A HYBRID CASE-BASED REASONING ARCHITECTURE AND ITS APPLICATION

Chien-Chang Hsu[1] *and Cheng-Seen Ho*[2]

[1]Department of Information Management, Ming Chuan University,
5 Teh-Ming Rd., Gwei Shan District, Taoyuan County, Taiwan 333
[2]Department of Electronic Engineering, National Taiwan University of Science and Technology,
43 Keelung Road, Section 4, Taipei, TAIWAN 106
[1]cch133@mcu.edu.tw and [2]csho@ et.ntust.edu.tw

ABSTRACT

This paper proposes a hybrid CBR architecture to help CBR reasoning. It hybridizes CBR, fuzzy neural networks, induction, utility-based decision theory, and knowledge-based planning technology to facilitate solutions finding. The basic mechanism is CBR which accumulates experiences as cases in the case library and proposes solutions by adapting the old cases that have successfully solved similar previous case. The distributed fuzzy neural network is introduced to perform approximate matching to tolerate potential noise in case retrieval. The induction technology along with relevance theory is used in case selection, adaptation, and retaining. Knowledge-based planning is used as a general architecture for case adaptation by creating an adaptation plan, whose execution, in turn, proposes a solution. Hybridizing these techniques in the CBR module can effectively produce a high-quality solution for a given problem.

1. INTRODUCTION

Case-based reasoning (CBR) is a problem solving method that maps problem features to potential solutions through the process of case retrieval, case selection, case adaptation, and case retaining. However, a solution may get involved in a huge number of problem features. How do we know which are relevant or significant? How do we correctly and completely identify them? How do we efficiently retrieve relevant past cases that are most worth adaptation without too much adaptation effort? How do we manipulate candidate cases in order to generate a new solution that fits a given problem? Finally, how do we maintain a case library so that it only assimilates worthy adapted cases? These are "classical" issues associated with CBR. Currently published CBR systems have proposed a variety of hybridization techniques to solve all or some of the issues. Their efforts, however, still haven't promoted CBR into one of the main stream intelligent problem solving methods.

We noticed that the following shortcomings appear in current CBR systems and need to be carefully addressed. First, most existent CBR systems use a hierarchy structure to index the cases in the case library. The hierarchy structure extracts common features into a prototype that classifies cases into different groups or classes to facilitate search of similar cases. It is, however, not appropriate for a static index hierarchy, to be used in different types of problems since the significance of a feature may change in different contexts. Second, some CBR systems use surface features and complex similarity measurement to find the most similar case for a given problem without considering the adaptability of the case. The most similar case, however, does not guarantee to be the most adaptable. Finally, many CBR systems use context-dependent adaptation knowledge to do case adaptation. The adaptation mechanisms are *ad hoc* and few of them can be re-used in other domains.

This paper proposes a hybrid case-based reasoner (HCBR) architecture to alleviate the above issues. It hybridizes a

distributed fuzzy neural network with induction, utility-based decision theory, and knowledge-based planning technology to facilitate CBR. The basic mechanism is CBR. The distributed fuzzy neural network performs approximate matching to tolerate potential noise in case retrieval. The induction technology along with relevance theory is used in case selection, adaptation, and retaining, which helps a lot in hammering out valuable features for the target case from existent ones and in pruning unnecessary search space. Knowledge-based planning is used as a general architecture for case adaptation. It creates an adaptation plan from an adaptation tree that covers all the relevant problem features, satisfies all the relevant constraints, and contains all the cases whose expected utilities are over a threshold. Execution of the case adaptation plan can successfully propose solutions.

To validate the approach, the hybridized techniques are applied in the medicine domain to do medical diagnosis with multiple diseases. The experiment shows that hybridizing these techniques in the CBR paradigm does effectively produce a high quality solution for a given medical consultation.

The rest of paper is organized as follows. Section 2 gives a description of the HCBR architecture. Section 3 then explores the case retrieval and selection, while Section 4 elaborates case adaptation and retaining. Section 5 illustrates how the HCBR system works on medical diagnosis. Section 6 concludes the work and makes some comparisons with other works.

2. SYSTEM ARCHITECTURE

Fig. 1 shows the architecture of HCBR. It contains six major modules, namely, case library, case retrieval, case selector, case adapter, adaptation plan library, and case retainer. Basically, the case library contains instances of problem data and solutions in form of feature-value pairs, each called a case. A problem data contains two types of features, namely, general features and specific features. The former describes domain-related general information about the problem. The latter describes domain-specific information about the problem. Take medical diagnosis as an example, the general features refer to subjective findings and objective findings, while specific features refer to pathology and laboratory data. All cases are pre-classified into a set of categories to facilitate retrieval. Each category is further refined into a set of sub-types to help the training process of the distributed neural network used in the case selector.

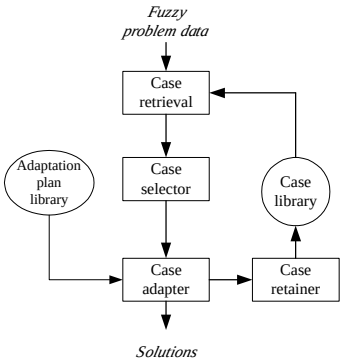


Fig. 1 Hybrid case-based reasoner

The case retrieval takes the fuzzy problem data as a pattern to retrieve the cases from the case library. Candidate cases are those that are mostly likely to be useful to solve this data by a distributed fuzzy neural network. The case selector then induces an induction tree from the features of the selected cases and calculates expected utility for each feature. Note that the output of the case retrieval may contain more than one candidate case. All of them contain features that are somewhat related to the characteristics of the problem. The induction tree is a decision tree with constraint links that guide the selection of most possible diagnoses. It provides a way to structure the features in terms of their usefulness. It is easy for the selector to follow the tree to calculate expected utility for each node, i.e., each feature in the tree. The calculation process, basically, compares the difference of the context of the feature value against the given case and accordingly estimates the adaptability of the feature. The adaptability is then transformed into expected utility by decision theory for the feature. The expected utility serves as a metric for selecting cases for adaptation.

The case adapter does actual adaptation with the help of the adaptation plan library by following the induction tree. The basic adaptation strategy is as follows. It first creates a subtree from the induction tree, called adaptation tree, that covers all the

problem features, satisfies all the relevant constraints, and contains no nodes whose expected utilities are below a threshold. It then generates a feature adaptation plan for each node in the adaptation tree from the adaptation plan library, which contains experienced feature adaptation plans and adaptation operators. It finally produces an adapted case severing as a solution to the problem data by executing each above plan to adapt the corresponding value. This may involve the manipulation of feature values that appear in multiple adaptation paths, i.e., multiple solutions.

Finally, the adapted case is sent to the case retainer to see whether it deserves storage in the case library. It uses the adaptation tree to check whether there already exist analogous or subsumed cases in the case library. This approach of case retaining features the comparison of all existing candidate cases (represented by the adaptation tree) at the same time, which reduces the growing speed of the case library and the re-training need of the distributed neural network.

3. CASE RETRIEVAL AND SELECTION

3.1 Case Retrieval

The retrieval of candidate cases is performed by a distributed fuzzy neural network that contains two layers of nets (Fig. 2). The first layer is a general net. It determines the similarity between a given problem and the cases in the case library with respect to the general problem features and hypothesized solution types. The second layer is a specific net, which does the similar job on the specific problem features. Each layer is further partitioned into sub-nets in accord with the sub-types of the cases. The main advantage of this sub-net design is to alleviate re-training complexity; when a new case is to be introduced into the case library, we only need to re-train the corresponding sub-nets while keeping the others intact. In addition, introducing fuzzy representation avoids the problem of matching nothing from the case library.

Each sub-net is a fuzzy ARTRON model by combining the Fuzzy ART architecture and the superposed perceptrons [4]. The fuzzy ART architecture allows for flexible partition of the input feature space and self-regulation of the cluster nodes. The superposed perceptrons allows for dynamic association from the cluster nodes to the class nodes. The node number of the input node layer of each sub-net is related to the number of problem features. Each feature is encoded by the importance of a *numeric* feature value or by the feature code of a *symbolic* feature value [8]. Note that the input node layer does complement coding on the input vector before sending it out. The outputs of the general net are the case numbers of the selected cases. They are fed, along with the specific problem data, into each sub-net of the specific net to find all the candidate cases.

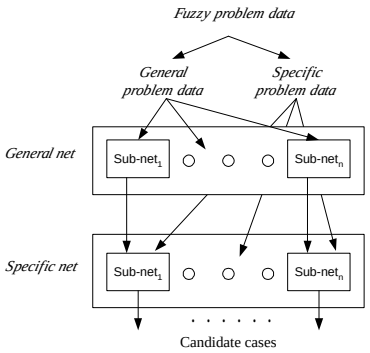


Fig. 2 Distributed fuzzy neural network

3.2 Case Selection

Given a set of candidate cases, HCBR needs to do evaluation before being actually selected for case adaptation. The following two concepts are involved in this evaluation process. First, the evaluation is based on the utility of each feature to support the subsequent feature-based adaptation. Second, the evaluation is on all the candidate cases, not just a single one to improve the quality of the subsequent adapted solution. First, to take care of all the candidate cases at the same time, HCBR induces their features into an induction tree. The construction algorithm for an induction tree is shown in Fig. 3, which is based on the CLS algorithm [3, 12]. Fig. 4 exemplifies an induction tree. Note that each nonterminal node in an induction tree represents a feature of the candidate cases. Each outgoing link from a node represents a value for the node. Each leaf of an induction tree stores the surface feature similarity of each candidate case. One interesting feature of this tree is that feature values that appear in more candidate cases are grouped more closely to the root for easy and fast subsequent inspection.

1. If all the feature value v_i, $i=1..n$ in each case of training set S are the same, then create a same node and go to step 7.
2. Otherwise, select the attribute in the following sequence:
   A、 Problem data (both general and specific problem features)
   B、 Solution
3. Select an attribute A with values v_i, $i=1..n$ and create a decision node.
4. Partition the training features in training set S into subsets s_i, $i=1..n$ according to the value of v_i.
5. Compute the probability $P(A \grave{\imath} s_i)$ for each attribute value.
6. Apply the algorithm recursively to each of the set s_i.
7. Create a constraint node for each constraint c_i; connect a constraint link to each node in c_i.

Fig. 3 Construction algorithm of induction tree

	A	B	D	E	Constraint 1
Case (C_1)	V_{11}	V_{21}	V_{31}	V_{41}	
Case (C_2)	V_{12}	V_{21}	V_{32}	V_{42}	C_{21}: (A, D)
Case (C_3)	V_{11}	V_{21}	V_{31}	V_{43}	
Case (C_4)	V_{11}	V_{21}	V_{32}	V_{44}	C_{41}: (B, E)

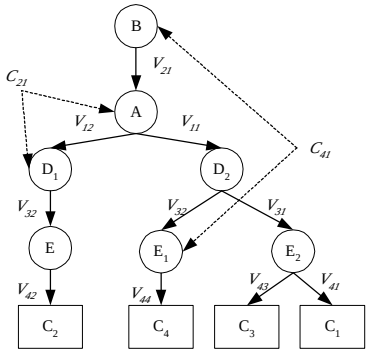


Fig. 4 Example induction tree

HCBR evaluates the expected utility for each feature in the tree. This involves the concept of adaptability of a feature value, formally defined below.

$$AD(A_j \to A_{jk}) = 1 - DV(A_j \to A_{jk}) \bullet DWV(A_j \to A_{jk})[1 - m(A_j \to A_{jk})], \quad (1)$$

where **AD** stands for adaptability, $A_j \grave{\imath} A_{jk}$ represents a value of the feature A_j, **DV** is the difference vector, **DWV** is the difference weight vector, and m is the domain model difference between the given case and the candidate case [8]. The calculated adaptability value can be further transformed into one of the following fuzzy adaptability values: *definite, high, medium, low,* and *hard* as shown in Fig. 5.

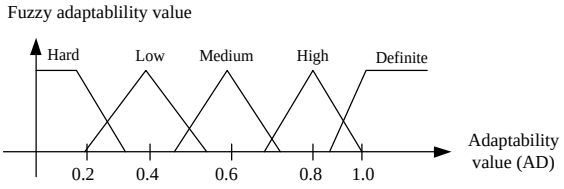


Fig. 5 Fuzzy partition for case adaptability value

Now, we can use the feature value adaptability to recursively calculate expected utility for each feature as follows.

$$EU(A_j) = \sum_{k=1}^{n} EU(A_{jk}) * AD(A_j \to A_{jk}) * P(A_j \to A_{jk}), \quad (2)$$

where A_{jk} stands for the kth child node of the feature node A_j , n is the number of children of the feature node A_j , $AD(A_j \to A_{jk})$ is the adaptability value of the feature value represented by link $A_j \to A_{jk}$, and $P(A_j \to A_{jk})$ stands for the probability of the feature value represented by link $A_j \to A_{jk}$.

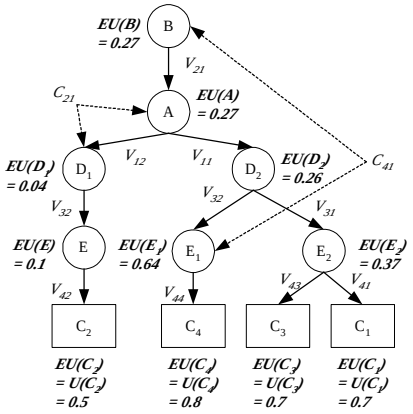
The calculation of expected utility starts by setting the expected utility of the leaf node C_i to C_i's similarity, i.e., $EU(C_i)=S_i$, where S_i represents the similarity of case C_i to the given problem [13]. The $AD(A_j \grave{\imath} A_{jk})$ is calculated by Equation (1). The probability $P(A_j \grave{\imath} A_{jk})$ represents the occurrences of the feature value $A_j \grave{\imath} A_{jk}$ in the case library, and is calculated by Equation (3).

$$P(A_j \to A_{jk}) = \frac{N(A_j \to A_{jk})}{\sum_{m=1}^{n} N(A_j \to A_{jm})}, \quad (3)$$

where $N(A_j \to A_{jk})$ is the occurrences of $A_j \to A_{jk}$, n is the number of siblings, and k is the kth sibling of A_j, $1 \le k \le n$.

The $EU(C_i)$ is then backed up to its father node by Equation (2) to compute its father's expected utility. This process repeats until it reaches the root node. Fig. 6 exemplifies the computation of expected utility for Fig. 4

This evaluation method essentially says that the more frequently a feature value occurs in the case base, the higher the probability that the value will occur again in the new problem. The evaluation is based on the ability of adaptation from the viewpoint of both cases and individual features. It uses the expected utility $EU(A_j)$ to estimate how useful it would be to use the subtree rooted at A_j for adaptation; higher expected utility means higher relevance [7, 11].



Link	Value	Adaptability	Probability
$B \rightarrow A$	V_{21}	1.0	1.0
$A \rightarrow D_1$	V_{12}	0.6	0.33
$A \rightarrow D_2$	V_{11}	1.0	0.67
$D_1 \rightarrow E$	V_{32}	0.4	1.0
$D_2 \rightarrow E_1$	V_{32}	0.8	0.57
$D_2 \rightarrow E_2$	V_{31}	0.6	0.43
$E \rightarrow C_2$	V_{42}	0.2	1.0
$E_1 \rightarrow C_4$	V_{44}	0.8	1.0
$E_2 \rightarrow C_3$	V_{43}	0.6	0.67
$E_2 \rightarrow C_1$	V_{41}	0.4	0.33

Fig. 6 Example of expected utility computation

4. CASE ADAPTATION AND RETAINING

4.1 Case Adaptation

The basic strategy to do case adaptation contains three steps, namely, adaptation tree creation, adaptation plan generation, and adaptation plan execution, to be detailed below. First, the adaptation process creates an adaptation tree from a subtree of the induction tree by pruning the nodes whose expected utilities are below a threshold. It then re-arranges the nodes that are under constraints into a proper causal sequence using the constraint adaptation operators. It also checks for those features that contain no values in the candidate cases. The corresponding feature values in the problem data will be used for the features if they satisfy the related constraints. Conversely, if some feature in the problem data contains no value, the process computes and uses the following *closeness* measurement to decide how to adapt the case.

$$Closness(A_j \rightarrow A_{jk}) = P(A_j \rightarrow A_{jk}) * PR_i(A_j \rightarrow A_{jk}), \quad (4)$$

where $P(A_j \rightarrow A_{jk})$ is the occurrence probability of the feature value $A_j \rightarrow A_{jk}$ and $PR_i(A_j \rightarrow A_{jk})$ is the proximity of the feature value $A_j \rightarrow A_{jk}$ to the solution type i. If the closeness is above a threshold, the corresponding feature value in the candidate case is considered to be relevant to the new problem and retained in the adaptation tree. In this case, the judged solution is stated to be true "under the condition that the feature value $A_j \rightarrow A_{jk}$ occurred," i.e., **IF** $(A_j \rightarrow A_{jk})$ **THEN** (solutions). If the closeness is below the threshold, the corresponding feature value as well as the associated constraints is removed. This strategy of handling null feature values in adaptation can reduce most user intervention. Fig. 7 shows the adaptation tree created from Fig.6.

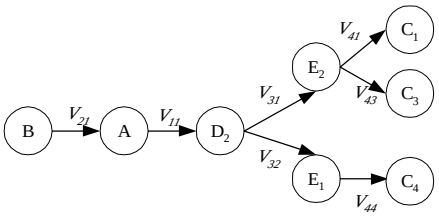


Fig. 7 Example adaptation tree

Second, the adaptation process develops a case adaptation plan from the adaptation tree with the help of a feature adaptation library by creating a feature adaptation plan for each node. Basically, if a feature adaptation plan that can reduce the **DV** of the feature can be found in the feature adaptation plan library, it is selected. If none of such feature adaptation plan exists, a partial-order planning (POP) planner [2, 10] is called to produce a new feature adaptation plan from the adaptation operators of the plan library for the feature. Fig. 8 depicts the POP algorithm to do feature adaptation planning. Note that the algorithm resolves the multiple paths problem by choosing a path that has the minimum *adaptation effort* (Step 6). Adaptation effort *(AE)* is defined below to calculate the adaptation cost of an adaptation step in a feature adaptation plan.

$$AE(S_i) = \mathbf{DV}(S_i) \bullet \mathbf{DWV}(S_i), \quad (5)$$

where S_i is the i^{th} step in the feature adaptation plan. Fig. 9 illustrates a feature adaptation plan with multiple paths. Note that each step contains an *AE* value and the path containing Steps D_1 and D_2 is selected as the feature adaptation plan since its total path *AE* is minimum.

Step 1: Initialize a plan that contains a start and a finish step.
   a. $DV_{Postcondition}$ of the start step equals the DV of the feature value,
   b. $DV_{Precondition}$ of the finish step is a zero vector.
Step 2: Pick a plan step with $DV_{Precondition}$ that has not been satisfied.
Step 3: Create a new step containing an operator whose precondition matches the $DV_{Precondition}$.
Step 4. Go to step 3 until all matched operators are applied.
Step 5: Go to step 2 until all $DV_{Precondition}$ of all steps have been satisfied.
Step 6: Find a minimal adaptation effort path.

Fig. 8 Feature adaptation planning

Step #	Feature name	Feature value	$DV_{Precondition}$	$DV_{Posconditiont}$	AE	Adaptation operator
Start	D	V_{51}	N/A	<1, 0, 1, 1, 0>	0.5	N/A
D_1	D	V_{51}	<1, 0, 1, 1, 0>	<0, 0, 0, 0, 1>	0.2	Heuristic adjustment
D_2	D	V_{52}	<0, 0, 0, 0, 1>	<0, 0, 0, 0, 0>	0	Constraint deletion
D_3	D	V_5	<1, 0, 1, 1, 0>	<1, 0, 1, 0, 0>	0.35	Problem abstraction
D_4	D	V_{50}	<1, 0, 1, 0, 0>	<1, 0, 0, 0, 0>	0.1	Problem Refinement
D_5	D	V_{52}	<1, 0, 0, 0, 0>	<0, 0, 0, 0, 0>	0	Value specialization
Finish	D	V_{52}	<0, 0, 0, 0, 0>	N/A	0	N/A

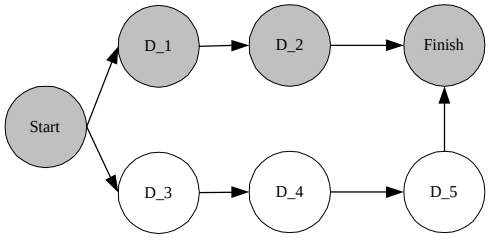


Fig. 9 Example feature adaptation plan

Finally, the adaptation process follows the case adaptation plan to adapt the feature values in the adaptation tree to meet the problem features in order to produce a new solution. If there are multiple paths in the adaptation tree, each path has to be visited in order to take care of multiple solutions. This process eventually generates a new case containing judged solution types. Fig. 10 shows the newly adapted case given the problem data of Table 1 and the adaptation tree in Fig.7. Note that the solution in Fig. 10, i.e., S_1, S_3, and S_4, contains the solutions from Cases 1, 3, and 4.

1. B: V_{21}
2. A:V_{11}
3. D:V_{33}
4. E: V_{45}
5. Constraints: [B(V_{21}), E(V_{45})]

- Solution model
  Solution type: T_1
  Solution: S_1, S_3, and S_4

Fig. 10 Newly adapted case

Table 1 Example problem data

Feature	A	B	D	E
Value	V_{11}	V_{21}	V_{33}	V_{45}

4.2 Case Retaining

The adaptation tree servers as a clue to whether we need to store a newly adapted case in the case library for later reuse. This is done by calculating an average adaptation effort for a new case, defined as follows.

$$AAF = \frac{1}{m}\sum_{k=1}^{m}\sum_{j=1}^{n} DV(A_j \to A_{jk}) \bullet DWV(A_j \to A_{jk}), \quad (6)$$

where m is the number of cases in the adaptation tree, n is the number of features in the new case, and $\boldsymbol{DV}$ is the feature different vector and $\boldsymbol{DWV}$ is the difference weight vector for the feature $A_j \to A_{jk}$. If the value of AAF is over some reuse threshold (e.g., 10%), the new case is worth storage in the case library. The philosophy for this case retaining strategy is that a new case with AAF below the reuse threshold is highly similar to some cases in the case library. It can be re-produced the second time without much effort. One immediate benefit of this approach is that the index structure in the distributed fuzzy network need not be re-trained too often. The apparent advantage of this philosophy is thus the case library only grows when its coverage is too narrow.

5. MEDICAL DIAGNOSIS AS AN APPLICATION

Medical diagnosis from surface etiology is difficult since there involve lots of complications. A clinician has to carefully investigate a patient's symptoms, chief complaints, and pathology examination in order to decide possible diseases. It takes years of training and practice for a physician to make correct decisions. This worsens when the related etiology is hard to discern or multiple diseases suffered. The rapidly growing medical knowledge and new patient cases make the diagnosis process even more difficult. Updating the medical knowledge incrementally in a traditional medical diagnosis system to cope with this is not that easy [6]. It can be made easier and reliable, however, if supplied with a system that contains and provides recommendations from the past diagnosis cases of different morbidity, since the clinician can benefit a lot from these prior

cases. This implies that the CBR approach is appropriate to the problem.

For medical diagnosis, we define a case to be a structure that contains a patient model and a diagnosis model [8]. The patient model describes subjective findings, objective findings, and pathology and laboratory examinations about a patient. The diagnosis model records the scenario of how a diagnosis is proceeded. It serves as the diagnosis the case recommends. It includes judged disease types, affected organs, and impression. A judged disease type refers to one of the 11 disease categories [5]. An affected organ indicates a human body's system that was under attack by the diseases. The impression indicates the judged disease names. The feature-value pair format is used in a case to represent the information in both models. Some features fuzzy linguistic terms to denote the importance of their values. Finally, a case is also annotated with specific adaptation knowledge and differential diagnosis knowledge. The former helps constrain the causal relations among the symptoms, test data, and diseases, and hypothesize the suspected diseases from the adapted case data. The latter helps differentiate diseases with analogous morbidity.

Our medicine case library contains 280 cases, including 20 cases for “cardiology,” 20 cases for “respiratory,” 30 cases for “hematology,” 35 cases for “GIH (gastric, intestine, and hepatology),” 20 cases for “neoplasm,” 20 cases for “urology,” 40 cases for “immune,” 30 cases for “infectious,” 30 cases for “endocrine,” 20 cases for “neurology,” and 15 cases for “miscellaneous” [5].

There are 22 nodes in the input node layer of each sub-net in the symptom net [8]. They represent the following subjective findings and objective findings in the patient data: sexuality, history 1, history 2, chief compliant, present illness 1, present illness 2, present illness 3, present illness 4, temperature, pulse, respiratory rate, blood pressure, consciousness, HEENT, neck, heart, thorax and lung, breast, abdomen, extremity, urinary, and neurologic. The training data for each subnet are taken from the case library and encoded in numeric values between 0 and 0.87. The vigilance parameter ρ in the fuzzy ARTRON is set to 0.9 with learning rate set to 1 initially. We have shown in [8] that the architecture can successfully produce a solution that contains multiple diagnoses for a given patient data input.

6. CONCLUSION

We have described a hybrid CBR architecture and how it is applied in medical diagnosis. It hybridizes CBR, fuzzy neural networks, induction, utility-based decision theory, and knowledge-based planning technology to facilitate solutions finding. The basic mechanism is CBR that accumulates experiences as cases in the case library and proposes solutions by adapting the old cases that have successfully solved previous cases. The distributed fuzzy neural network performs approximate matching to tolerate potential noise in case retrieval. The induction technology along with relevance theory is used in case selection, adaptation, and retaining, which helps a lot in hammering out valuable features for the target case from existent ones and in pruning unnecessary search space. Knowledge-based planning is used as a general architecture for case adaptation. It creates an adaptation plan from an adaptation tree that covers all the relevant problem features, satisfies all the relevant constraints, and contains all cases whose expected utility are over a threshold. Execution of the case adaptation plan can successfully propose high-quality solutions.

In summary, the case retrieval accepts fuzzy input, which is more natural and efficient, with the help of fuzzy neural network. The case adaptation is effective with the help of feature expected utilities. It also dynamically tackles multiple cases with the help of the adaptation tree. Moreover, the case adaptation is a planning-based general mechanism, which is thus unlikely to be performance-degraded. Finally, the case retaining is efficient with the help of the adaptation tree.

We have applied the hybridized techniques in the medicine domain to do medical diagnosis with multiple diseases. The experiment shows that the architecture can successfully interact with the physician with different proficiency levels in collecting complete patient data, which in turns leads to a correct diagnosis of multiple diseases. The best thing is that it also successfully

discovers new medical adaptation knowledge from the relevant cases [8]. The application of the technique in the medicine domain not only improves the quality of the diagnosed diseases but reduces the burden of a physician.

Most current integrations of induction and CBR are applied in case retrieval. For instance, Inreca+ integrates CBR and induction for diagnosing poison cases caused by psycholotropes. It focuses on case retrieval and uses Inreca trees and compiled knowledge to measure case similarity [1]. M. C. Jaulent et al. [9] uses CBR to diagnose histopathology. It employs an index tree to structure specific features of a case, and uses structure similarity as a metric for searching of cases. They focus on using induction to support case retrieval and/or similarity measurement by constructing a static induction tree to index the case library. They neither can reflect problem contexts in CBR reasoning nor can handle complex situations like multiple solutions or noise feature values. Even worse, the maintenance of the index structure is difficult if new cases need to be frequently added to the library. Our approach is quite different in the integration and application of the various techniques. It not only supports case retrieval, but also supports case selection, case adaptation and knowledge discovery [8].

7. REFERENCES

[1] K. D. Althoff, R. Bergman, S. Wess, M. Manago, E. Auriol, O. I. Laricher, A. Bolotor, Y. I. Zhuravlev, S. I. Gurov, “Case-Based Reasoning for Medical Decision Support Tasks: The Inreca Approach,” *Artificial Intelligence in Medicine,* vol. 12, no. 1, pp. 25-41, 1998.

[2] A. Barrett and D. Weld, "Partial Order Planning: Evaluating Possible Efficiency Gains," *Artificial Intelligence,* vol. 67, no. 1, pp. 71-112, 1994.

[3] P. R. Cohen and E. A. Feigenbaum, *Handbook of Artificial Intelligence,* vol. III, Willian Kaufmann Pub, Los Altos, CA., 1982.

[4] J. S. Chou and C. S. Ho, "A Fuzzy-ART-Enhanced Neural Classifier," *Proceedings of the third International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES'99),* pp. 488-491, Adelaide, SA, Australia, 1999.

[5] A. S. Fauui, E. Brauwald, K. J. Isselbacker, J. D. Wilson, J. B. Martin, D. L. Kasper, S. L. Hauser, and D. L. Longo, *Harrison’s Principles of Internal Medicine,* 14th ed., McGraw-Hill, NY, 1998.

[6] L. Gierl, M. Bull, and R. Schmidt, CBR in Medicine, in M. Lenz, S. B. Bartsch, & S. Wess (Eds.) Case-Based Reasoning Technology, From Foundations to Applications, Springer, Berlin, pp. 273-297, 1998.

[7] M. Hadzikadic, B. F. Bohren, “Learning to Predict: INC2.5,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 9, no. 1, pp. 168-173, 1997.

[8] C. C. Hsu, A Hybrid Case-Based Reasoning Architecture and Its Application, Ph. D. Dissertation of National Taiwan University of Science and Technology, Taiwan, 2000.

[9] M. C. Jaulent, C. L. Bozee, E. Zapletal, and P. Degoulet, A Case-Based Reasoning Method for Computer-Assisted Diagnosis in Histopathology, Proceedings of AI in Medicine (AIME'97), Grenoble, France, pp. 39-242, 1997.

[10] S. Kambhampti, C. A. Knoblock, and Q. Yang, "Planning as Refinement Search: A Unified Framework for Evaluating the Design Tradeoffs in Partial Order Planning," *Artificial Intelligence,* vol. 76, no. 1-2, pp. 107-238, 1995.

[11] J. Kolodner, *Case-Based Reasoning,* Morgan Kaufmann, San Mateo, CA., 1993.

[12] J. R. Quinlan, "Introduction of Decision Trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.

[13] A. Tversky, "Features of Similarity," *Psychological Review,* vol. 84, pp. 327-352, 1977.