Neural Network Strategies for Solving Synthesis Problems in Non-conflict Cases in Distributed Expert Systems

Minjie Zhang and Chengqi Zhang

Department of Mathematics, Statistics and Computing Science
University of New England, Armidale, N.S.W. 2351, Australia
{minjie,chengqi }@neumann.une.edu.au

Abstract

In this paper, two neural network mechanisms for synthesis of solutions in non-conflict cases in distributed expert systems (DESs) are proposed. The ideas are: inputs of the neural network are different solutions for the same problem from different expert systems in DESs; outputs of the neural network are the final solutions for the problem after combining different solutions which should be the same as the human experts' final solutions. The first point is to set up the architecture of the neural network and train the neural network by adjusting weights of links to match the outputs of the neural network against the human experts' solutions for all patterns. The second point is that the neural network mechanism proposed in this paper can accommodate the variable number of inputs and outputs without changing neural network architecture.

1 Introduction

In a distributed expert system (DES), each cooperative member is an expert system (ES) with its own domain knowledge and inference engine. Each ES can work individually for solving some specific problems and can also cooperate with other expert systems (ESs) when they deal with complex problems.

Due to the limitation of knowledge, the ability of problem solving in single ESs, and the uncertain features of problems, some complex tasks need to be solved by diverse expertises in order to increase reliability of the solution. A typical example is when "several doctors diagnose the same patient".

If a subtask is allocated to more than one ES, each ES could obtain a solution. For example, two expert systems (ESs) predict a potential earthquake in a particular area. ES_1 believes that the possibility of the potential earthquake being class 4 is $x_1 = 0.8$, while ES_2 believes that the possibility of the potential earthquake being class 4 is $x_2 = 0.6$. It is not good practice for a DES to give a user diverse solutions to the same subtask. The problem here is how to obtain the final uncertainty if more than one uncertainty for the same solution exists. The synthesis strategies are responsible for synthesizing the uncertainties of the solution from different ESs to produce the final uncertainty of the solution.

Consider the following two cases based on the above example.

Case (1) Two ESs obtain the uncertainties of the solution (a class 4 earthquake in a particular area) $x_1 = 0.8$ and $x_2 = 0.6$, respectively, based on the same geochemical results. This case demonstrates a belief conflict between ES_1

and ES_2 because they obtained the same solution with different uncertainties given the same evidences [8]. The final uncertainty $S(x_1, x_2)$ should be between x_1 and x_2, i .e. $\min(x_1, x_2) \leq S(x_1, x_2) \leq \max(x_1, x_2)$ where S is a synthesis strategy [8].

Case (2) ES_1 predicts a class 4 earthquake in an area with uncertainty of $y_1 = 0.8$ based on the evidence from geophysical experiment and ES_2 obtains the same solution with uncertainty of $y_2 = 0.6$ based on the geological structure of this area. The final uncertainty for the solution of a class 4 earthquake in this case should be bigger than any one of y_1 and y_2, i.e. $S(y_1, y_2) \geq max\{y_1, y_2\}$, because two ESs obtain the same solution from different evidences, so this solution is more reliable than if it comes from the same evidence [8].

From the above two cases, we can see that in different synthesis cases, we should use different strategies.

Roughly speaking, synthesis cases can be divided into two parts, conflict cases and non-conflict cases.

Traditional methods of solving belief conflicts in DESs are synthesis strategies which include: uncertainty management strategy developed by Khan [2] in 1985; a synthesis strategy for heterogeneous DESs introduced by Zhang in 1992 [6]; and an improved synthesis strategy which considers both uncertainties and authorities of ESs proposed by Liu in 1992 [3]. In 1994, based on the above research, we also proposed a new comprehensive synthesis strategy to overcome several limitations of above strategies [7].

Also in 1994, we have identified a number of potential synthesis cases [8] in DESs and have proposed two strategies to solve belief conflicts in DESs [7]. Then, we have proposed necessary conditions of synthesis strategies and synthesis strategies for non-conflict cases [10].

These traditional methods of synthesis strategies are based on the mathematical analysis and the proposer of the strategy must know the detail mathematical relationships between the inputs (different solutions from different ESs) and the outputs (final solutions).

Generally, a good synthesis strategy is one which can almost always get the same final solutions as the human experts do in the real world. The synthesis strategies based on mathematical analysis do not work well in some cases since the relationship between the inputs and outputs are too hard to be summarized. The reason is that human experts do not strictly follow the mathematical formula and they sometimes use their experiences to solve synthesis problem.

However, a neural network is a right mechanism (after training) to simulate the relationship between the inputs and the outputs if reasonable numbers of inputs and corresponding outputs are known. Recently, we proposed a dynamic neural network synthesis strategy solving belief conflicts in DESs [9]. To our knowledge, this neural network strategy is the first attempt at using neural network mechanism for solving synthesis problems in DESs. For the cases with enough patterns, neural network strategy is successful. We now extend this idea to non-conflict synthesis cases.

This paper describes two neural network strategies working in two different non-conflict synthesis cases and their performances. The paper is organized as follows. In Section 2, basic problems and different synthesis cases are described. In Section 3, a neural network synthesis strategy for overlap synthesis cases is proposed. In Section 4, a neural network synthesis strategy for disjoint synthesis cases is discussed. Finally, in Section 5, this research is concluded.

2 Basic problems and different synthesis cases

2.1 Problem description

Let's see an example first. Suppose there are three ESs (e.g. ES_1, ES_2, ES_3) to decide the identity of the organism for a specific patient. ES_1 says that it is pseudomonas with uncertainty 0.36 and proteus with uncertainty -0.9, ES_2 says that it is pseudomonas with uncertainty 0.5 and serratia with uncertainty 0.4, and ES_3 says that it is serratia with uncertainty 0.1 and proteus with uncertainty 0.85. Because ES_1 doesn't mention serratia, we believe that ES_1 has no idea about it. We can represent this unknown by using uncertainty 0 in the EMYCIN model [5]. Then the above solutions are represented in Table 1.

	Pseudomonas	Serratia	Proteus
ES_1	0.36	0	-0.9
ES_2	0.5	0.4	0
ES_3	0	0.1	0.85

Table 1: The uncertainties for each attribute value obtained by the ESs.

The purpose of the synthesis of solutions here is how to decide the final uncertainty of pseudomonas, serratia, and proteus according to Table 1.

We now formally describe the problems. Suppose there are n ESs in a DES to evaluate the values of an attribute of an object (e.g. in a medical DES, the identity of an organism infecting a specific patient). The solution for an ES_i can be represented as

$(< object >< attribute > (V_1\ CF_{i1}\ A_i)\ (V_2\ CF_{i2}\ A_i)\ \ldots\ (V_m\ CF_{im}\ A_i))$ 2.1

where V_j $(1 \leq j \leq m)$ represents jth possible value, CF_{ij} $(1 \leq i \leq n, 1 \leq j \leq m)$ represents the uncertainty for jth value from ES_i, A_i represents the authority of ES_i, and m indicates that there are m possible values for this attribute of the object. For example, there are 6 possible values for the face-up of a dice.

From the synthesis point of view, all ESs are concerned with the same attribute of an object. So we will only keep the attribute values, uncertainties, and authorities in the representation. Here is the representation of m possible values with uncertainties from n ESs.

$$\begin{bmatrix} (V_1\ CF_{11}\ A_1)(V_2\ CF_{12}\ \ A_1)...(V_m\ CF_{1m}\ A_1) \\ (V_1\ CF_{21}\ A_2)(V_2\ CF_{22}\ A_2)...(V_m\ CF_{2m}\ A_2) \\ .... \\ (V_1\ CF_{n1}\ A_n)(V_2\ CF_{n2}\ A_n)...(V_m\ CF_{nm}\ A_n) \end{bmatrix} \quad 2.2$$

The synthesis strategy is responsible for obtaining final uncertainties of the vector $(V_1\ CF_{*1}\ A_*)(V_2\ CF_{*2}\ A_*\ )\ ...\ (V_m\ CF_{*m}\ A_*)$ based on matrix 2.2 where * indicates the synthesis result from corresponding values with the subscriptions of $1, 2, ..., n$ in the same place such as $CF_{*m} = S(CF_{1m}, A_1, CF_{2m}, A_2, ..., CF_{nm}, A_m)$ where S is a synthesis function.

Let's see another example. Suppose that there are 4 ESs using the PROBABILITY inexact reasoning model [1] to decide the identity of the organism infecting a specific patient. Each ES obtains six values for the organism. The uncertainties of values obtained by ESs are shown in Table 2.

	$Value_1$	$Value_2$	$Value_3$	$Value_4$	$Value_5$	$Value_6$	Authorities
ES_1	0.05	0.60	0.20	0.05	0.05	0.05	0.95
ES_2	0.01	0.65	0.04	0.03	0.17	0.03	0.85
ES_3	0.03	0.67	0.03	0.05	0.10	0.02	0.70
ES_4	0.06	0.40	0.14	0.19	0.18	0.02	0.80

Table 2: The uncertainties in the PROBABILITY model

The aim of the synthesis strategies here is to decide the final uncertainty of $Value_1$ to $Value_6$ from Table 2.

2.2 Potential synthesis cases in DESs

From the introduction, we know, for the same uncertainties, the synthesis results can be different. That is, when we develop synthesis strategies, the first problem is to identify the different synthesis cases in DESs [8].

There are four potential synthesis cases in DESs based on the relationship between evidence sets of a solution from different ESs [8] which are: (a) conflict synthesis cases, (b) inclusion synthesis cases, (c) overlap synthesis cases, and (d) disjoint synthesis cases. We call *inclusion* synthesis cases, *overlap* synthesis cases, and *disjoint* synthesis cases non-conflict synthesis cases.

Here we briefly describe the definitions of four synthesis cases.

(a) A *conflict synthesis case* [8] occurs when the original evidence sets of a proposition from different ESs are equivalent, but different ESs produce the same solution with different uncertainties. That is, for a proposition B, if there exist $E_{B_i} = E_{B_j}$, where E_{B_i} is the original evidence set to deduce the proposition B in ES_i, E_{B_j} is the original evidence set to deduce the proposition B in ES_j, and $CF_i \neq CF_j$, where CF_i is the uncertainty of the proposition B from ES_i, CF_j is the uncertainty of the proposition B from ES_j.

(b) An *inclusion synthesis case* occurs when the original evidence set of a proposition from one ES is a subset of the original evidence set of another ES.

Formally, for a proposition B, $E_{B_i} \subset E_{B_j}$ or vice versa where E_{B_i} in ES_i and E_{B_j} in ES_j.

(c) A *partial overlap synthesis case* occurs when the original evidence sets of a proposition from different ESs are not equivalent, but the intersection of original evidence sets is not empty. Formally, for a proposition B, $E_{B_i} \cap E_{B_j} \neq \phi$, $E_{B_i} \cap E_{B_j} \neq E_{B_i}$, and $E_{B_i} \cap E_{B_j} \neq E_{B_j}$ where E_{B_i} in ES_i and E_{B_j} in ES_j.

(d) A *disjoint synthesis case* occurs when the intersection of original evidence sets of a proposition from different ESs is empty. Formally, for a proposition B, $E_{B_i} \cap E_{B_j} = \phi$, where E_{B_i} in ES_i, and E_{B_j} in ES_j.

So far, most papers have been concerned with a conflict synthesis case. However, solving non-conflict cases is as important as solving a conflict case.

In this paper, we will introduce two neural network strategies working in overlap and disjoint synthesis cases, respectively.

3 A neural network synthesis strategy for overlap synthesis cases

In this section, a neural network synthesis strategy to solve synthesis problems in overlap synthesis cases will be introduced.

3.1 Basic principles

Generally speaking, there are two methods to build neural networks: to train neural networks by learning patterns, or to train neural networks by learning rules. Since it is difficult to find rules for synthesis of solutions in DESs (recall Section 1), we use learning patterns to create our neural network.

The ideas are: the inputs of the neural network are the matrix of uncertainty values of the same attribute of an object from different ESs (recall matrix 2.2 or Table 2); the outputs of neural network should be a vector of uncertainty values of the attribute of the object which are the final uncertainty values of the DESs after synthesizing input uncertainties. The closer the vector of uncertainty values to human experts' final uncertainty values, the better. We would like to investigate whether a neural network can be trained to solve synthesis problems in overlap synthesis cases if enough patterns are given. We should collect enough number of patterns (input matrix and corresponding output vector) from human experts; to set up a neural network architecture (input layer nodes, hidden layer nodes, and output layer nodes), to decide an activation function, and to train this neural network by adjusting the weights of links to accommodate all patterns.

The following activation function is used.

$$a_j(t+1) = \frac{1}{1 + e^{(\sum_i w_{ij} o_i(t) - \theta_j)}}$$

where $a_j(t)$ is the activation of unit j in step t, $o_i(t)$ is the output of unit i in step t, w_{ij} is the weight of link from unit i to unit j, and θ_j is the threshold of unit j.

One of the important research issues in neural networks is how to adjust the weight of links to get the desired system behavior [4].

In our strategy, we use a backpropagation learning algorithm to train the network since it is most commonly used for classification problem solving.

The backpropagation weight update rule is described as follows:

$$\Delta w_{ij} = \eta \delta_j o_i$$

$$\delta_j = \begin{cases} f'_j(net_j)(t_j - o_j) & \text{if unit } j \text{ is a output-unit} \\ f'_j(net_j)\sum_k \delta_k w_{jk} & \text{if unit } j \text{ is a hidden-unit} \end{cases}$$

where η is a learning factor, δ_j is the error of unit j, t_j is the teaching input of unit j, o_i is the output of preceding unit j, i is the index of a predecessor to the current unit j with link w_{ij} from i to j, j is an index of the current unit and k is an index of a success or to the current unit j with link w_{jk} from j to k.

3.2 A dynamic neural network mechanism

A neural network should have $n * m$ input nodes and m output nodes (recall matrix 2.2 for the meaning of n and m) in order to solve our problem. For a DES, the number of ESs (n) is relatively stable. However, the possible number of values (m) for each attribute varies from case to case.

It is not practical to train a different neural network every time when the number of values is changed. In this subsection we will propose a method to build a fixed neural network architecture for synthesis of solutions in overlap synthesis cases to adapt the problem with variable numbers of inputs and outputs.

Suppose k is the maximum number of values of all attributes in a DES. The idea is to set up a neural network with $n * k$ input nodes and k output nodes. The training patterns can be $n * 2$ inputs with 2 outputs, $n * 3$ inputs with 3 outputs, ..., and $n * k$ inputs with k outputs. If the number of values for some attributes is less than k, the default values "0" are chosen to substitute these absent values as inputs and corresponding outputs. If the neural network after training can accommodate all (enough number of) patterns, we believe that this neural network can work well for untested cases. We have built a neural network to test the above ideas. We choose $n = 4$, $k = 6$. The input layer has $4 * 6$ nodes and output layer has 6 nodes.

There are three layers in this network. The input layer is a 4*6 matrix, the hidden layer is a 2*10 matrix, and the output layer is a 1*6 vector. The neural network is full connection. The input of the neural network is the matrix of uncertainty values of the same attribute of an object from 4 ESs (recall matrix 2.2); and the output of the neural network is a vector of uncertainty values of attribute of the object which are the final uncertainty values of DESs after synthesizing input uncertainties in a overlap synthesis case.

The architecture of the neural network is shown in Figure 1.

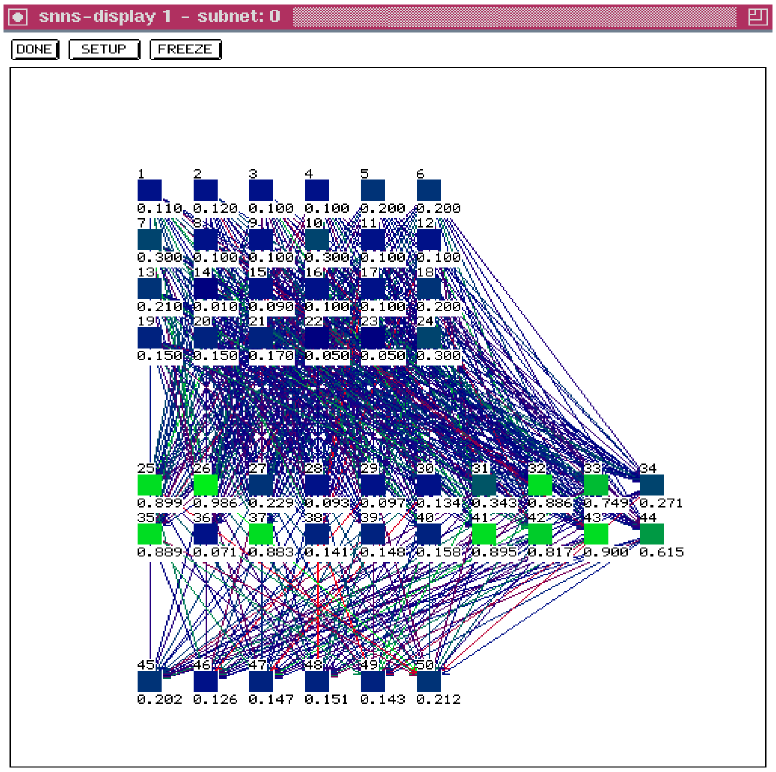

Figure 1: The architecture of a neural network for overlap synthesis cases

Table 3 and Table 4 are examples of training patterns with different default values.

3.3 Training and testing results

So far we haven't collected real data from human experts. What we would like to investigate in this paper is that whether a neural network can converge for a set of complicated artificial data. If it succeeds, it would be a positive signal to encourage us to collect real data from human experts to train neural networks.

In this paper, there are 102 different patterns created by the mathematical synthesis strategy for overlap synthesis cases [10] considering different possibilities. 92 of them are used as training patterns and other 10 untrained patterns are used as test patterns.

The training results are shown in Table 5.

The error value δ^0_{pq} for the qth output neuron acting of input vector (or matrix) p is calculated as follows:

$$\delta^0_{pq} = (y_{pq} - o_{pq})$$

Input	$Value_1$	$Value_2$	$Value_3$	$Value_4$	$Value_5$	$Value_6$
ES_1	0.11	0.12	0.10	0.10	0.20	0.20
ES_2	0.30	0.10	0.10	0.30	0.10	0.10
ES_3	0.21	0.01	0.09	0.10	0.10	0.20
ES_4	0.15	0.15	0.17	0.05	0.05	0.03
Output	0.162	0.162	0.184	0.133	0.133	0.227

Table 3: An example of training patterns without default values

Input	$Value_1$	$Value_2$	$Value_3$	$Value_4$	$Value_5$	$Value_6$
ES_1	0	0	0.20	0.20	0.30	0.30
ES_2	0	0	0.30	0.30	0.20	0.20
ES_3	0	0	0.05	0.16	0.09	0.66
ES_4	0	0	0.18	0.28	0.25	0.25
Output	0	0	0.151	0.134	0.161	0.554

Table 4: An example of training patterns with two default values

Training Cycles	SSE	SSE/o-units
10,000	0.12175	0.02029
20,500	0.02695	0.00449
64,000	0.01617	0.00234
100,000	0.00907	0.00151

Table 5: Training results

where y_{pq} is the desired output of the qth unit of vector (or matrix) p and o_{pq} is the final output of the qth unit of vector (or matrix) p. The total error of the network E_p^0 is described by the summation of the error values for each output unit k:

$$E_p^0 = \frac{1}{2}\sum_{k=1}^{m}(\delta_{pq}^0)^2 = \frac{1}{2}\sum_{k=1}^{m}(y_{pq} - o_{pq})^2$$

where m is the number of nodes in output layer.

In Table 5, SSE is the sum-squared error of the learning function for all nodes and SSE/o-units is the result of SSE divided by the number of output units. The training result is good. When we select 10 patterns from 92 training patterns to test this network, the mean error is less than 1.5%. For 10 untrained patterns, the mean error is on 2.8% level.

The percentage of the error from test patterns in each stage and both maximum and mean errors are shown in Table 6 and Table 7.

range of error	percentage
0 – 0.01	15/60
0.01 – 0.02	12/60
0.02 – 0.03	8/60
0.03 – 0.04	9/60
0.04 – 0.05	5/60
0.05 – 0.06	5/60
0.06 – 0.07	2/60
0.07 – 0.08	2/60
0.08 – 0.09	1/60
0.09 – 0.11	1/60

Table 6: The percentage of error in each stage

Maximum error	0.111
Mean error	0.028
Number of data	60

Table 7: Information of final errors

Figure 2 shows the distribution of errors from test patterns.

Table 6 shows that 82% ($15/60 + 12/60 + 8/60 + 9/60 + 5/60 = 49/60 = 0.82$) of errors from output data are below 0.05, 93.3% ($15/60 + 12/60 + 8/60 + 9/60 + 5/60 + 5/60 + 2/60 = 0.933$) of errors from output data are below 0.07, and only 6.7% of errors from output data are between 0.07 to 0.11. This result can verify that the neural network mechanism can also use in overlap synthesis cases.

Table 8 shows the synthesis result from both mathematical synthesis strategy [10] and this neural network strategy.

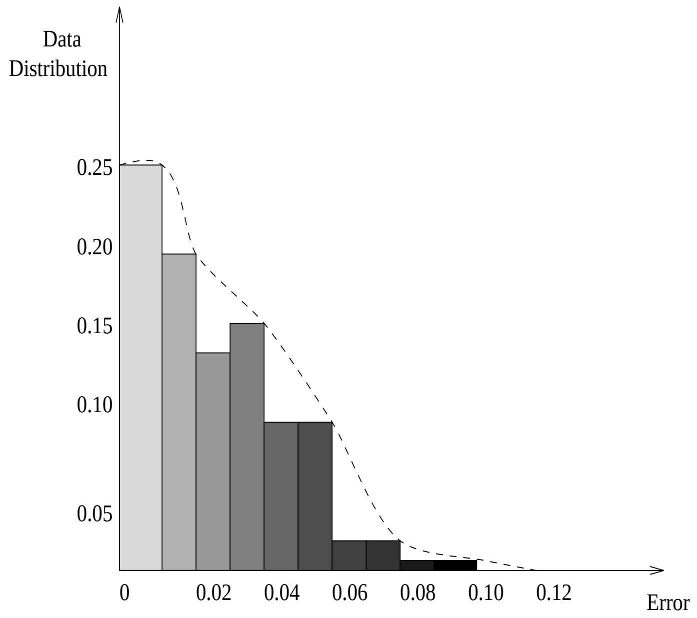

Figure 2: The distribution of the test errors

Input (Uncertainties from ESs)	V_1	V_2	V_3	V_4	V_5	V_6
ES_1	0.05	0.60	0.20	0.05	0.05	0.05
ES_2	0.01	0.65	0.04	0.03	0.17	0.03
ES_3	0.03	0.67	0.03	0.05	0.10	0.02
ES_4	0.06	0.40	0.14	0.19	0.18	0.02
Final result (Mathematical strategy)	0.035	0.555	0.122	0.098	0.139	0.051
Output (Neural network strategy)	0.040	0.600	0.099	0.089	0.130	0.060

Table 8: The comparison of the final results between two strategies

Now let us compare the output of neural network strategy with the final result of the mathematical synthesis strategy in overlap synthesis cases (see Table 8). The mean error of the neural network strategy is 0.014 and maximum error is 0.045. That means the synthesis result from neural network can be trusted.

4 A neural network synthesis strategy for disjoint synthesis cases

In this section, we will proposed a neural network synthesis strategy for disjoint synthesis cases.

4.1 Architecture of a neural network

We also use the idea of dynamic mechanism to develop this strategy (recall Section 3.2).

There are also three layers in the neural network which are input layer, hidden layer, and output layer. Both the input layer and the output layer are the same sizes as the neural network for overlap synthesis cases. The hidden layer is a 7*7 matrix. Please see Subsection 3.1 for the meaning of the input and the output of a neural network synthesis strategy. The neural network is also fully connected.

Figure 3 is the architecture of the neural network.

The learning algorithm is also backpropagation (recall Section 3.1). There are 95 patterns created by the mathematical synthesis strategy for disjoint synthesis cases [10] to be used as training patterns. There are 7 untrained patterns used as test patterns.

4.2 Result analyses

The neural network was trained by 53,000 cycles. Table 9 is the training result.

Training Cycles	SSE	SSE/o-units
10,000	0.11413	0.01902
20,000	0.02629	0.00438
53,000	0.01040	0.00173

Table 9: Training results

Please see Subsection 3.3 for the meaning of SSE and SSE/o-units.

When we randomly selected 10 patterns from training patterns to test the network, the mean error is around 1%. For 7 untrained patterns, the percentage of the error from testing results in each stage is shown in Table 10. Both maximum and mean errors are shown in Table 11.

Figure 4 figures out the distribution of test errors from test patterns.

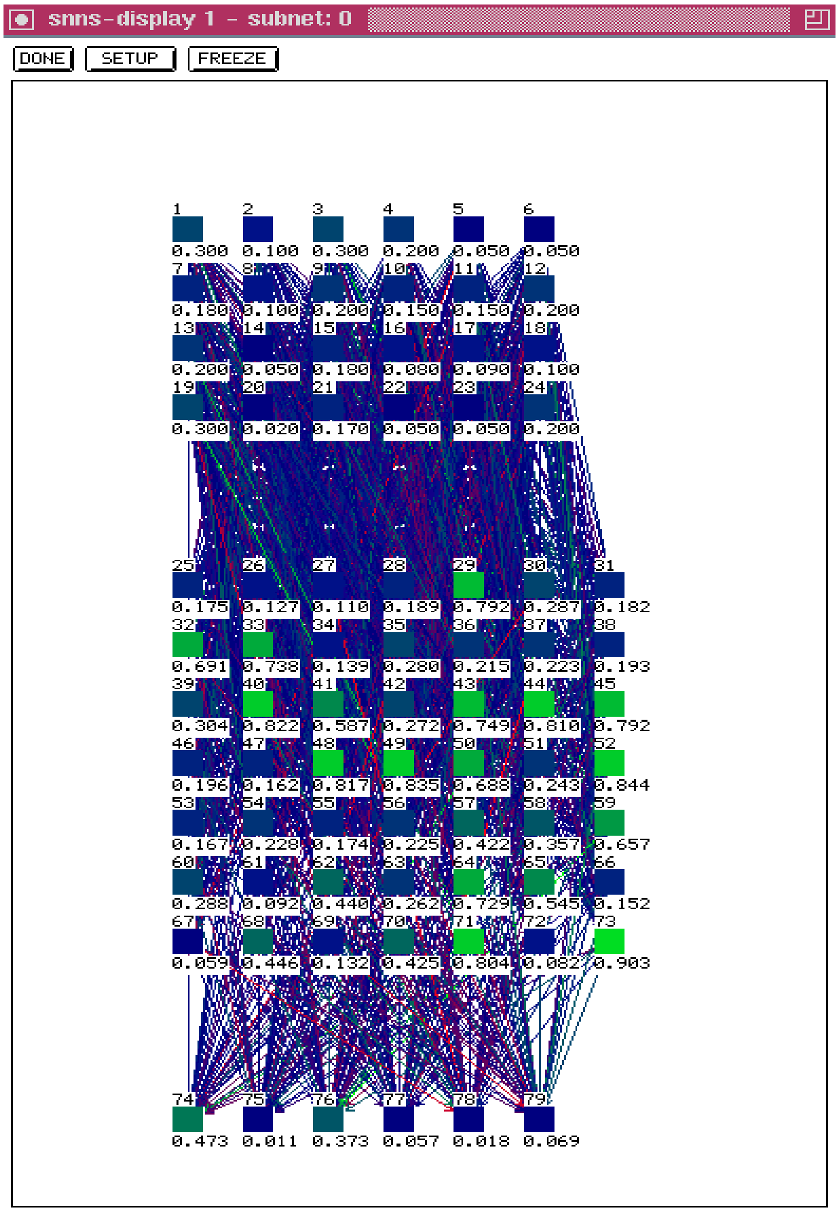

Figure 3: The architecture of a neural network for disjoint synthesis cases

range of error	percentage
0 – 0.01	11/42
0.01 – 0.02	7/42
0.02 – 0.03	4/42
0.03 – 0.04	3/42
0.04 – 0.05	3/42
0.05 – 0.06	4/42
0.06 – 0.07	2/42
0.07 – 0.08	1/42
0.08 – 0.09	1/42
0.09 – 0.11	1/42

Table 10: The percentage of error in each stage

Maximum error	0.107
Mean error	0.036
Number of data	42

Table 11: Information of final errors

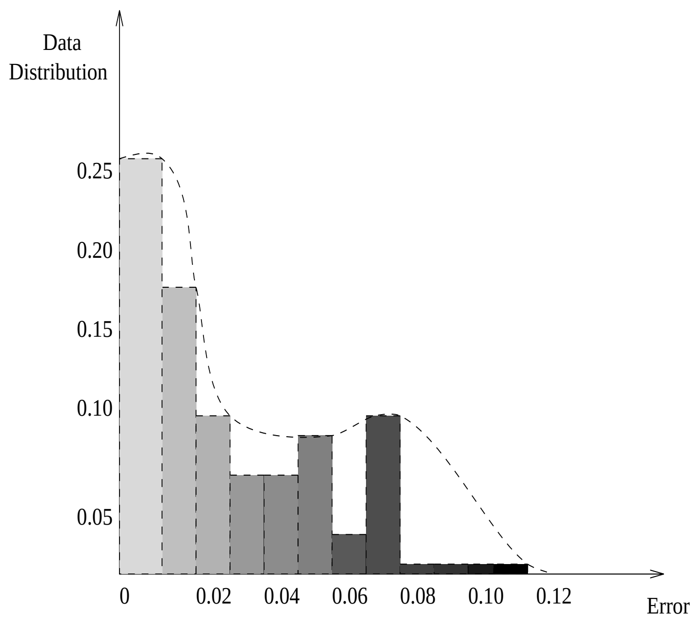

Figure 4: The distribution of the test errors

From Table 11, we can see that the mean error for test patterns is on 3.6% level. From Table 10, we can find that 67% of errors are below 0.05, 92.9% of errors are below 0.08, and 7% of errors are from 0.08–0.107.

We use the same example as in Section 3.3 to synthesize a disjoint synthesis case by this neural network strategy. Table 12 shows the synthesis result by using this neural network strategy.

Input	$Value_1$	$Value_2$	$Value_3$	$Value_4$	$Value_5$	$Value_6$
ES_1	0.05	0.60	0.20	0.05	0.05	0.05
ES_2	0.01	0.65	0.04	0.03	0.17	0.03
ES_3	0.03	0.67	0.03	0.05	0.10	0.02
ES_4	0.06	0.40	0.14	0.19	0.18	0.02
Output	0.001	0.941	0.033	0.006	0.002	0.000

Table 12: Synthesis result by the neural network strategy

Table 13 is results of final uncertainties between the mathematical synthesis strategy and the neural network strategy for the same disjoint synthesis case.

Strategies	$Value_1$	$Value_2$	$Value_3$	$Value_4$	$Value_5$	$Value_6$
Mathematical model	0.002	0.913	0.026	0.010	0.047	0.002
Neural network	0.001	0.941	0.033	0.006	0.002	0.000

Table 13: The comparison of the final results between mathematical and neural network strategies

The above table shows that the maximum error from neural network strategy is 4.5% and the mean error from neural network strategy is only 1.4%. This result indicates that a neural network synthesis strategy is a good way to solve synthesis problems if enough training patterns are available.

Two neural network strategies introduced in this paper were implemented by the Stuttgart Neural Network Simulator [4].

5 Conclusion

In this paper, we introduced two neural network mechanisms to solve synthesis problems in overlap and disjoint synthesis cases and tested that a fixed neural network architecture can solve the problems with a variable number of inputs and outputs of neural networks.

This is the first time a neural network mechanism has been used to solve synthesis problems in non-conflict cases in DESs. The contributions are as follows.

(1) The neural network is easy to build because we need to know enough patterns of inputs and outputs only without knowing the relationship between them.

(2) The results of synthesis of solutions in non-conflict cases by using neural network mechanism are good so far because trained neural network can simulate patterns very well.

(3) A fixed neural network architecture can be used to solve the problem with a variable number of inputs and outputs. This result makes it practical to use the neural network mechanism to solve synthesis problems in DESs, even though the number of inputs and outputs of the neural network is changeable. This result can also be used in other application fields of neural networks.

(4) If the neural network converges and there are enough patterns available, the neural network strategy is better than mathematical formula because it can simulate human expert quite closely.

The further work will include two directions: (1) when the data in the real world are collected and used to train neural networks, we will see whether the performance is still good enough; (2) when the number of patterns increase extremely, we will see whether neural networks still converge in reasonable time.

References

[1] R. Duda, P. Hart and N. Nilsson (1976). Subjective Bayesian Method for Rule-based Inference System, *AFIPS*, **45**, pp. 1075-1082.

[2] N. A. Khan and R. Jain (1985). Uncertainty Management in a Distributed Knowledge Based System. In *Proceedings of 5th International Joint Conference on Artificial Intelligence*, Los Angeles, pp. 318-320, 1985.

[3] D. Liu, F. Zheng, Z. Ma and Q. Shi (1992). Conflict Resolution in Multi-ES cooperation Systems. *Journal of Computer Science and Technology*, Vol. 7, No. 2, pp. 144-152, 1992.

[4] N. Mache, R. Hubner, G. Mamier, and M. Vogt (1993). *SNNS,* Report No. 3/93, Institute for Parallel and distributed High Performance Systems, Germany, 1993.

[5] W. Van Melle (1980). A Domain-Independent System That Aids in Constructing Knowledge-Based Consultation Programs, *Ph.D. Dissertation, Report STAN-CS-80-820,* Computer Science Department, Stanford University, CA.

[6] C. Zhang (1992). Cooperation under Uncertainty in Distributed Expert Systems, *Artificial Intelligence*, **56**, pp.21-69.

[7] M. Zhang and C. Zhang (1994). A Comprehensive Strategy for Conflict Resolution in Distributed Expert Systems, *Australian Journal of Intelligent Information Processing Systems*, **Vol. 1**, No. 2, pp. 21-28.

[8] M. Zhang and C. Zhang (1994). Synthesis of Solutions in Distributed Expert Systems. In *Artificial Intelligence - Sowing the Seeds for the Future*, edited by Chengqi Zhang, Jone Debeham and Dickson Lukose, *World Scientific Publishers, Singapore*, pp. 362-369.

[9] M. Zhang and C. Zhang (1995). A Dynamic Neural Network Mechanism for Solving Belief Conflict in Distributed Expert Systems. In *Proceedings of Sixth Australian Conference on Neural Networks*, Sydney, Australia, pp. 105-108, 1995.

[10] M. Zhang and C. Zhang (1995), The Synthesis Strategies for Non-conflict Cases in Distributed Expert Systems. *AI'95*, World Scientific Publishers Co., November, 1995.