# 10

# Parity-*N* Problems as a Vehicle to Compare Efficiency of Neural Network Architectures

Bogdan M. Wilamowski
*Auburn University*

Hao Yu
*Auburn University*

Kun Tao Chung
*Auburn University*

## 10.1 Introduction

Parity-*N* problems have been studied deeply in many literatures [WH03,HLS99]. The *N*-bit parity function can be interpreted as a mapping (defined by $2^N$ binary vectors) that indicates whether the sum of the *N* elements of every binary vector is odd or even. It is shown that threshold networks with one hidden layer require *N* hidden threshold units to solve the parity-*N* problem [M61,HKP91]. If the network has bridged connections across layers, then the number of hidden threshold units can be reduced by half. In this case, only *N*/2 neurons are required in the hidden layer for the parity-*N* problem [M61]. After that, Paturi and Saks [PS90] showed that only $N/\log_2 N$ neurons are required. Siu, Roychowdhury, and Kailath [SRT91] showed that when one more hidden layer is introduced, the total number of hidden units could be only $2\sqrt{N}$.

In this chapter, the parity-*N* problem is solved by different networks, so as to compare the efficiency of neural architecture.

One may notice that, in parity problems, the same value of sum of all inputs results with the same outputs. Therefore, considering all the weights on network inputs as “1,” the number of training patterns of parity-*N* problem can be reduced from $2^N$ to $N$+1.

Figure 10.1 shows both the original eight training patterns and the simplified four training patterns, which are identical.

Based on this pattern simplification, a linear neuron (with slope equal to 1) can be used as the network input (see Figure 10.2b). This linear neuron works as a summator. It does not have bias input and does not need to be trained.

| Input | Sum of inputs | Output |
|---|---|---|
| 0 0 0 | 0 | 0 |
| 0 0 1 | 1 | 1 |
| 0 1 0 | 1 | 1 |
| 0 1 1 | 2 | 0 |
| 1 0 0 | 1 | 1 |
| 1 0 1 | 2 | 0 |
| 1 1 0 | 2 | 0 |
| 1 1 1 | 3 | 1 |

(a)

| Input | Output |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 0 |
| 3 | 1 |

(b)

**FIGURE 10.1** Training simplification for the parity-3 problem: (a) original patterns and (b) simplified patterns.

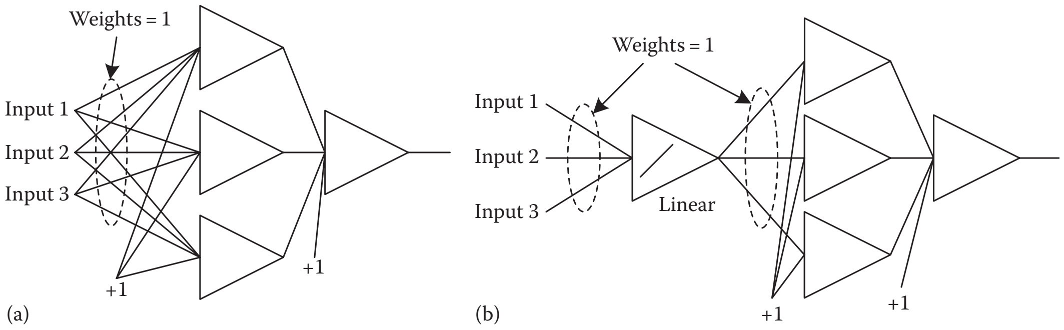

**FIGURE 10.2** Two equivalent networks for the parity-3 problem: (a) parity-3 inputs and (b) linear neuron inputs.

## 10.2 MLP Networks with One Hidden Layer

Multilayer perceptron (MLP) networks are the most popular networks, because they are regularly formed and easy for programming. In MLP networks, neurons are organized layer by layer and there are no connections across layers.

Both parity-2 (XOR) and parity-3 problems can be visually illustrated in two and three dimensions respectively, as shown in Figure 10.3.

Similarly, using MLP networks with one hidden layer to solve the parity-7 problem, there could be at least seven neurons in the hidden layer to separate the eight training patterns (using a simplification described in introduction), as shown in Figure 10.4a.

In Figure 10.4a, eight patterns {0, 1, 2, 3, 4, 5, 6, 7} are separated by seven neurons (bold line). The thresholds of the hidden neurons are {0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5}. Then summing the outputs of hidden neurons weighted by {1, −1, 1, −1, 1, −1, 1}, the net inputs at the output neurons could be only {0, 1}, which can be separated by the neuron with threshold 0.5. Therefore, parity-7 problem can be solved by the architecture shown in Figure 10.4b.

Generally, if there are $n$ neurons in MLP networks with a single hidden layer, the largest possible parity-$N$ problem that can be solved is

$$N = n - 1 \tag{10.1}$$

where

$n$ is the number of neurons

$N$ is the parity index

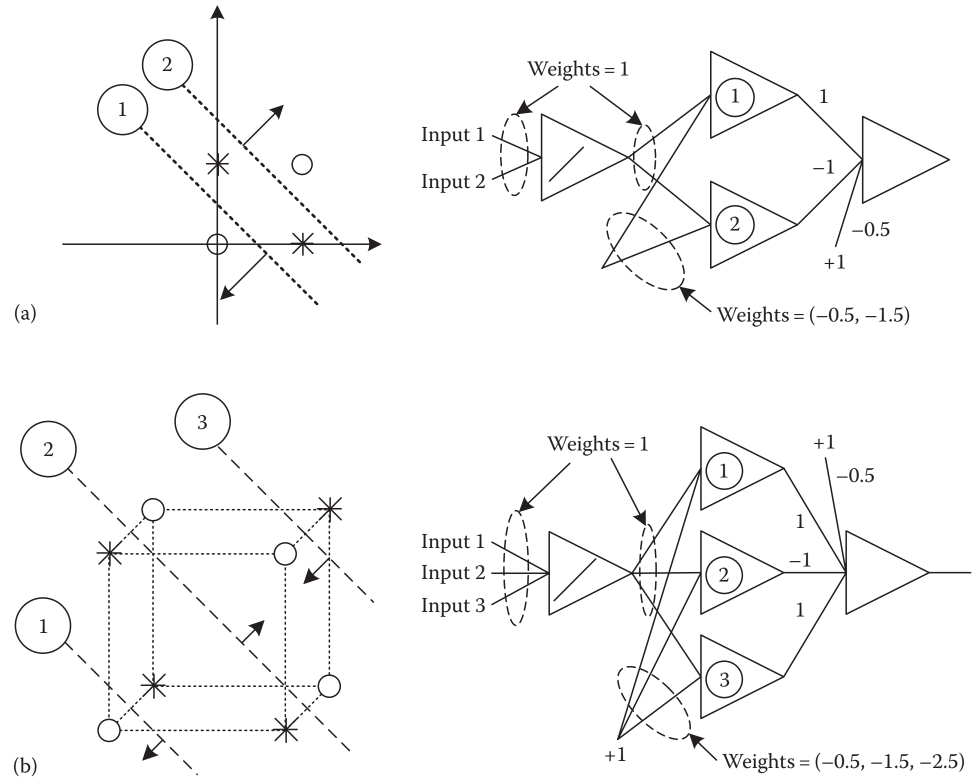

**FIGURE 10.3** Graphical interpretation of pattern separation by hidden layer and network implementation using unipolar neurons for (a) XOR problem and (b) parity-3 problem.

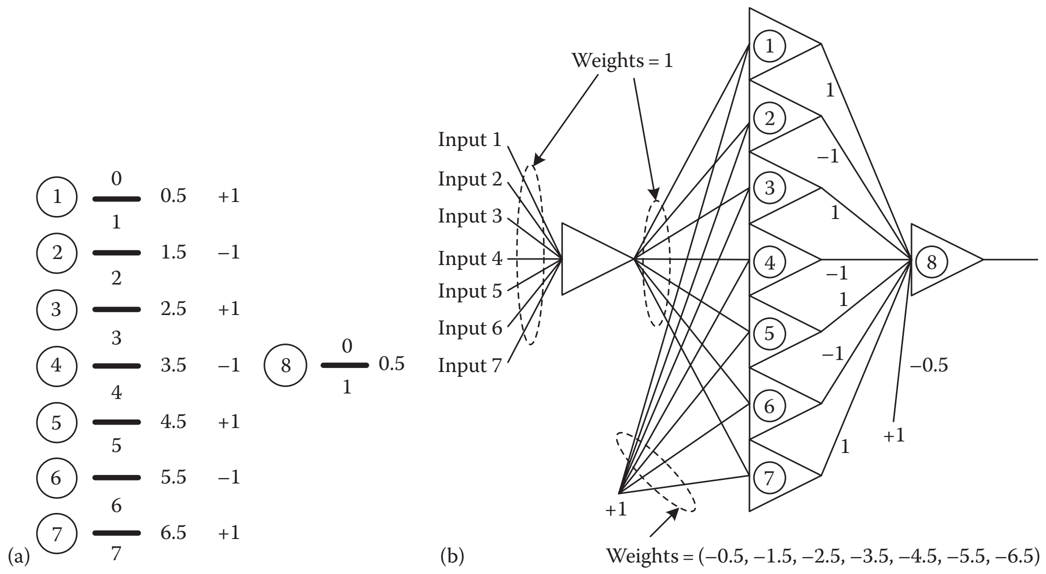

**FIGURE 10.4** Solving the parity-7 problem using MLP network with one hidden layer: (a) analysis and (b) architecture.

## 10.3 BMLP Networks

In MLP networks, if connections across layers are permitted, then networks have bridged multilayer perceptron (BMLP) topologies. BMLP networks are more powerful than traditional MLP networks.

### 10.3.1 BMLP Networks with One Hidden Layer

Considering BMLP networks with only one hidden layer, all network inputs are also connected to the output neuron or neurons.

For the parity-7 problem, the eight simplified training patterns can be separated by three neurons to four subpatterns {0, 1}, {2, 3}, {4, 5}, and {6, 7}. The threshold of the hidden neurons should be {1.5, 3.5, 5.5}. In order to transfer all subpatterns to the unique pattern {0, 1} for separation, patterns {2, 3}, {4, 5}, and {6, 7} should be reduce by 2, 4, and 6 separately, which determines the weight values on connections between hidden neurons and output neurons. After pattern transformation, the unique pattern {0, 1} can be separated by the output neuron with threshold 0.5. The design process is shown in Figure 10.5a and the corresponding solution architecture is shown in Figure 10.5b.

For the parity-11 problem, similar analysis and related BMLP networks with single hidden layer solution architecture are presented in Figure 10.6.

Generally, for $n$ neurons in BMLP networks with one hidden layer, the largest parity-$N$ problem that can be possibly solved is

$$N = 2n - 1 \tag{10.2}$$

### 10.3.2 BMLP Networks with Multiple Hidden Layers

If BMLP networks have more than one hidden layer, then the further reduction of the number of neurons are possible, for solving the same problem.

For the parity-11 problem, using 4 neurons, in both 11 = 2 = 1 = 1 and 11 = 1 = 2 = 1 architectures, can find solutions.

Considering the 11 = 2 = 1 = 1 network, the 12 simplified training patterns would be separated by two neurons at first, into {0, 1, 2, 3}, {4, 5, 6, 7}, and {8, 9, 10 11}; the thresholds of the two neurons are 3.5 and 7.5, separately. Then, subpatterns {4, 5, 6, 7} and {8, 9, 10, 11} are transformed to {0, 1, 2, 3} by subtracting −4

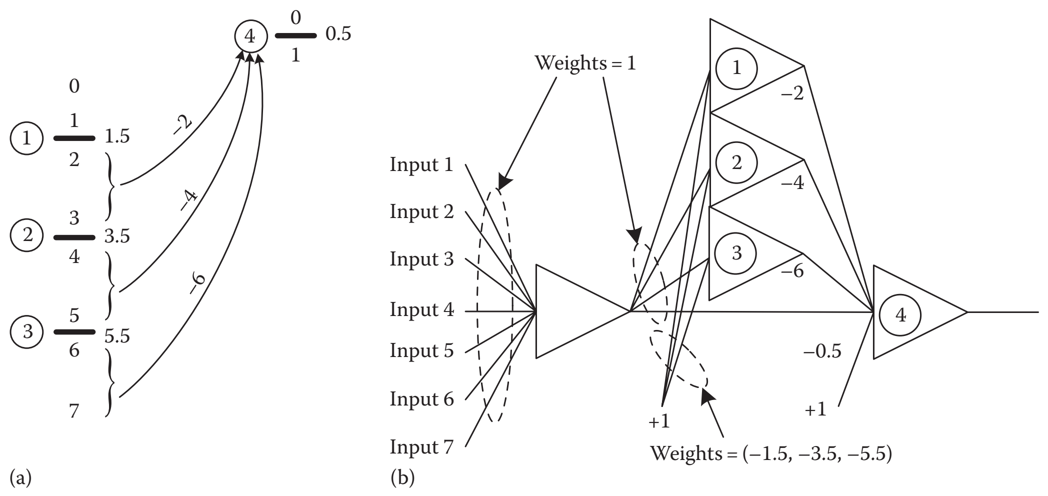

**FIGURE 10.5** Solving the parity-7 problem using BMLP networks with one hidden layer: (a) analysis and (b) architecture.

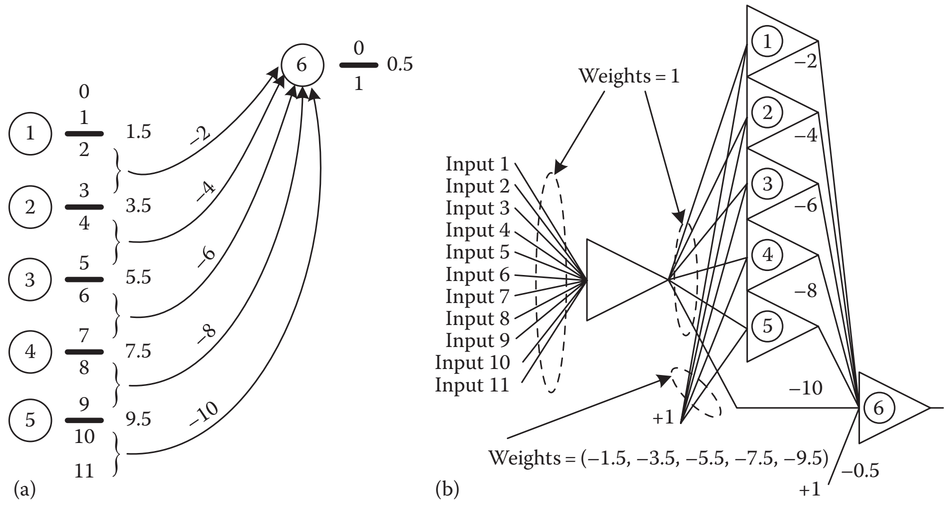

**FIGURE 10.6** Solving the parity-11 problem using BMLP networks with single hidden layer: (a) analysis and (b) architecture.

and –8 separately, which determines the weight values on connections between the first hidden layer and followed layers. In the second hidden layer, one neuron is introduced to separate {0, 1, 2, 3} into {0, 1} and {2, 3}, with threshold 1.5. After that, subpattern {2, 3} is transferred to {0, 1} by setting weight value as –2 on the connection between the second layer and the output layer. At last, output neuron with threshold 0.5 separates the pattern {0, 1}. The whole procedure is presented in Figure 10.7.

Figure 10.8 shows the 11 = 1 = 2 = 1 BMLP network with two hidden layers, for solving the parity-11 problem.

Generally, considering the BMLP network with two hidden layers, the largest parity-*N* problem can be possibly solved is

$$N = 2(m+1)(n+1) - 1 \tag{10.3}$$

where *m* and *n* are the numbers of neurons in the two hidden layers, respectively.

For further derivation, one may notice that if there are *k* hidden layers and $n_i$ is the number of neurons in related hidden layer, where *i* is ranged from 1 to *k*, then

$$N = 2(n_1+1)(n_2+1)\cdots(n_{k-1}+1)(n_k+1) - 1 \tag{10.4}$$

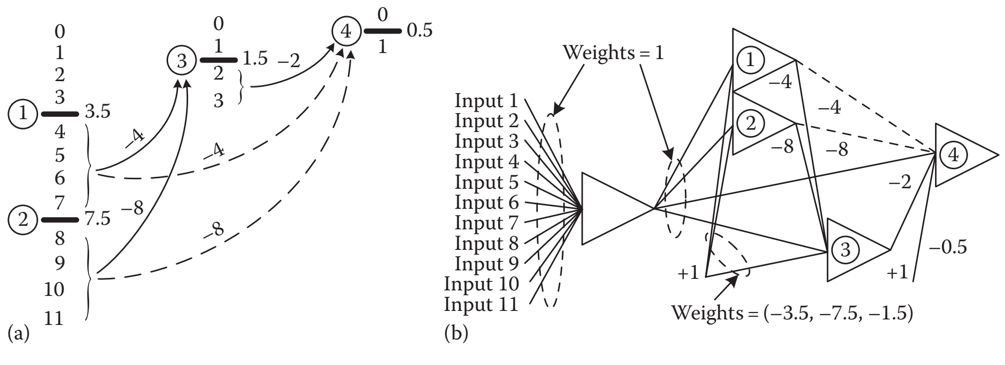

**FIGURE 10.7** Solving the parity-11 problem using BMLP networks with two hidden layers, 11=2=1=1: (a) analysis and (b) architecture.

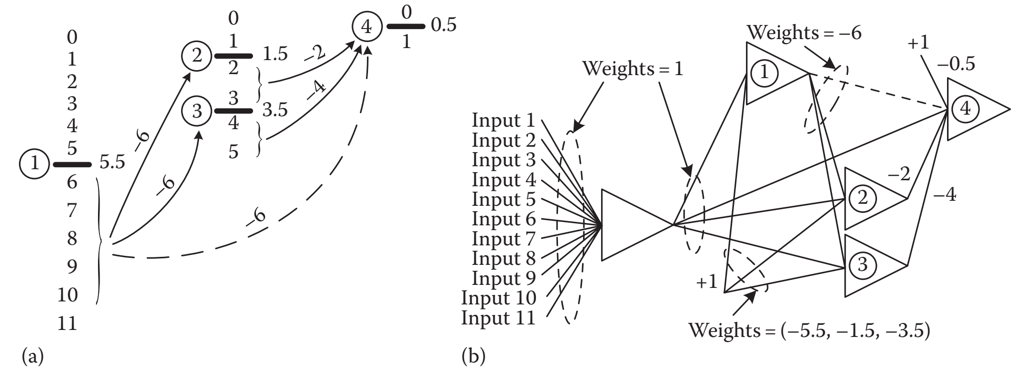

**FIGURE 10.8** Solving the parity-11 problem using BMLP networks with two hidden layers, 11=1=2=1: (a) analysis and (b) architecture.

## 10.4 FCC Networks

Fully connected cascade (FCC) networks can solve problems using the smallest possible number of neurons. In the FCC networks, all possible routines are weighted, and each neuron contributes to a layer.

For parity-7 problem, the simplified 8 training patterns are divided by one neuron at first, as {0, 1, 2, 3} and {4, 5, 6, 7}; the threshold of the neuron is 3.5. Then the subpattern {4, 5, 6, 7} is transferred to {0, 1, 2, 3} by weights equal to –4, connected to the followed neurons. Again, by using another neuron, the patterns in the second hidden layer {0, 1, 2, 3} can be separated as {0, 1} and {2, 3}; the threshold of the neuron is 1.5. In order to transfer the subpattern {2, 3} to {1, 2}, 2 should be subtracted from subpattern {2, 3}, which determines that the weight between the second layer and the output layer is –2. At last, output neurons with threshold 0.5 is used to separate the pattern {0, 1}, see Figure 10.9.

Figure 10.10 shows the solution of parity-15 problem using FCC networks.

Considering the FCC networks as special BMLP networks with only one neuron in each hidden layer, for *n* neurons in FCC networks, the largest *N* for parity-*N* problem can be derived from Equation 10.4 as

$$N = 2\underbrace{(1+1)(1+1)\cdots(1+1)(1+1)}_{n-1} - 1 \tag{10.5}$$

or

$$N = 2^n - 1 \tag{10.6}$$

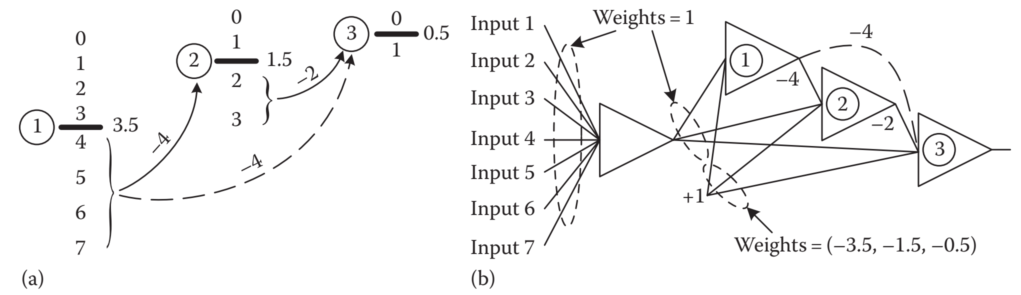

**FIGURE 10.9** Solving the parity-7 problem using FCC networks: (a) analysis and (b) architecture.

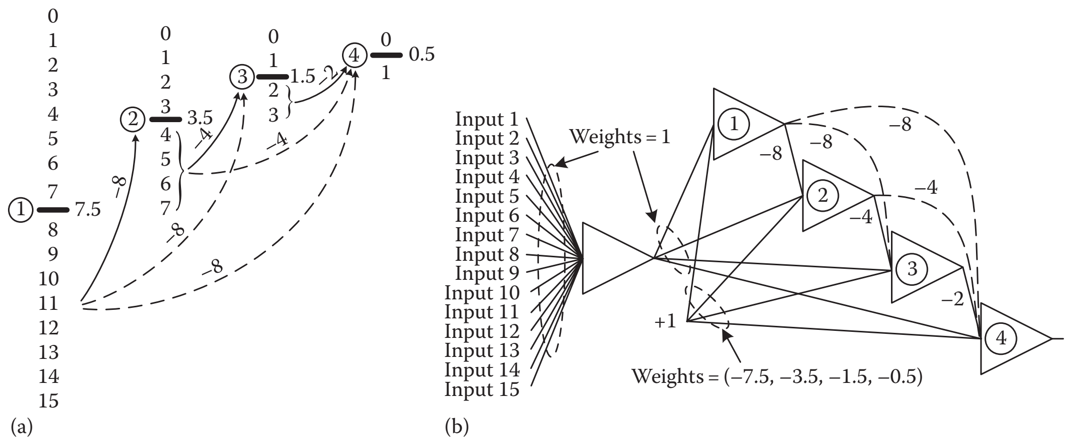

**FIGURE 10.10** Solving the parity-15 problem using FCC networks: (a) analysis and (b) architecture.

**TABLE 10.1** Different Architectures for Solving the Parity-$N$ Problem

| Network Structure | Parameters | Parity-$N$ Problem |
|---|---|---|
| MLP with single hidden layer | $n$ neurons | $n-1$ |
| BMLP with one hidden layer | $n$ neurons | $2n+1$ |
| BMLP with multiple hidden layers | $h$ hidden layers, each with $n_i$ neurons | $2(n_1+1)(n_2+1)\cdots(n_{h-1}+1)(n_h+1)-1$ |
| FCC | $n$ neurons | $2^n-1$ |

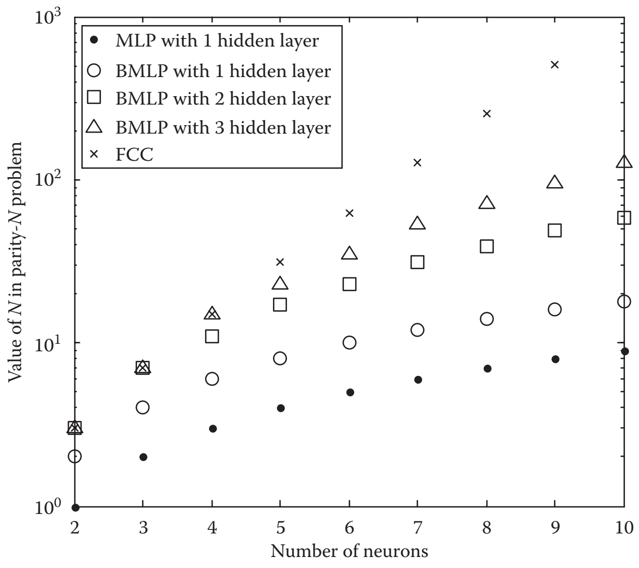

**FIGURE 10.11** Efficiency comparison among various neural network architectures.

## 10.5 Comparison of Topologies

Table 10.1 concludes the analysis above, for the largest parity-*N* problem that can be solved with a given network structure.

Figure 10.11 shows comparisons of the efficiency of various neural network architectures.

## 10.6 Conclusion

This chapter analyzed the efficiency of different network architectures, using parity-*N* problems. Based on the comparison in Table 10.1 and Figure 10.11, one may notice that, for the same number of neurons, FCC networks are able solve parity-*N* problems with the least number of neurons than other architectures.

However, FCC networks also have the largest number of layers and this makes them very difficult to be trained. For example, few algorithms can be so powerful to train the parity-*N* problem with the given optimal architectures, such as 4 neurons for the parity-15 problem. So, the reasonable architecture would be the BMLP network with couple hidden layers.

AQ1

## References

[AW95] T. J. Andersen, and B. M. Wilamowski, A modified regression algorithm for fast one layer neural network training, *World Congress of Neural Network*s, vol. 1, pp. 687–690, Washington, DC, July 17–21, 1995.

[HKP91] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, MA, 1991.

[HLS99] M. E. Hohil, D. Liu, and S. H. Smith, Solving the N-bit parity problem using neural networks, *Neural Networks,* 12, 1321–1323, 1999.

[M61] R. C. Minnick, Linear-input logic, *IRE Transactions on Electronic Computers*, EC-10, 6–16, March 1961.

AQ2 [PS90] R. Paturi and M. Saks, On threshold circuits for parity, *IEEE Symposium on Foundations of Computer Science*, pp. 397–404, October 1990.

[SRT91] K. Y. Siu, V. Roychowdhury, and T. Kailath, Depth size trade off for neural computation, *IEEE Transactions on Computers*, 40, 1402–1412, December 1991.

[W09] B. M. Wilamowski, Neural network architectures and learning algorithms, *IEEE Industrial Electronics Magazine*, 3(4), 56–63, 2009.

[WH03] B. M. Wilamowski and D. Hunter, Solving parity-N problems with feedforward neural network, *Proceedings of the IJCNN'03 International Joint Conference on Neural Networks*, pp. 2546–2551, Portland, OR, July 20–23, 2003.

AQ3 [WH10] B. M. Wilamowski and H. Yu, Improved computation for Levenberg Marquardt training, *IEEE Transactions on Neural Networks*, 21, 2010.

[WT93] B. M. Wilamowski and L. Torvik, Modification of gradient computation in the back-propagation algorithm, Presented at *ANNIE'93—Artificial Neural Networks in Engineering*, St. Louis, MO, November 14–17, 1993.