# Potential Ways for Developing Synthesis Strategies in Distributed Expert Systems

Minjie Zhang
School of Computer, Information, & Mathematical Sciences
Edith Cowan University
W.A. 6027, Australia

**Abstract**

This paper introduces three methods for solution synthesis in distributed expert systems (DESs). This paper also demonstrates different strategies which are a neural network strategy, and a case-based strategy. Furthermore, these strategies are compared based on the design principles and performances. This research is to study different methods for designing synthesis strategies and offers guidelines for selecting a suitable method for solution synthesis in different synthesis cases.

## 1 Introduction

Generally, most distributed expert systems (DESs) concentrate on producing a cooperative solution to a problem by using both artificial intelligence and distributed problem solving techniques [1, 4]. In a distributed expert system (DES), a task may be allocated to more than one expert system (ES) to solve in order to increase the reliability of the solution [1, 3, 8]. In such a case, each ES will obtain an individual solution. These solutions from different expert systems (ESs) could be same, or different. This paper introduces potential ways for developing synthesis strategies in DESs in attempting to synthesize multiple solutions from different ESs to produce the final solution to the task (or subtask).

The remainder of this paper is organized as follows. Section 2 is the domain problem description. In Section 3, three possible approaches for designing synthesis strategies are discussed, and the basic principles of a computational strategy, a neural network strategy, and a case-based strategy have been introduced. In Section 4, different approaches are compared and the guide line of selecting different approaches is discussed. Finally, in Section 5, this paper is concluded and the further work is outlined.

## 2 Domain problem description

A formal description of synthesis of solutions is as follows: Suppose task $T$ is allocated to $ES_i(1 \leq i \leq n)$, the solutions to task $T$ from $ES_i(1 \leq i \leq n)$ are represented as $R_i(T)(1 \leq i \leq n)$. Then

$$R(T) = SS(R_1(T), R_2(T), ..., R_n(T))$$

where $R(T)$ represents the final solution to task $T$, and $SS$ is one of the synthesis strategies. In this paper, $T$ is used to represent either a task or a subtask without distinction. Task $T$ can be generalized as the evaluation of the values of an attribute of an object, (e.g. "What is the identity of the organism in a specific patient?"), a solution to task $T$ from $ES_i$ can be represented as:

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

where $V_j$ represents the $jth$ possible value, $CF_{ij}$ represents the uncertainty for the $jth$ value from $ES_i$, $A_i$ represents the authority (or confidence) for $ES_i$, and $m$ indicates that there are exactly $m$ possible values for this attribute of the object. From the synthesis point of view, all ESs are concerned with the same attribute of an object. So we omit the attribute of an object in the representation. Each vector represents a solution from one ES. All solutions from $n$ ESs are represented as a matrix in Table 1:

| | $V_1$ | $V_2$ | ... | $V_m$ | Authority |
|---|---|---|---|---|---|
| $ES_1$ | $CF_{11}$ | $CF_{12}$ | ... | $CF_{1m}$ | $A_1$ |
| $ES_2$ | $CF_{21}$ | $CF_{22}$ | ... | $CF_{2m}$ | $A_2$ |
| ... | ... | ... | ... | ... | ... |
| $ES_n$ | $CF_{n1}$ | $CF_{n2}$ | ... | $CF_{nm}$ | $A_n$ |

Table 1: Formal descriptions of the synthesis problems

Synthesis strategy ($SS$) is responsible for obtaining final uncertainty distribution among attribute values which are ($CF_{*1}$ $CF_{*2}$ ... $CF_{*m}$ ) based on Table 1 where * indicates the synthesis result from corresponding values with subscriptions of 1, 2, .., $n$ in the same place.

# 3 Three possible methods

In this section, we discuss the three methods for finding functions to match *inputs* (multiple solutions) and *outputs* (final solution) in DESs based on conditions of synthesis situations.

## 3.1 Analytic methods

**Principle of analytic methods**

Analytic methods are used to define synthesis functions (called strategies) from input $X_i$ to output $Y_i$ (refer to Definition 1) through analysis of the characteristics of $X_i$. These characteristics may include relationships among original evidence sets [8] from ESs which derive the input matrix, the factors which affect the final solution, and the weights for all factors. These methods are useful for areas in which the individual solution from each expert can be described by uncertainties, or numbers, and relationships between $X_i$ and $Y_i$ are not so complicated. In particular, these methods are suitable for areas in which patterns are difficult to obtain. Popular mathematical theories used to implement analytic methods are evidential theory, probability theory [2], decision theory, and so on. Generally, there are five levels to be explored in this kind of methods:

- analysis of the characteristics of the evidence to derive an input $X_i$ (such as conflict cases and non-conflict cases [8]);
- analysis of input $X_i$ itself (such as all experts give consistently positive solutions, or negative solutions, or some experts give positive solutions while other experts give negative solutions);
- analysis of the factors which affect $Y_i$ ( such as, average of individual solutions, the consistency among individual solutions, confidence of each expert, and so on);
- how to weight each factor; and
- how to combine different factors.

We have developed a comprehensive strategy to solve synthesis problems by analytic methods [6].

## 3.2 Inductive methods

**Principle of inductive methods**

The basic idea of inductive method is to find the general relationship between $X_i$ and $Y_i$ based on sufficient patterns. The principle of inductive methods is from specific to general. Suppose $\forall i(1 \leq i \leq N) f(X_i) = Y_i$, in which $f$ is a synthesis strategy, $X_i$ is the input matrix of multiple solutions, and $Y_i$ is the known synthesis solution from $X_i$, we believe that $f$ also works well for any $X_j (j > N)$ although we cannot guarantee that it is always true.

**Example 1: Structures of neural network approaches**

For some synthesis problems, if the relationship between *inputs* and *outputs* is linear, we can use mathematical models to find this relationship such as "The Least Square Method". However, it doesn't work well for complicated problems. Another tool to implement inductive methods is neural network.

The characteristics of a neural network strategy are based on patterns of both inputs and corresponding outputs to find the best mapping functions from inputs to corresponding outputs.

We have developed several neural network strategies working in different synthesis cases [7, 8]. Figure 1 is a basic structure of a neural network strategy.

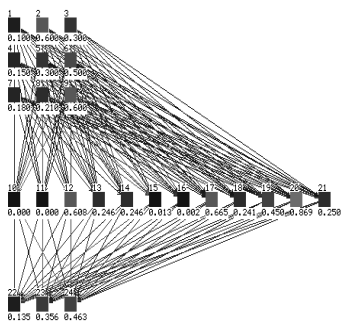

Figure 1: The architecture of a neural network strategy

The design principles of a neural network strategy are: the inputs of the neural networks are the matrix of multiple solutions from ESs (refer to Table 1 ); the outputs of neural networks should be the final solutions after synthesizing multiple solutions. If a neural network can converge for all of patterns after training, this neural network can act as an inductive function.

Based on our experience [7], a neural network is a

good mechanism to implement an inductive method. In the strategy of solving conflict cases, the neural network converges well for 180 artificial patterns, ($Minimun_{error} < 0.00038$).

### 3.3 Analogical methods

**Principle of analogical methods**

The principle of analogical methods is to find $Y_j$ from $X_j$ by comparing $X_j$ to similar known patterns $X_i$ where $X_i$ and $X_j$ represent two input matrices for problem $i$, and $j$, and $Y_i$, and $Y_j$ represent the final solution after synthesis from $X_i$, and $X_j$, respectively. Then, $Y_j$ should be similar to $Y_i$. This method is useful in the following situations. (a) It is hard to find many patterns to enable use of inductive methods; (b) A few patterns do exist for similar cases; and (c) It is difficult to analyze the relationship between inputs and outputs by mathematical formulas.

Case-based reasoning [5] is a good tool to implement analogical methods because the principle of case-based reasoning is to find similarity between cases.

**Example 2: A basic framework of a case-based approach**

Figure 2 is our synthesis strategy developed by a case-based approach in 1997.

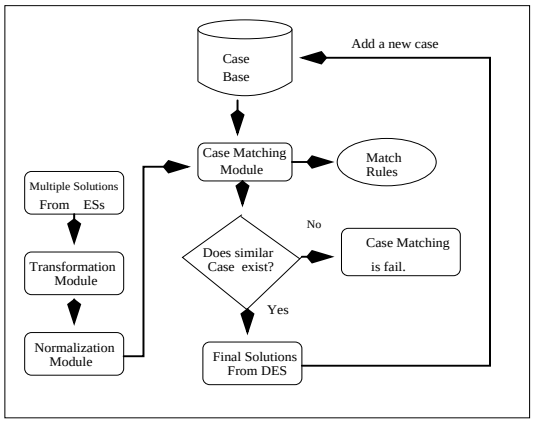


Figure 2: The principle of the case based approach

The basic idea of the case-based strategy is to solve a new problem (an input case) by reusing solutions that were used to solve old problems (existing cases). In this research, we explore how to solve synthesis problems by a case-based approach. It involves (1) how should an input case be generated from multiple solutions from ESs if these ESs use different inexact reasoning models; and (2) how is the relevant case selected? In current stage, we do not use adaptation and we directly apply whatever solution is retrieved from past solutions to the new situation. The reason for without doing adaptation comes from special features of synthesis of solutions in DESs. Results of solutions themselves are very simple, but reasonings leading to solutions are very complex and involve many factors such as the sizes of knowledge bases in ESs, methods used for a synthesis, and the weights used to measure each factor. If a suitable matching case does not exist, the system will offer some roughly matching cases to users. In this situation, a user can make his/her choices or choose other strategies.

The basic design principles of this strategy are: (a) For an input case, the system will search the case base to find the best matching case. If a matching case exists, the final solution part in the matching case is used as the solution for the input case. Both the current case and the final solution might be added into the case base. (b) If there is no similar case available in the case base, the matching procedure is not successful.

In order to increase the system efficiency, the searching space is classified into 6 levels according to ranks of $\overline{error}$. We use the nearest neighborhood approach to reduce the searching area. If a user doesn't choose a starting level, searching normally starts from level 1. If a matching case is not found in level 1, the searching area will be extended to level 2, and so on until a similar case is found or whole searching space has been examined.

The working procedure of this strategy is briefly described as follows.

**(a) Transformation module** — If the range of uncertainties of a proposition is not in the range [0,1], the uncertainties of the proposition are transformed from that range to the range of [0, 1] by using the heterogeneous transformation functions in the transformation module. For example, if an ES uses the range of $[-1, 1]$, it should be transformed into the range of [0, 1] in the PROBABILITY model [2].

**(b) Normalization module** — After transformation, the sum of uncertainties may not satisfy the requirements in the PROBABILITY model [2]. If such a case happens, the normalization function is used to normalize uncertainties in the PROBABILITY model.

**(c) Case matching module** — Multiple solutions from ESs are as an input case to the case matching module. In this module, the system examines the case base and searches for the best matching case. Once a suitable case is found, its corresponding solution might be used as a refereed solution for the input case. The refereed solution will be considered as a new case

which may be added to the case base.

**(d) Case base** — The case base is a main resource used in this strategy. Currently, there are 180 cases stored in the case base.

**(e) Matching rule sets** — There are six match functions in this set which are implemented case matching rules in different levels.

## 4 Selection guidelines

- Analytic methods can work well for some simple problems. Especially this method is perfect for the case if the desired final solutions are derived from some formulas based on inputs.
- Inductive methods can be used to solve complicated problems. So far neural network mechanism is a good way to implement inductive function. For complicated cases, a neural network approach is better than an computational approach since a neural network can simulate complicated relationships quite closely if it converges.
- If both analytic methods and inductive methods cannot work for some situations, analogical methods can be considered as an alternative to solve the problem such as using case-based reasoning to find similar cases for the mapping functions.

After comparison, our conclusion is that they compensate each other. How to select a suitable method is mainly based on different kinds of synthesis problems and available information.

## 5 Conclusions

In this paper, the problem of solution synthesis in DESs has been described on both the general level and the abstract level, and three potential methods for designing synthesis strategies have been proposed and analyzed, which are analytic methods, inductive methods, and analogical methods. Furthermore, the basic principle of a computational strategy, a neural network strategy, and a case-based strategy, which are developed by different approaches, have been introduced, and the requirements and performances of different strategies have been discussed. The aim of this research is to study the potential approaches for solution synthesis in DESs, and offers the guideline for selecting a suitable method for developing synthesis strategies based on a particular problem and available information.

The further direction of this research is to investigate the methodologies of solution synthesis, to improve neural network strategies and the case-based strategy, and to apply these strategies in real application domains.

## References


[1] A. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Mateo, California, 1998.

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

[3] N. Findler and R. Lo, "An Examination of Distributed Planning in the World of Air Traffic Control", *Parallel Distributed Computer*, **Vol. 3**, pp. 411-431, 1986.

[4] V. Lesser and D. Corkill, "The Distributed Vehicle Monitoring Testbed", *AI Magazine,* **Vol. 4,** pp. 63-109, 1983.

[5] K. Sycara, "Planning for Negotiation: A Case-Based Approach", *Proceedings of Darpa Knowledge-Based Planning Workshop*, pp. 11.1-11.10, 1987.

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

[7] M. Zhang and C. Zhang, "Neural Network Strategies for Solving Synthesis Problems in Non-conflict Cases in Distributed Expert Systems", *Distributed Artificial Intelligence: Architecture and Modeling*, LNAI **Vol 1087**, Lecture Notes in Artificial Intelligence, Springer Verlag Publishers, pp. 174-188, 1996.

[8] M. Zhang and C. Zhang, "Analysis and Methodologies of Synthesis of Solutions in Distributed Expert Systems", *Proceedings of the Second International Conference on Multi-Agent Systems*, Kyoto, Japan, pp. 417-424, 1996.