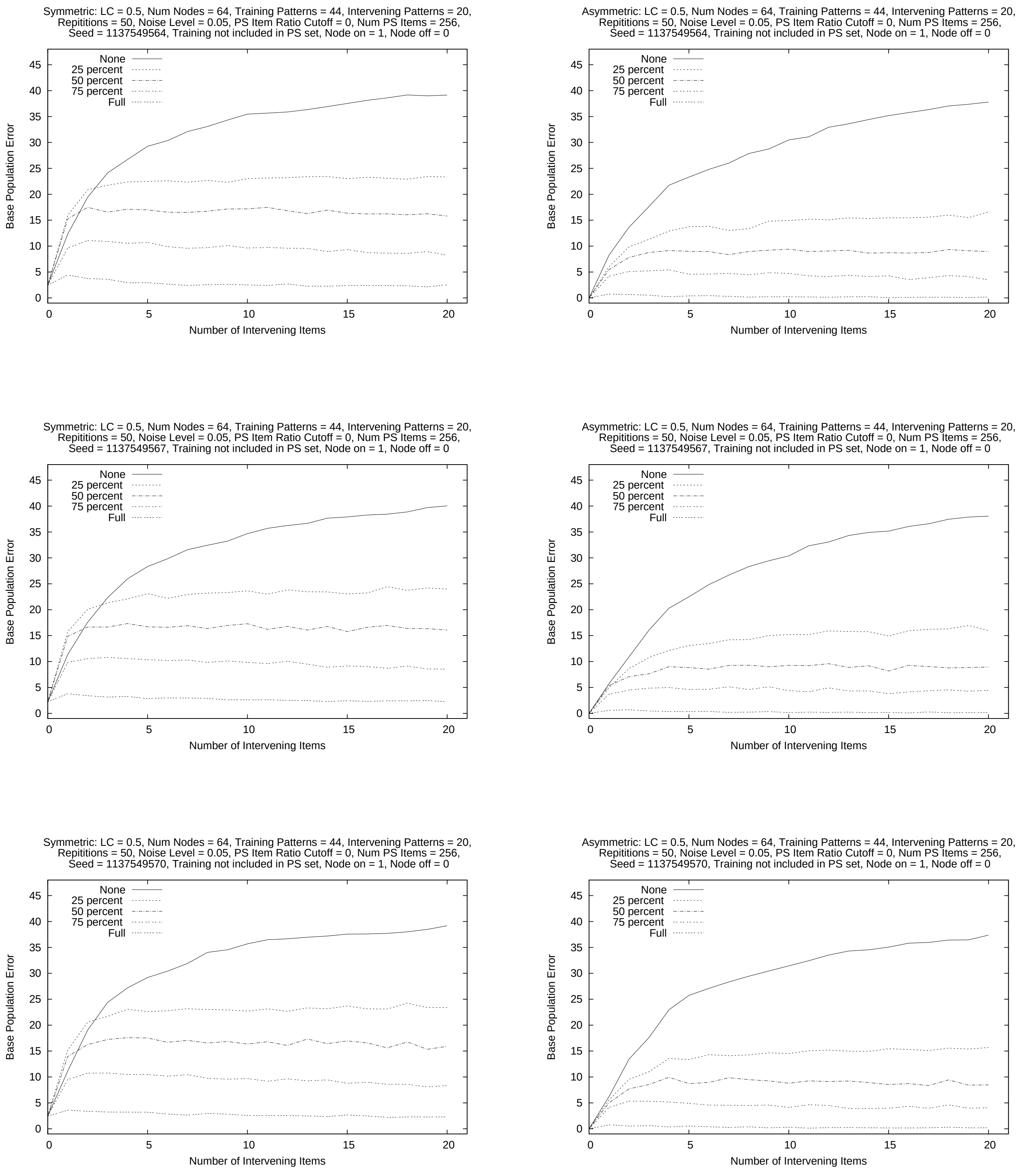


Figure 3: Symmetric (left) and Asymmetric (right), using Full (top), Random (middle) and Sweep (bottom) rehearsal mechanisms when using pseudo data sets. Shows the number of original training patterns forgotten after each intervening item has been learnt.

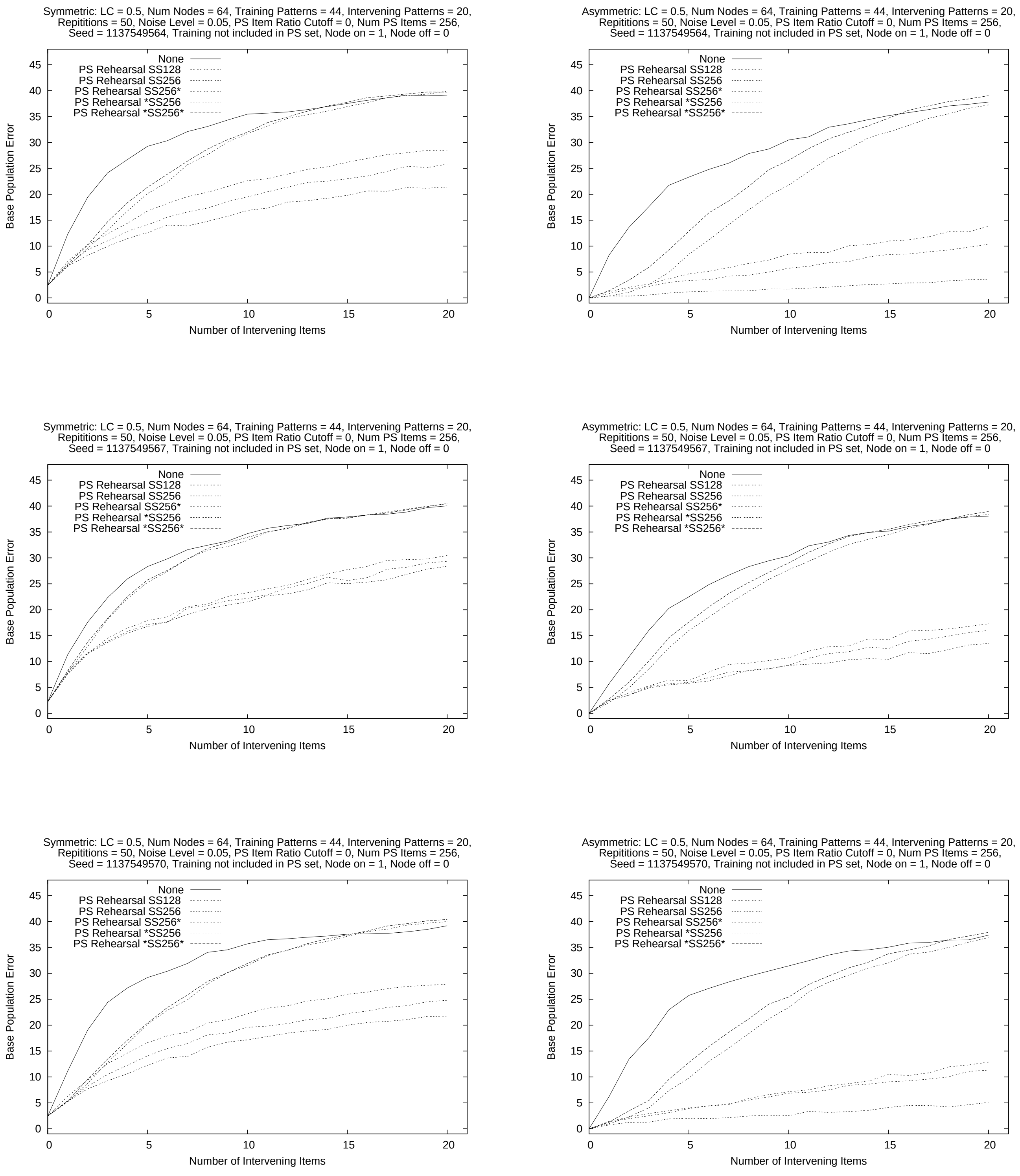


Figure 4: Symmetric (left) and Asymmetric (right), using Full (top), Random (middle) and Sweep (bottom) rehearsal mechanisms when using pseudo data sets. Shows the number of original training patterns forgotten after each intervening item has been learnt.

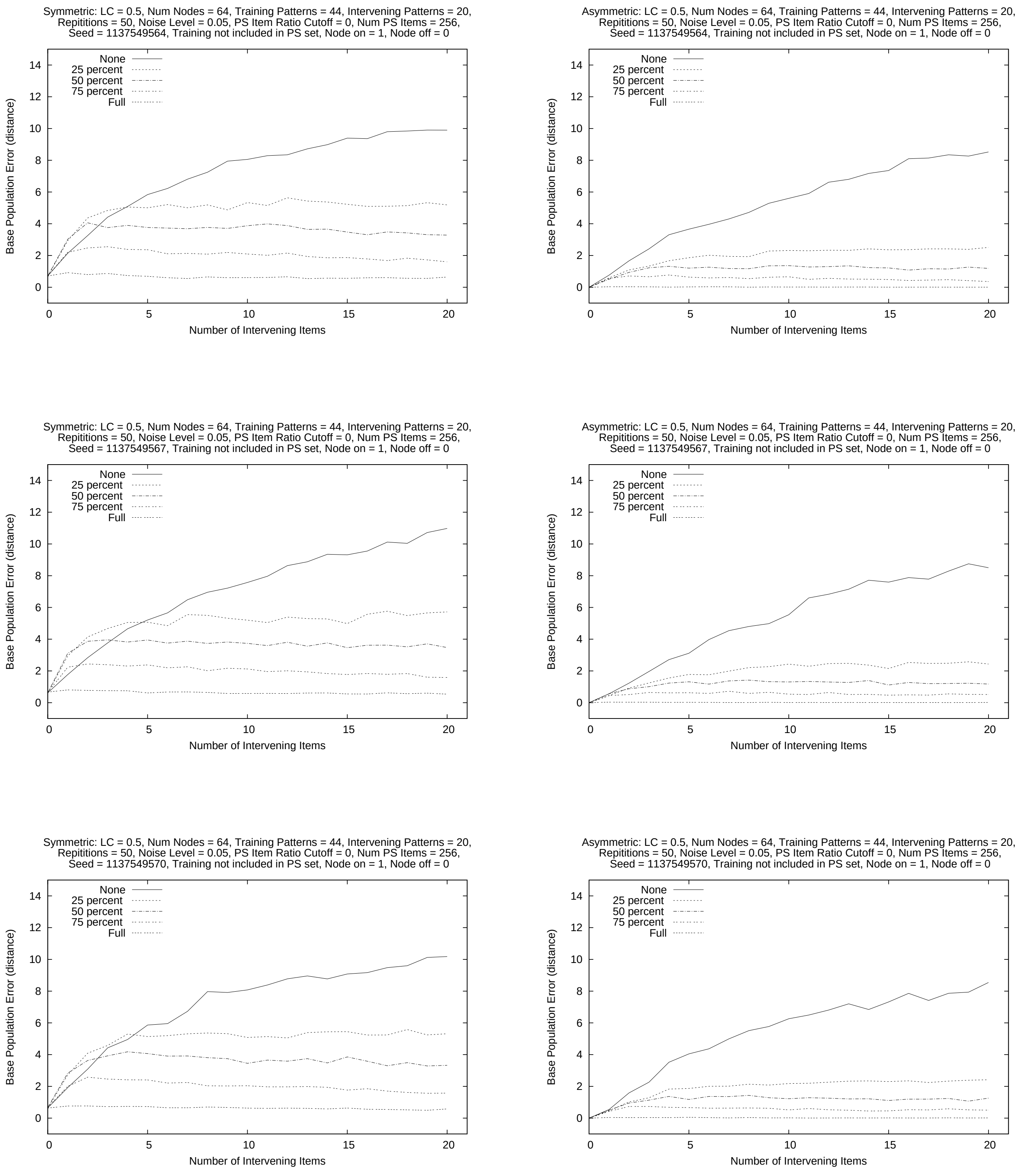
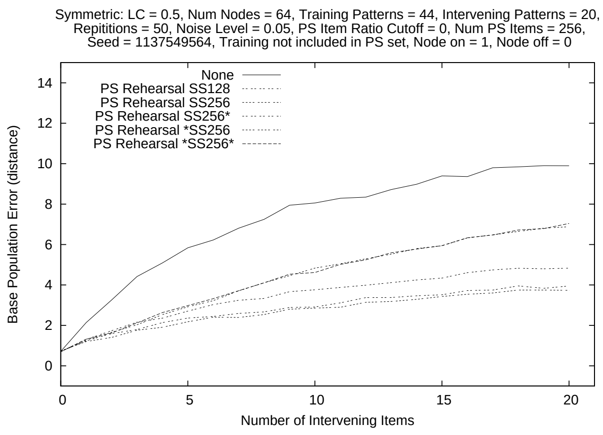
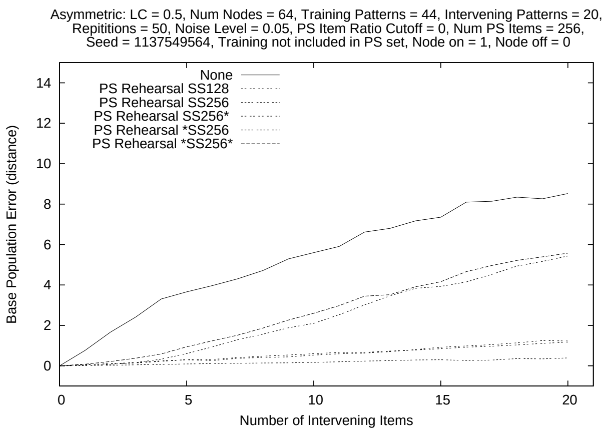
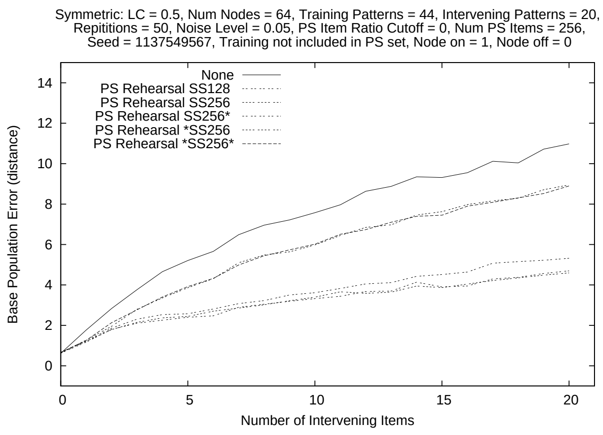
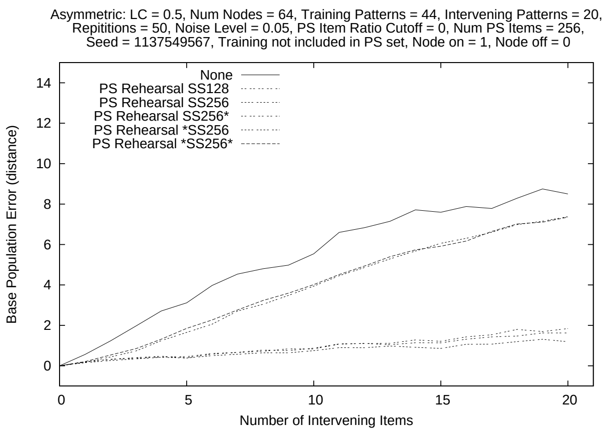


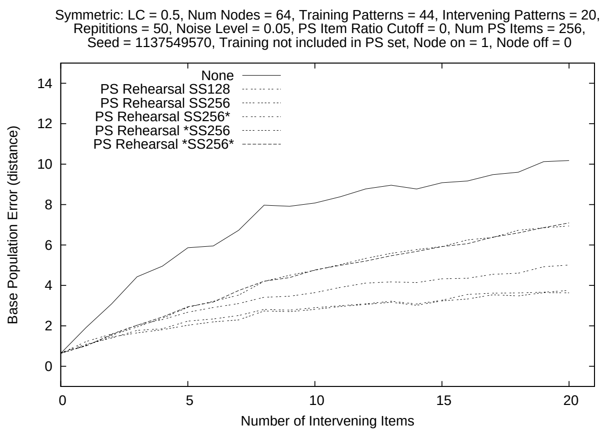
Figure 5: Symmetric (left) and Asymmetric (right), using Full (top), Random (middle) and Sweep (bottom) rehearsal mechanisms when using pseudo data sets. Shows the average Hamming distance from the original that a training pattern will settle to.











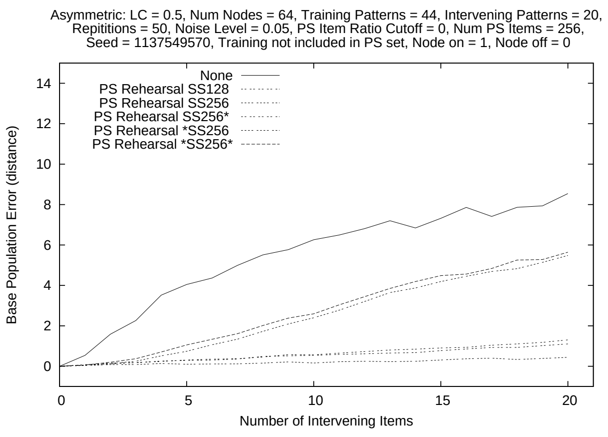


Figure 6: Symmetric (left) and Asymmetric (right), using Full (top), Random (middle) and Sweep (bottom) rehearsal mechanisms when using pseudo data sets. Shows the average Hamming distance from the original that a training pattern will settle to.

5.3 Basin Of Attraction vs Pattern Energy Profile

This method attempts to draw correlations between the depth of the basin of attraction of a pattern and the energy ratio of the pattern. The function commences with the construction of a new network as described in Section 3.

For each iteration required, the following steps are processed:

1. A set of patterns is created which is equal to the number of initial patterns supplied from the parameter file.
2. All network weights are reset from the previous iteration.
3. A number of stable patterns are generated for the two Delta trained networks.
4. Test the depth of the basin of attraction for both trained and pseudo patterns by loading each pattern. Then for each pattern:
   (a) From a depth of 0 to n, where n is the depth provided in the parameter file:
      i. Repeat 100 times:
         A. Generate a random list of all the nodes in the network.
         B. For the first n nodes in the random list, flip their activations.
         C. Asynchronously cycle the network and compare the settled pattern against the original pattern. Record this result.
      ii. Average the results.
   (b) Load the original pattern, cycle the network and record the energy profile of the network.

This function is affected by the following parameters. [Nodes], [Initial], [Pseudo], [IncludeTraining], [AsymmetricLC], [PatternRatio], [NodeOn], [NodeOff], [RatioCut], [TrainingNoise], [InputFile], [Depth].

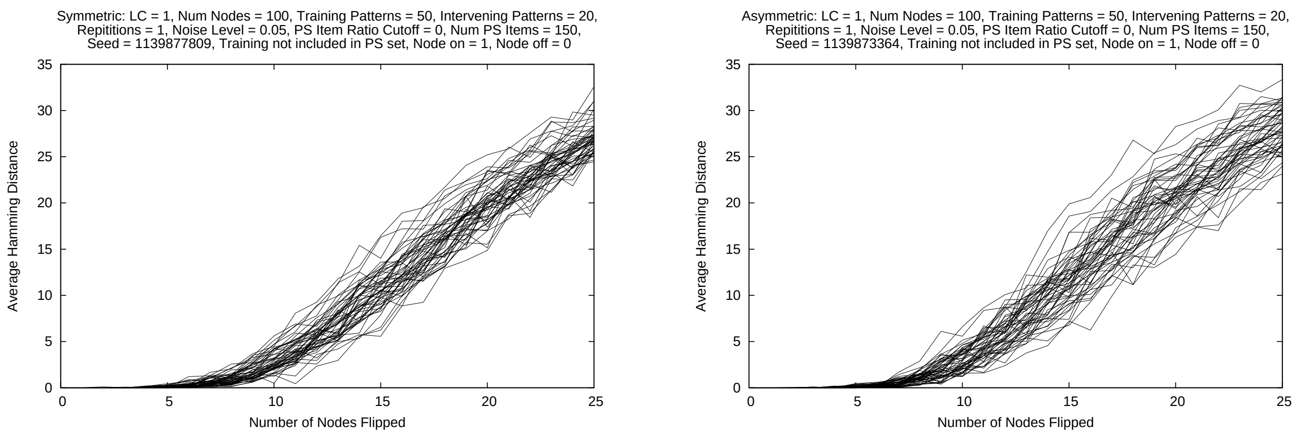


Figure 7: Basin of Attraction: The Average Hamming distance away from the original pattern where a trained pattern will settle after flipping a number of nodes.

Figure 7 shows the average Hamming distance that a trained pattern will settle to when a number of nodes are flipped. We can see here that the both the symmetrically connected network (left) and the asymmetrically connected network (right) follow similar patterns. By comparison against Figure 8, which shows the spurious

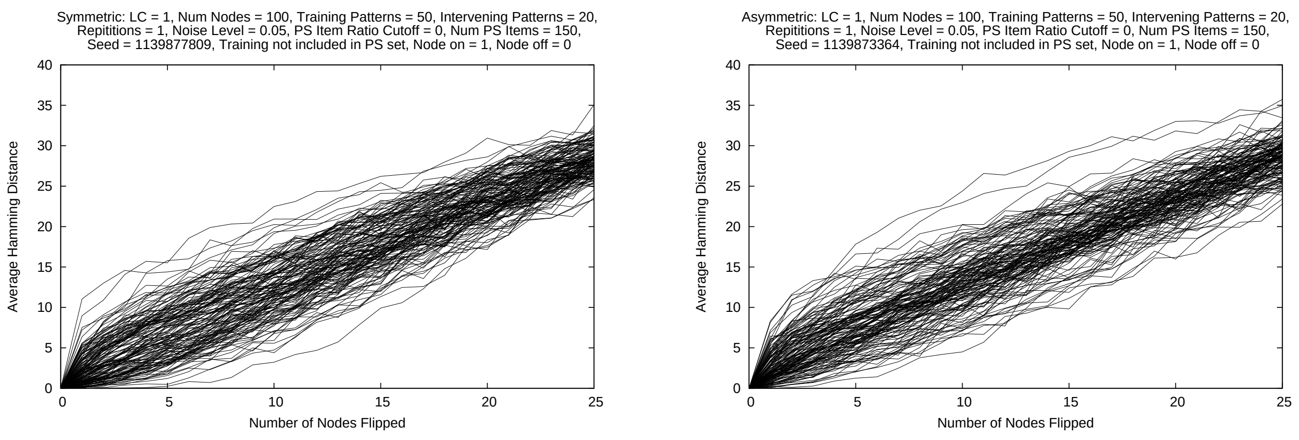


Figure 8: Basin of Attraction: The number of training patterns which do not return to their original state when a number nodes are flipped.

patterns found in both network types, we can see that the spurious patterns are more likely to settle into a pattern which is not the same as the original. This means that the size of the basin of attraction surrounding these spurious attractors must be smaller than those surrounding the trained patterns. This seems resonable since these spurious patterns mostly originate from interactions between the training patterns.

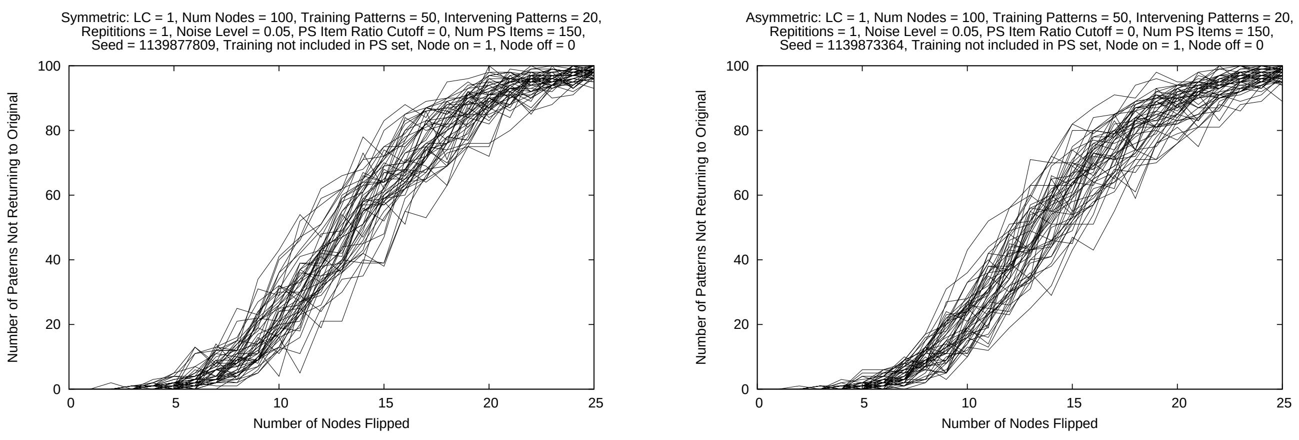


Figure 9: Basin of Attraction: The Average Hamming distance away from the original pattern where a trained pattern will settle after flipping a number of nodes

Figures 9 and 10 shows a similar pattern by comparison. Here we are looking at the number of patterns which do not return to the original. We can see that with the trained patterns (9), most are likely to return to the original, whereas the spurious patterns (10) are more likely to settle into another attractor. Both of these results leads us to believe that the size of the basin of attraction surrounding a trained pattern is larger and more stable than an attractor surrounding a spurious pattern.

Figures 11, 12 and 13 show a number of various measures of the basin of attraction plotted against the energy

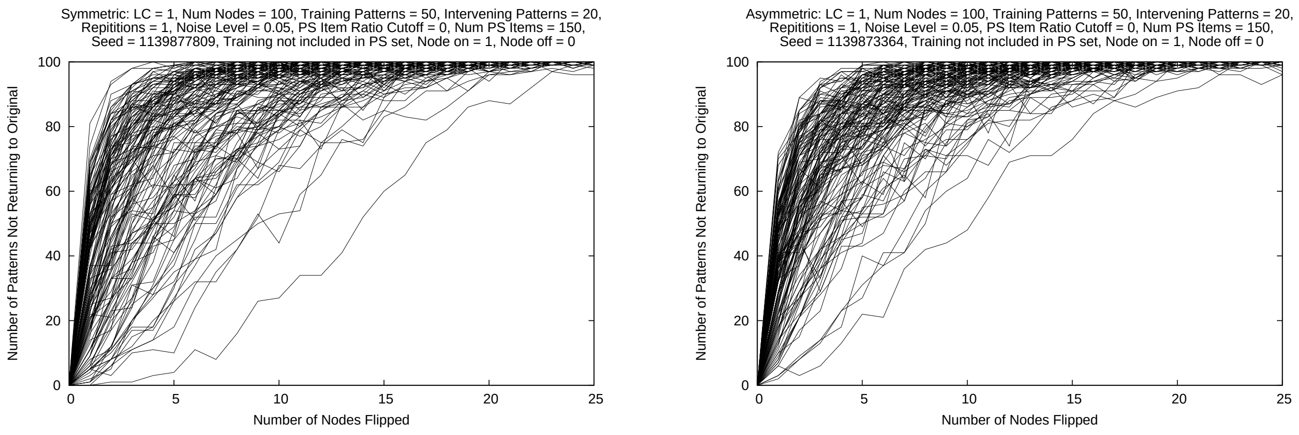


Figure 10: Basin of Attraction: The Average Hamming distance away from the original pattern where a trained pattern will settle after flipping a number of nodes

profile of the same pattern. Figure 11 shows the average Hamming distance, Figure 12 shows the average number of patterns not returning to the original and Figure 13 shows the average number of patterns which do return to the original.

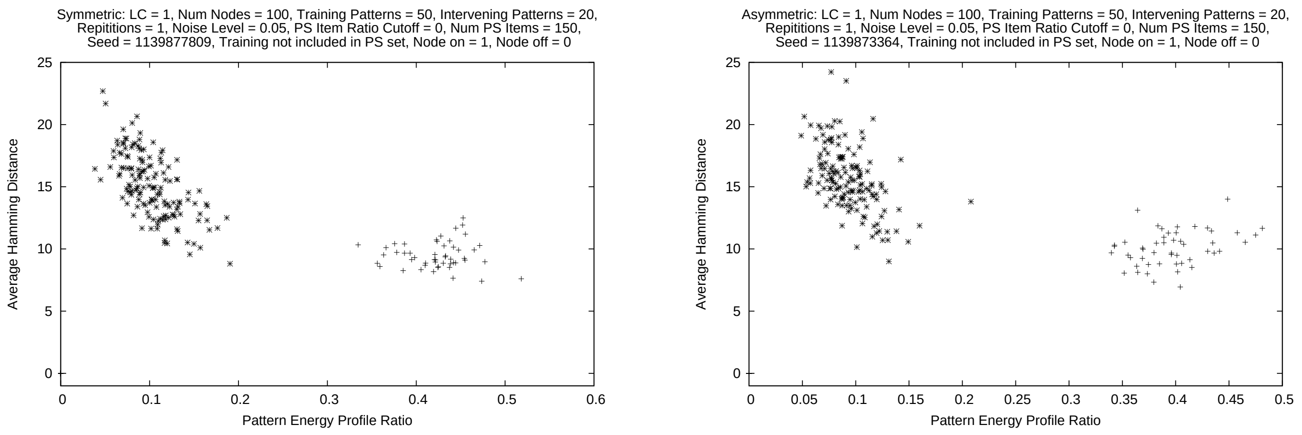


Figure 11: Basin of Attraction: The number of training patterns which do not return to their original state when a number nodes are flipped.

In all three of these graphs it is easy to see that two separate clusters are formed. In all cases these clusters are difficult to distinguish based on the y axis. The energy profile (x axis) however shows a strong gap between the clusters. This gap is present up to a loading of about 70% where a network consisting of 100 nodes has been trained on 70 random patterns. After this point the two clusters become inseparable by means of the energy profile. The network shown here has a pattern loading of 50%.

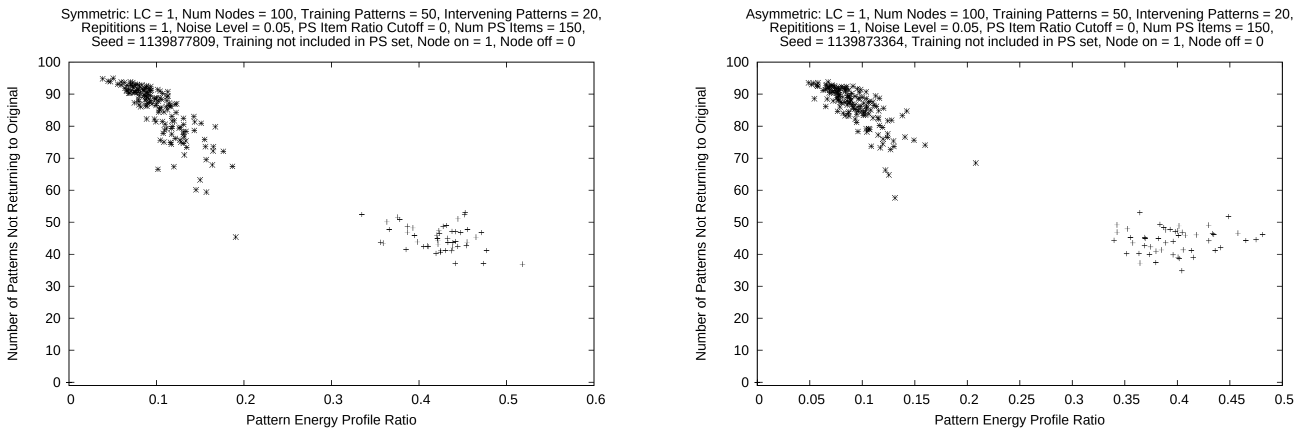


Figure 12: Basin of Attraction: The number of training patterns which return to their original state when a number nodes are flipped.

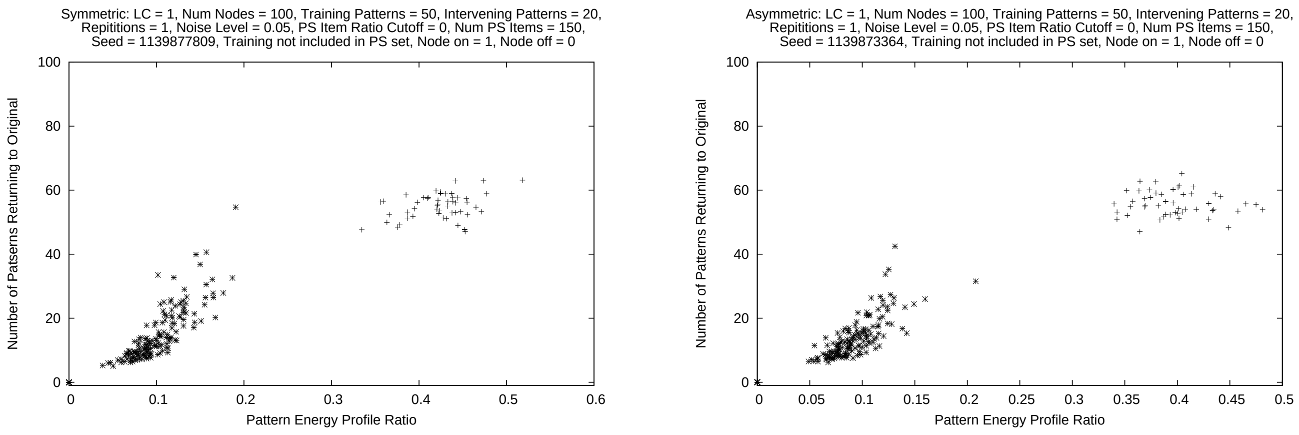


Figure 13: Basin of Attraction: The number of training patterns which return to their original state when a number nodes are flipped.

5.4 Pattern Prototyping

This function attempts to get all three networks to create a desired stable attractor which the network was not trained on. The function commences with the construction of a new network as described in Section 3.

For each iteration required, the following steps are processed:

1. From the start value to the end value in the provided number of steps (all set in the parameter file):
    (a) Create a number of training patterns. These training patterns are based on a prototype pattern which has a number of nodes turned on and all others turned off (proto ratio in parameter file). There may also be more than one prototype pattern. In this case the training patterns are split up as evenly as possible and each group is based on a different prototype. The training patterns differ from the prototypes by flipping the activation of each node by a given probability (the value of this loop).

    (b) Train the networks on the training patterns.

    (c) A number of pseudo patterns are found and stored. These sets do not include training patterns but may contain the prototype pattern.

    (d) For each network type:

        i. Load each prototype pattern, cycle and record network energy profile.

        ii. Load each trained pattern, cycle and record network energy profile.

        iii. Load each pseudo pattern, cycle and record network energy profile.

This function is affected by the following parameters. [Nodes], [Loops], [Initial], [Pseudo], [IncludeTraining], [HebbianLC], [SymmetricLC], [AsymmetricLC], [NodeOn], [NodeOff], [RatioCut], [TrainingNoise], [InputFile], [ProtoRatio], [Min], [Max], [Steps], [Prototypes].

The results can be graphed either by individual runs where the energy ratios are displayed as bar graphs, or as averages over all the runs where the average trained and spurious ratios are shown along with each of the prototypes.

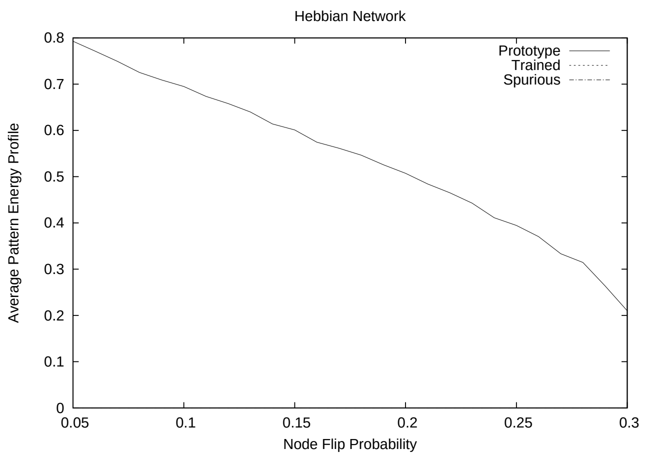
Figure 14 shows the results from a run of this function. The graphs are from the Hebbian trained symmetric (top), Delta trained symmetric (middle) and the Delta trained asymmetric (bottom) networks. The network consists of 100 nodes and is trained on 50 patterns with a prototype pattern of 20 nodes on and 80 nodes off. The x axis shows the probability of any node in a training pattern being flipped from the prototype. This can be thought of as a low probability meaning that the training patterns are similar to the prototype while with a large probability the training patterns bear little resemblance.

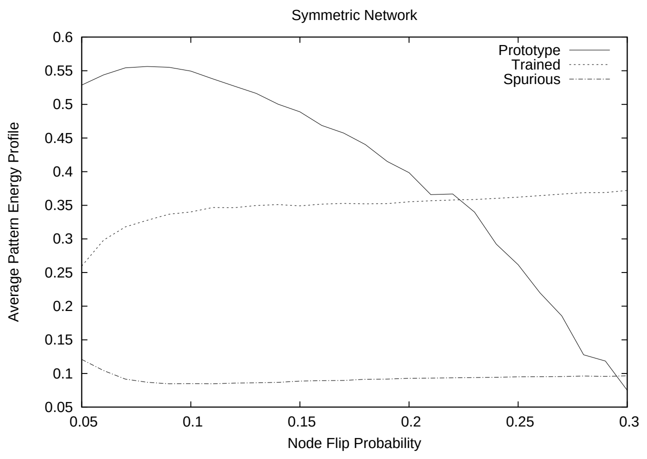
As can be seen, the prototype is a strong attractor in the weight space even though none of the networks were trained on it. As the training patterns move away from the prototype, we see an gradual decrease in the strength of the energy profile of the prototype.

Figures 15 and 16 show the relative energy of each pattern during a run. The runs consist of only six stopping points. Figure 15 shows the symmetrically connected Delta trained network while Figure 16 shows the asymmetrically connected delta trained network. In both cases the graphs show each stopping point moving from the top left (starting point) to the bottom right (end point). The graphs show that as the training pattern variance from prototype increases, the energy profile of the spurious patterns becomes markedly lower than that of the trained patterns. Up to a ratio of about 0.2, the prototype is a strong attractor in both the symmetric and asymmetric networks. After this point the variance in the training patterns is to great to create a strong stable prototype attractor.

6 References

Robins, A. & McCallum, S. Catastrophic forgetting and the pseudorehearsal solution in Hopfield type networks. *Connection Science: Journal of Neural Computing, Artificial Intelligence and Cognitive Research, 10* : 121 - 135 (1998)





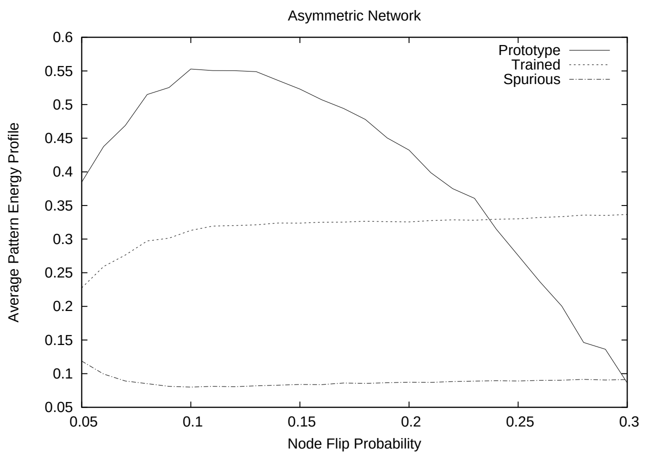


Figure 14: Prototyping Graphs. Showing the energy profile from the Hebbian trained symmetric network (top), Delta trained symmetric network (middle) and Delta trained asymmetric netwok (bottom). The graphs show the energy of the prototype , the average energy of a trained pattern and the average energy of the spurious patterns.

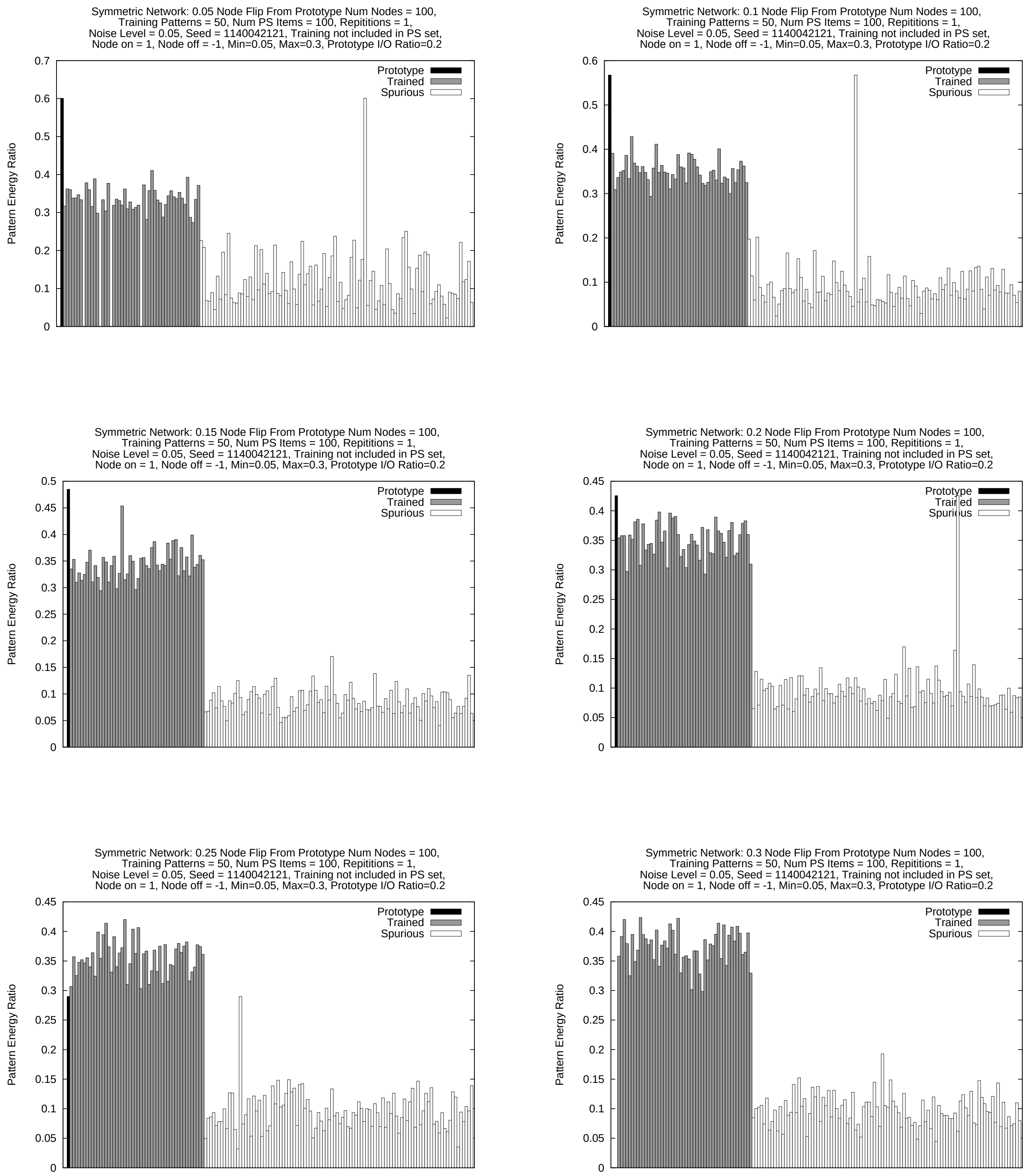


Figure 15: A single run of the prototype algorithm on the symmetrically connected Delta trained network.

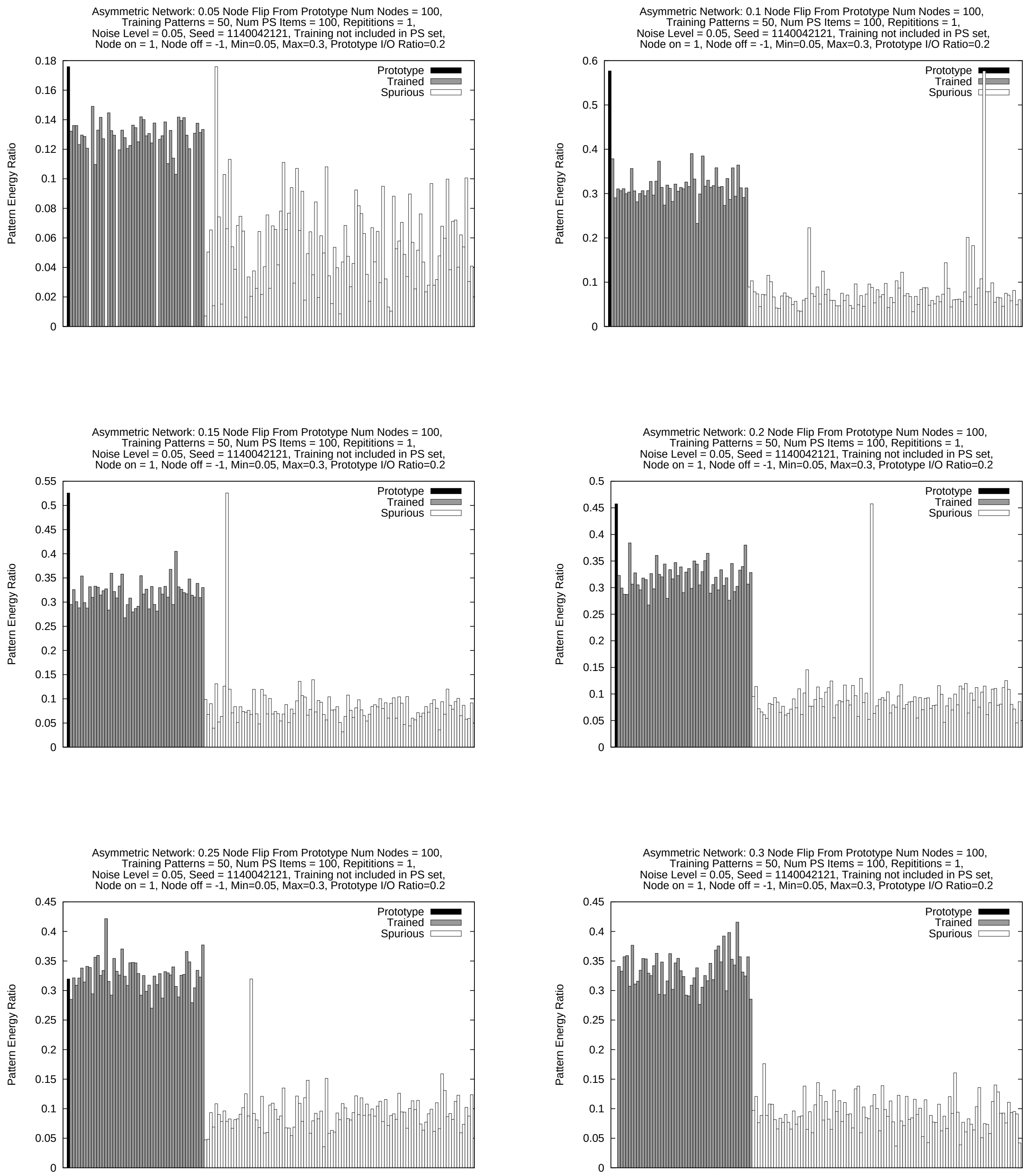


Figure 16: A single run of the prototype algorithm on the asymmetrically connected Delta trained network.