A Case-based Strategy for Solution Synthesis among Cooperative Expert Systems

Minjie Zhang
School of Computer, Information, and Mathematical Sciences
Edith Cowan University, W.A. 6027, Australia
m.zhang@cowan.edu.au

Abstract. Most synthesis strategies for solution synthesis in distributed expert systems (DESs) are developed based on analytic methods or inductive methods. A synthesis strategy presented in this paper is developed by case-based reasoning based on an analogical method. In this paper, the basic frame work of the strategy is introduced and preliminary results are demonstrated. Furthermore, this strategy is compared with related work.

1 Introduction

Synthesis of solutions has been become one of critical issues among cooperative expert systems for more than ten years [8]. Some synthesis strategies have been developed in distributed expert systems (DESs) such as [2, 7, 9]. These strategies were mainly based on the mathematical analysis of the characteristics of the *inputs* (multiple solutions from ESs). In recent years, we have developed some synthesis strategies [10] using neural networks. These strategies were based on a number of samples with both *inputs* and corresponding *outputs* (final solutions after synthesizing corresponding *input*). Such samples are called *patterns*.

In current literature, there are two methodologies used for designing synthesis strategies to solve synthesis problems. They are analytic methods, and inductive methods [11]. An analytic method is a methodology which can be used to define a synthesis strategy from *inputs* to *outputs* by analyzing the characteristics of *input*. These characteristics may include relationships among original evidence sets from ESs which derive the *inputs*, the factors which affect the final solution, and the weights for all factors. An inductive method is a methodology which can be used to find the general relationship between *input* and *output* based on sufficient patterns.

Different kinds of strategies are good in different kinds of problems. Analytic methods can work well for some simple problems if the relationship between *inputs* and *outputs* can be summarized by an analytic method. In particular, this method is perfect for use with cases where the *outputs* can be derived from some formulas based on *inputs*. Inductive methods can solve some complicated synthesis problems when the enough patterns exist and an inductive function can be found.

There are two major limitations in the above methodologies: (1) if the synthesis problem can not be summarized by a mathematical model, it is very hard to solve the problem by an analytic method; and (2) if an inductive function does not exist for the problem solving, or there are no enough learning patterns available, it is impossible to use inductive methods such as neural network strategies.

In this paper, we introduce a new synthesis strategy, case-based reasoning, which is an alternative way for overcoming the above limitations, by using an analogical method. Analogical methods are different from analytic methods and inductive methods. Analytic methods are based on only analysis without taking real examples into consideration. Inductive methods need too many real examples which are not available in most application fields. Case-based reasoning is an instance of analogical methods which is a popular method used in the learning science even in human society [5]. Case-based reasoning is an approach for problem-solving based on retrieving and applying stored solutions. In case-based reasoning, only some real examples are needed.

This paper is organized as follows: In Section 2, general procedure of solution synthesis in DESs are described. In Section 3, the system architecture of a synthesis strategy by case-based reasoning is proposed and described in detail. In Section 4, case matching rules and several examples are introduced. In Section 5, this strategy is compared with related work. Finally, in Section 6, this paper is concluded and the future direction of this research is outlined.

2 Description of solution synthesis

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 0.06 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 0.80. 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. Then the above solutions are represented in Table 1.

	Pseudomonas	Serratia	Proteus	Authority
ES_1	0.06	0	-0.9	0.6
ES_2	0.5	0.4	0	0.8
ES_3	0	0.1	0.80	0.9

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

The purpose of the solution synthesis here is to decide the final uncertainty distribution among 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

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

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. The authority A_i $(1 \leq i \leq n)$ is the confidence level for the solution from ES_i. It can be assigned for each ES by human experts or generated based on the historical performance of ESs.

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:

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

The synthesis strategy is responsible for obtaining final uncertainties (CF_{*1} CF_{*2} ... CF_{*m}) based on Matrix 2.2 where * indicates the synthesis result from corresponding values with the subscriptions of $1, 2, ..., n$ in the same place.

3 A basic framework of a case-based strategy

3.1 Definitions

Throughout this paper, we make the following definitions:

Definition 1: *An input* of a synthesis strategy is the matrix representation of multiple solutions from different ESs (recall Matrix 2.2).

Definition 2: *An output* of a synthesis strategy is the final uncertainties of the vector (CF_{*1} CF_{*2} ... CF_{*m}) after synthesis of the input matrix.

Definition 3: *An input case* is a set of multiple solutions from different ESs for the same problem. It can be represented by an input matrix.

Definition 4: *An existing case* consists of an input matrix and an output vector.

A general case representation is demonstrated in Figure 1.

Examples of both an input case and an existing case in the PROBABILITY model are shown in Table 2 and Table 3.

	ES_s	$Value_1$	$Value_2$	$Value_3$
multiple solutions	ES_1	0.04	0.14	0.40
	ES_2	0.3	0.01	0.50
	ES_3	0.11	0.83	0.05

Table 2. A example of an input case in the PROBABILITY model

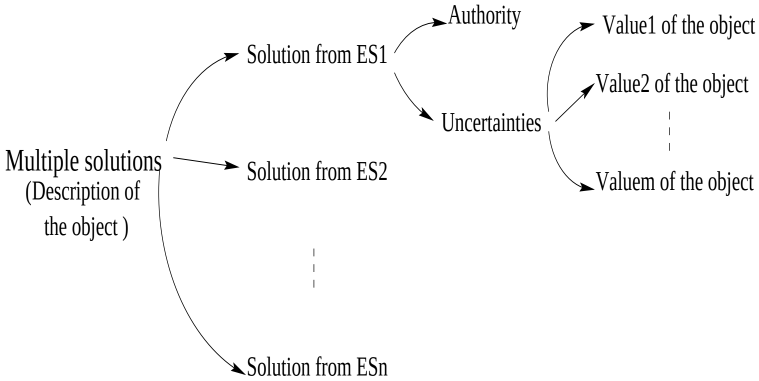

Fig. 1. Case representation

	ES_s	$Value_1$	$Value_2$	$Value_3$
multiple solutions	ES_1	0.07	0.19	0.40
	ES_2	0.15	0.10	0.30
	ES_3	0.01	0.03	0.80
final solution	DES	0.08	0.11	0.48

Table 3. A example of an existing case in the PROBABILITY model

3.2 System architecture

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) [5]. In this preliminary 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 choices or choose other strategies.

The basic principle 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.

Figure 2 is the architecture of the synthesis strategy by using a case-based reasoning.

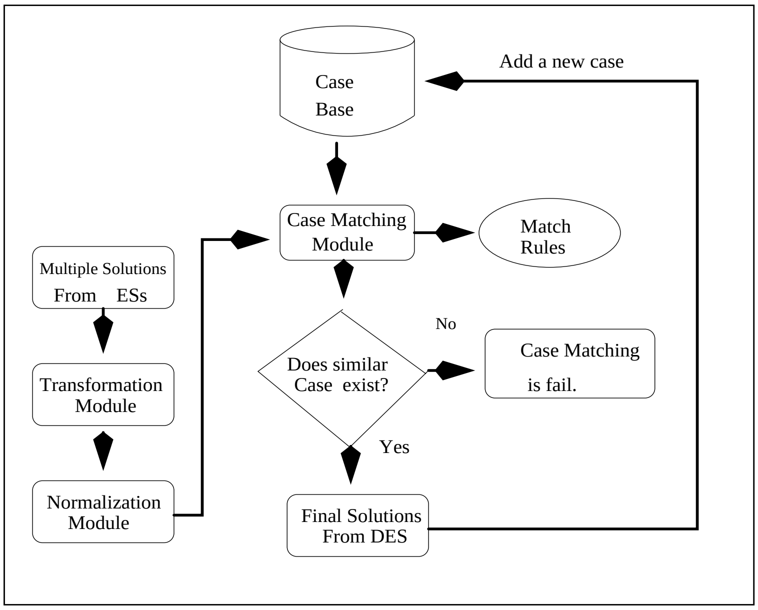

Fig. 2. The principle of the case based approach

This strategy consists of a transformation module, a normalization module, a case matching module, a case base, and a matching rule set. The strategy works in the PROBABILITY [1] inexact reasoning model. The area of uncertainties should be in the range [0, 1]. The following is a brief introduction to each module and a case base.

(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 [7] in the transformation module. For example, if an ES uses the EMYCIN [6] model, the range of $[-1, 1]$ should be transformed into the range of [0, 1] in the PROBABILITY model [1].

(b) Normalization module — After transformation, the sum of uncertainties may not satisfy the requirements in the PROBABILITY model [1]. 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. These functions will be introduced in detail in the next section.

4 Case searching procedure and examples

Definition 5: The mean error $\overline{error}$ for a case matching is defined as

$$\overline{error} = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m}|CF_{ij} - CF'_{ij}|}{m * n}$$

where CF_{ij} represents the uncertainty for *jth* value from ES_i for the input case, CF'_{ij} indicates the uncertainty for *jth* value from ES_i for an existing case, m indicates that there are m possible values for this attribute of the object, and n is the number of ESs in a DES (recall Matrix 2.2).

Definition 6: The maximum error $error_{max}$ for a case matching is defined as

$$error_{max} = max\{|CF_{ij} - CF'_{ij}|\}$$

where $i = 1, 2, ..., m$; $j = 1, 2, ..., n$.

If for a case matching, $error_{max} = 0$, we call this matching a perfect matching.

Normally, it is not easy to get a perfect matching for most of our problems. After searching the case base, the system advises the nearest matching case which most nearly solves the problem.

The classification of case matching is shown in Figure 3.

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 (L_1 in Table 4). If a matching case is not found in level 1, the searching area will be extended to level 2 (L_2), and so on until a similar case is found or whole searching space has been examined. Table 4 describes the performance for each level.

4.1 Examples

In this subsection. we demonstrate some examples by using this synthesis strategy.

Suppose that there are 3 expert systems using the PROBABILITY model to evaluate the values of an attribute of an object. Table 5 is an input case which shows the uncertainties of values obtained by ESs. The synthesis strategy is to find a similar case which can reasonably match the input case from the case base, then solve the synthesis problem by choosing the matching case.

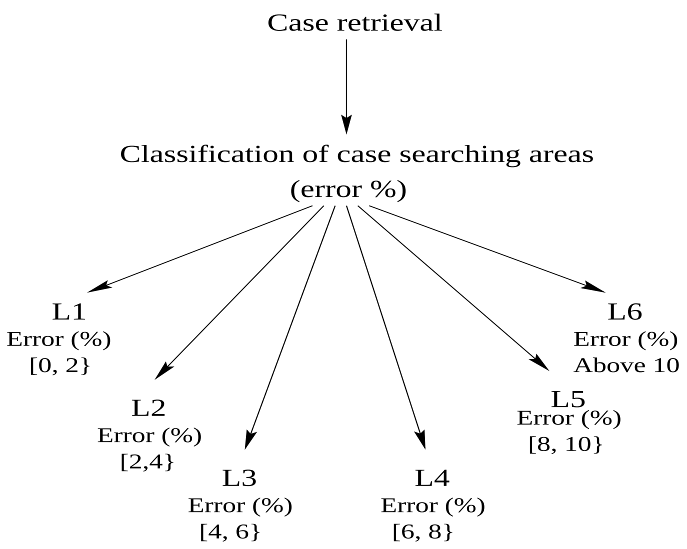

Fig. 3. Classification of case matching

$\overline{error}$	Function name	Performance	Matching result
2%	L_1	Choose the solution from the matching case	*Good*
2%–4%	L_2	Choose the solution from the matching case	$Good^-$
4%–6%	L_3	Choose the solution from the matching case	$Acceptable^+$
6%–8%	L_4	Choose the solution from the matching case	*Acceptable*
8%–10%	L_5	Choose the reference solution from the matching case	$Acceptable^-$
$> 10\%$	L_6	These is no solution for the current case from the case base	*Fail*

Table 4. Case matching rules

	ESs	$Value_1$	$Value_2$	$Value_3$
Input case	ES_1	0.02	0.14	0.60
	ES_2	0.1	0.01	0.50
	ES_3	0.11	0.03	0.80

Table 5. Example 1: Input case

After searching the case base in the area of L_1, there is no case which matches the input case in this searching area. The system then extends the searching area to L_2. In L_2, three cases are considered as suitable cases to match the input case. The detailed information for the matching results is shown in Table 6.

Matching cases	ESs	$Value_1$	$Value_2$	$Value_3$	$\overline{error}$	Matching results
$Case_1$	ES_1	0.13	0.15	0.66	2.3%	$Good^-$
	ES_2	0.15	0.01	0.55		
	ES_3	0.11	0.15	0.66		
	$Solution_1$	0.13	0.10	0.61		
$Case_2$	ES_1	0.10	0.15	0.7	2.8%	$Good^-$
	ES_2	0.15	0.01	0.57		
	ES_3	0.16	0.20	0.66		
	$Solution_2$	0.17	0.15	0.64		
$Case_3$	ES_1	0.05	0.05	0.35	3.9%	$Good^-$
	ES_2	0.1	0.11	0.70		
	ES_3	0.01	0.03	0.80		
	$Solution_3$	0.05	0.07	0.6		

Table 6. Example 1: Matching cases from the case base

From the above table we find that solutions of $Case_1$ and $Case_2$ are very similar. The matching results fall in the area of L_2 (recall Table 4). The user can choose one of solutions from the above two cases based on their preferences.

Now let us look at another example.

	ESs	$Value_1$	$Value_2$	$Value_3$
Input case	ES_1	0.20	0.40	0.20
	ES_2	0.00	0.10	0.05
	ES_3	0.05	0.05	0.90

Table 7. Example 2: Input case

Matching cases	ESs	$Value_1$	$Value_2$	$Value_3$	$\overline{error}$	Matching results
$Case_1$	ES_1	0.13	0.15	0.66	8.3%	$Acceptable^-$
	ES_2	0.15	0.01	0.55		
	ES_3	0.11	0.15	0.66		
	$Solution_1$	0.13	0.11	0.62		

Table 8. Example 2: Matching cases from the case base

In example 2, only one case in the case base can roughly match the current case. The matching result falls in L_5. In this situation, if there is no an alternative strategy available, the solution can still be used as a reference.

5 Comparison with related work

We have introduced the basic principle of a case-based strategy and its preliminary results. Case-based reasoning is an instance of analogical methods. Now we compare analogical methods with related work in this research area from a methodological point of view and analyze features of different methods.

- *An analysis method* is the methodology which can be used to evaluate *output* from *input* (as defined in Subsection 3.1) by analyzing the characteristics of *input* using mathematical theories. This method is useful for areas in which the individual solution from each expert can be described by uncertainties, or numbers, and the relationships between *inputs* and *outputs* are not so complicated [2, 7, 9].
- *An inductive method* is the methodology which can be used to find the general relationship between *inputs* and *outputs* based on sufficient patterns [10]. It is obvious that the first condition of using this method for a strategy is that enough patterns must be known (this requirement differs from the analytic methods above). The second condition is that the patterns are distributed fairly randomly. The third condition is that an inductive function can be found (such as 'converges' in neural network terms).
- *An analogical method* is the methodology which can be used to find an *output* by comparing an *input case* to similar known cases. This method is useful for application in areas with limited patterns (or cases). The advantages of analogical methods are: (1) The principle of problem solving is only based on the similarity without knowing the mathematical relationship between *inputs* and *outputs*. This advantage overcomes a limitation of analytic methods which cannot work well if the relationship between *input* and *output* cannot be summarized by mathematical models. and (2) The major limitation of inductive methods is that they need too many real examples which are not available in most application domains. Analogical methods can overcome the limitation of inductive methods since analogical methods only need some real examples. Disadvantages of analogical methods are that (1) the final solution after synthesizing multiple solutions is not very precise; and (2) the adaptation of a new solution by a used solution is a hard problem and has not been solved so far.

After comparison, our conclusion is that they compensate each other. Based on different kinds of synthesis problems and available information, different methods should be used to design synthesis strategies.

6 Conclusion

In this paper, a basic principle of new synthesis strategy has been proposed for solving synthesis problems in DESs and results of using this synthesis strategy have been demonstrated. This strategy is developed based on an analogical method by case-based reasoning. This preliminary research has proved that analogical methods can overcome some limitations of analytic and inductive methods. Case-based reasoning is considered as a good tool to implement an analogical method [4]. The original contribution of the strategy presented is to offer an alternative way for solution synthesis in DESs.

Further work aims at researching the problem of case adaptations and investigating how to combine analogical methods with analytic and inductive methods.

Acknowledgments

The author wishes to thank DR Chengqi Zhang and MR Jirapun Daengdej for their support during this work at the University of New England.

References

1. R. Duda, P. Hart and N. Nilsson (1976). Subjective Bayesian Method for Rule-based Inference System, *AFIPS*, **Vol. 45**, pp. 1075-1082.
2. N. A. Khan & R. Jain (1985). Uncertainty Management in a Distributed Knowledge Based System, *Proceedings of 5th International Joint Conference on Artificial Intelligence*, Los Angeles, pp. 318-320, 1985.
3. V. Lesser and D. Corkill (1983). The Distributed Vehicle Monitoring Testbed, *AI Magazine,* **Vol. 4,** pp. 63-109.
4. K. Sycara (1987). Planning for Negotiation: A Case-Based Approach, *Proceedings of Darpa Knowledge-Based Planning Workshop*, pp. 11.1-11.10.
5. C. K. Riesbeck and R. C. Schank (1989). Inside Cased-based Reasoning, Lawrence Erlbaum Associates Publishers, New Jersey.
6. 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.
7. C. Zhang (1992). Cooperation Under Uncertainty in Distributed Expert Systems. *Artificial Intelligence*, **Vol. 56**, pp. 21-69.
8. M. Zhang, Synthesis of Solutions in Distributed Expert Systems, *PhD Thesis*, The University of New England, Australia, 1995.
9. M. Zhang and C. Zhang (1994a). A Comprehensive Strategy for Conflict Resolution in Distributed Expert Systems, *Australian Journal of Intelligent Information Processing Systems*, **Vol. 1**, No. 2, pp. 21-28.
10. M. Zhang and C. Zhang (1996). 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.
11. M. Zhang and C. Zhang (1996). 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.

This article was processed using the LaTeX macro package with LLNCS style