130: Rule-based Expert Systems

Ajith Abraham
Oklahoma State University, Stillwater, OK, USA

1 PROBLEM SOLVING USING HEURISTICS

A general introduction to artificial intelligence methods of measurement signal processing is given in **Article 128, Nature and Scope of AI Techniques, Volume 2**.

Problem solving is the process of finding a solution when the path leading to that solution is uncertain. Even though we are familiar with several problem-solving techniques, in the real world, sometimes many problems cannot be solved by a technique we are familiar with.

Surprisingly, for some complicated problems, no straightforward solution technique is known at all. For these problems, heuristic solution techniques may be the only alternative. A heuristic can be simplified as a strategy that is powerful and general, but not absolutely guaranteed to provide best solutions.

Heuristic methods are very problem specific. Previous experience and some general rules – often called *rules of thumb* – could help find good heuristics easier. Humans use heuristics a great deal in their problem solving. Of course, if the heuristic does fail, it is necessary for the problem solver to either pick another heuristic, or know that it is appropriate to give up.

Choosing random solutions, adopting greedy approaches, evolving the basic heuristics for finding better heuristics are just some of the popular approaches used in heuristic problem solving (Michalewicz and Fogel, 1999).

Heuristic problem solving involves finding a set of rules, or a procedure, that finds satisfactory solutions to a specific problem. A good example is finding one's way through a maze. To make the way toward the final goal, a step-by-step movement is necessary. Very often false moves are made but in most cases we solve the problem without much difficulty. For the maze problem, a simple heuristic rule could be '*choose the direction that seems to make progress*'. Another good example is the job shop scheduling problem wherein the task is to schedule J_n independent jobs, where $n = \{1, 2, \ldots .N\}$ on R_m heterogeneous resources and $m = \{1, 2, \ldots, M\}$, with an objective of minimizing the completion time of all the jobs and utilizing all the resources effectively.

Each job J_n has processing requirement P_j cycles and resource R_m has speed of S_i cycles/unit time. Any job J_n has to be processed in resource R_m, until completion. If C_j is the completion time and the last job j finishes processing, then we define $C_{\max} = \max\{C_j, j = 1, \ldots, N\}$, the make-span and ΣC_j, as the flow-time.

The task is to find an optimal schedule that optimizes the flow-time and make-span. Some simple heuristic rules to achieve this are by scheduling the Shortest Job on the Fastest Resource (SJFR), which would minimize ΣC_j or by

Handbook of Measuring System Design, edited by Peter H. Sydenham and Richard Thorn.
© 2005 John Wiley & Sons, Ltd. ISBN: 0-470-02143-8.

scheduling the Longest Job on the Fastest Resource (LJFR), which would minimize $C_{\max}$.

Minimizing ΣC_j asks that the average job finishes quickly, at the expense of the largest job taking a long time, whereas minimizing $C_{\max}$, asks that no job takes too long, at the expense of most jobs taking a long time.

In summary, minimization of $C_{\max}$ will result in maximization of ΣC_j, which makes the problem more interesting.

By contrast, algorithms are straightforward procedures that are guaranteed to work every time for they are fully determinate and time invariant. For example, certain daily routine tasks could be formulated in a strict algorithm format (example, starting up an automobile). However, for a ‘problem solver’ to be more adaptive, novel elements or new circumstances must be introduced. Many real-world problems cannot be reduced to algorithms, which leads us to the quest to find more powerful techniques.

2 WHAT ARE RULE-BASED SYSTEMS?

Conventional problem-solving computer programs make use of well-structured algorithms, data structures, and crisp reasoning strategies to find solutions. For the difficult problems with which expert systems are concerned, it may be more useful to employ heuristics: strategies that often lead to the correct solution, but that also sometimes fail.

Conventional rule-based expert systems, use human expert knowledge to solve real-world problems that normally would require human intelligence. Expert knowledge is often represented in the form of *rules* or as *data* within the computer.

Depending upon the problem requirement, these rules and data can be recalled to solve problems. Rule-based expert systems have played an important role in modern intelligent systems and their applications in strategic goal setting, planning, design, scheduling, fault monitoring, diagnosis and so on.

With the technological advances made in the last decade, today’s users can choose from dozens of commercial software packages having friendly graphic user interfaces (Ignizio, 1991). Conventional computer programs perform tasks using a decision-making logic containing very little knowledge other than the basic algorithm for solving that specific problem. The basic knowledge is often embedded as part of the programming code, so that as the knowledge changes, the program has to be rebuilt. Knowledge-based expert systems collect the small fragments of human know-how into a knowledge base, which is used to reason through a problem, using the knowledge that is appropriate. An important advantage here is that within the domain of the knowledge base, a different problem can be solved using the same program without reprogramming efforts. Moreover, expert systems could explain the reasoning process and handle levels of confidence and uncertainty, which conventional algorithms do not handle (Giarratano and Riley, 1989). Some of the important advantages of expert systems are as follows:

- ability to capture and preserve irreplaceable human experience;
- ability to develop a system more consistent than human experts;
- minimize human expertise needed at a number of locations at the same time (especially in a hostile environment that is dangerous to human health);
- solutions can be developed faster than human experts.

The basic components of an expert system are illustrated in Figure 1. The knowledge base stores all relevant information, data, rules, cases, and relationships used by the expert system. A knowledge base can combine the knowledge of multiple human experts. A rule is a conditional statement that links given conditions to actions or outcomes. A frame is another approach used to capture and store knowledge in a knowledge base. It relates an object or item to various facts or values. A frame-based representation is ideally suited for object-oriented programming techniques. Expert systems making use of frames to store knowledge are also called *frame-based expert systems*.

The purpose of the inference engine is to seek information and relationships from the knowledge base and to provide answers, predictions, and suggestions in the way a human expert would. The inference engine must find the right facts, interpretations, and rules and assemble them correctly. Two types of inference methods are commonly used – Backward chaining is the process of starting with conclusions and working backward to the supporting facts. Forward chaining starts with the facts and works forward to the conclusions.

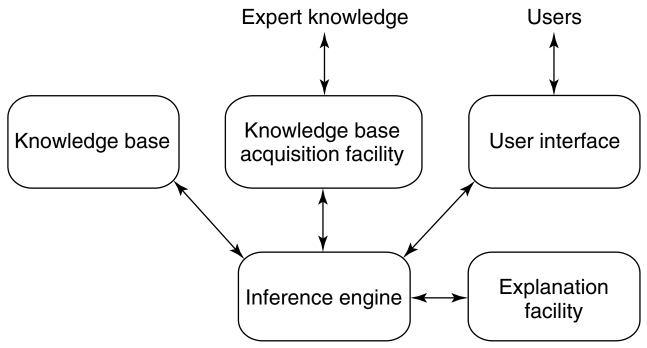


Figure 1. Architecture of a simple expert system.

The explanation facility allows a user to understand how the expert system arrived at certain results. The overall purpose of the knowledge acquisition facility is to provide a convenient and efficient means for capturing and storing all components of the knowledge base.

Very often specialized user interface software is used for designing, updating, and using expert systems. The purpose of the user interface is to ease use of the expert system for developers, users, and administrators.

3 INFERENCE ENGINE IN RULE-BASED SYSTEMS

A rule-based system consists of *if-then rules*, a bunch of *facts*, and an *interpreter* controlling the application of the rules, given the facts.

These *if-then* rule statements are used to formulate the conditional statements that comprise the complete knowledge base. A single *if-then* rule assumes the form 'if x is A then y is B' and the if-part of the rule 'x is A' is called the *antecedent* or *premise*, while the then-part of the rule 'y is B' is called the *consequent* or *conclusion*. There are two broad kinds of inference engines used in rule-based systems: *forward chaining* and *backward chaining* systems.

In a *forward chaining* system, the initial facts are processed first, and keep using the rules to draw new conclusions given those facts. In a *backward chaining* system, the hypothesis (or solution/goal) we are trying to reach is processed first, and keep looking for rules that would allow to conclude that hypothesis. As the processing progresses, new subgoals are also set for validation. Forward chaining systems are primarily data-driven, while backward chaining systems are goal-driven. Consider an example with the following set of *if-then* rules

Rule 1: *If A and C then Y*
Rule 2: *If A and X then Z*
Rule 3: *If B then X*
Rule 4: *If Z then D*

If the task is to prove that D is true, given A and B are true. According to *forward chaining*, start with Rule 1 and go on downward till a rule that fires is found. Rule 3 is the only one that fires in the first iteration. After the first iteration, it can be concluded that A, B, and X are true. The second iteration uses this valuable information. After the second iteration, Rule 2 fires adding Z is true, which in turn helps Rule 4 to fire, proving that D is true. Forward chaining strategy is especially appropriate in situations where data are expensive to collect, but few in quantity. However, special care is to be taken when these rules are constructed, with the preconditions specifying as precisely as possible when different rules should fire.

In the backward chaining method, processing starts with the desired goal, and then attempts to find evidence for proving the goal. Returning to the same example, the task to prove that D is true would be initiated by first finding a rule that proves D. Rule 4 does so, which also provides a subgoal to prove that Z is true. Now Rule 2 comes into play, and as it is already known that A is true, the new subgoal is to show that X is true. Rule 3 provides the next subgoal of proving that B is true. But that B is true is one of the given assertions. Therefore, it could be concluded that X is true, which implies that Z is true, which in turn also implies that D is true. Backward chaining is useful in situations where the quantity of data is potentially very large and where some specific characteristic of the system under consideration is of interest. If there is not much knowledge what the conclusion might be, or there is some specific hypothesis to test, forward chaining systems may be inefficient. In principle, we can use the same set of rules for both forward and backward chaining. In the case of backward chaining, since the main concern is with matching the conclusion of a rule against some goal that is to be proved, the 'then' (consequent) part of the rule is usually not expressed as an action to take but merely as a state, which will be true if the antecedent part(s) are true (Donald, 1986).

4 EXPERT SYSTEM DEVELOPMENT

Steps in the expert systems development process include determining the actual requirements, knowledge acquisition, constructing expert system components, implementing results, and formulating a procedure for maintenance and review.

Knowledge acquisition is the most important element in the development of expert system (Niwa, Sasaki and Ihara, 1988). Knowledge could be obtained by interviewing domain experts and/or learning by experience.

Very often people express knowledge as natural language (spoken language), or using letters or symbolic terms. There exist several methods to extract human knowledge. Cognitive Work Analysis (CWA) and the Cognitive Task Analysis (CTA) provide frameworks to extract knowledge.

The CWA is a technique to analyze, design, and evaluate the human computer interactive systems (Vicente, 1999).

The CTA is a method to identify cognitive skill, mental demands, and needs to perform task proficiency (Militallo and Hutton, 1998). This focuses on describing the representation of the cognitive elements that defines goal generation and decision-making. It is a reliable method for extracting human knowledge because it is based on the observations or an interview.

Most expert systems are developed using specialized software tools called *shells*. These shells come equipped with an inference mechanism (backward chaining, forward chaining, or both), and require knowledge to be entered according to a specified format.

One of the most popular shells widely used throughout the government, industry, and academia is the CLIPS (CLIPS, 2004). CLIPS is an expert system tool that provides a complete environment for the construction of rule- and/or object-based expert systems. CLIPS provides a cohesive tool for handling a wide variety of knowledge with support for three different programming paradigms: rule-based, object-oriented, and procedural. CLIPS is written in C for portability and speed and has been installed on many different operating systems without code changes.

5 FUZZY EXPERT SYSTEMS

The world of information is surrounded by uncertainty and imprecision. The human reasoning process can handle inexact, uncertain, and vague concepts in an appropriate manner. Usually, the human thinking, reasoning, and perception process cannot be expressed precisely. These types of experiences can rarely be expressed or measured using statistical or probability theory. Fuzzy logic provides a framework to model uncertainty, the human way of thinking, reasoning, and the perception process. Fuzzy systems were first introduced by Zadeh (1965).

A fuzzy expert system is simply an expert system that uses a collection of fuzzy membership functions and rules, instead of Boolean logic, to reason about data (Schneider *et al.,* 1996). The rules in a fuzzy expert system are usually of a form similar to the following:
If A is *low* and B is *high* then $X = medium$
where A and B are input variables, X is an output variable. Here low, high, and medium are fuzzy sets defined on A, B, and X respectively. The antecedent (the rule's premise) describes to what degree the rule applies, while the rule's consequent assigns a membership function to each of one or more output variables.

Let X be a space of objects and x be a generic element of X. A classical set A, $A \subseteq X$, is defined as a collection of elements or objects $x \in X$, such that x can either belong or not belong to the set A. A fuzzy set A in X is defined as a set of ordered pairs

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{1}$$

where $\mu_A(x)$ is called the *membership function* (MF) for the fuzzy set A. The MF maps each element of X to a membership grade (or membership value) between zero and one. Obviously (1) is a simple extension of the definition of a classical set in which the characteristic function is permitted to have any values between zero and one.

The intersection of two fuzzy sets A and B is specified in general by a function T: [0,1] × [0,1] → [0,1], which aggregates two membership grades as follows:

$$\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x)\bar{*}\mu_B(x) \tag{2}$$

where $\bar{*}$ is a binary operator for the function T. This class of fuzzy intersection operators are usually referred to as *T-norm operators* (Jang, Sun and Mizutani, 1997). Four of the most frequently used T-norm operators are

$$\text{Minimum: } T_{\text{min}}(a, b) = \min(a, b) = a \wedge b \tag{3}$$

$$\text{Algebraic product: } T_{\text{ap}}(a, b) = ab \tag{4}$$

$$\text{Bounded product: } T_{\text{bp}}(a, b) = 0 \vee (a + b - 1) \tag{5}$$

$$\text{Drastic product: } T_{\text{dp}}(a, b) = \begin{cases} a, & \textit{if } b = 1 \\ b, & \textit{if } a = 1 \\ 0, & \textit{if } a, b < 1 \end{cases} \tag{6}$$

Like intersection, the fuzzy union operator is specified in general by a function S: [0,1] × [0,1] → [0,1], which aggregates two membership grades as follows:

$$\mu_{A\cup B}(x) = S(\mu_A(x), \mu_B(x)) = \mu_A(x) \mp \mu_B(x) \tag{7}$$

where $\mp$ is the binary operator for the function S. This class of fuzzy union operators are often referred to as *T-conorm* (or *S-norm*) operators (Jang, Sun and Mizutani, 1997). Four of the most frequently used T-conorm operators are

$$\text{Maximum: } S_{\text{max}}(a, b) = \max(a, b) = a \vee b \tag{8}$$

$$\text{Algebraic sum: } S_{\text{as}}(a, b) = a + b - ab \tag{9}$$

$$\text{Bounded sum: } S_{\text{bs}}(a, b) = 1 \wedge (a + b) \tag{10}$$

$$\text{Drastic sum: } S_{\text{ds}}(a, b) = \begin{cases} a, & \textit{if } b = 0 \\ b, & \textit{if } a = 0 \\ 1, & \textit{if} a, b > 0 \end{cases} \tag{11}$$

Both the intersection and union operators retain some properties of the classical set operation. In particular, they are associative and commutative.

Figure 2 illustrates the basic architecture of a fuzzy expert system. The main components are a fuzzification interface, a fuzzy rule base (knowledge base), an inference engine (decision-making logic), and a defuzzification interface. The input variables are fuzzified whereby the membership functions defined on the input variables are applied to their actual values, to determine the degree of truth for each rule antecedent. Fuzzy *if-then* rules and fuzzy reasoning are the backbone of fuzzy expert systems, which are the most

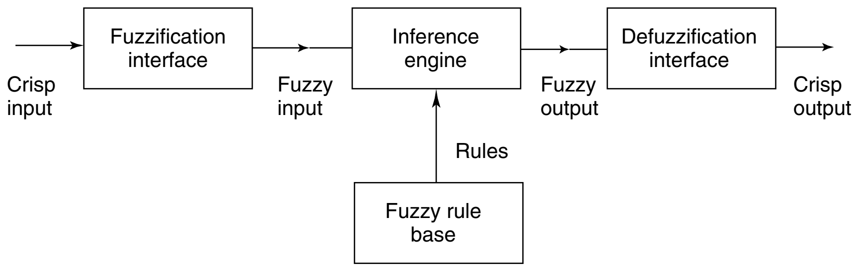


Figure 2. Basic architecture of a fuzzy expert system.

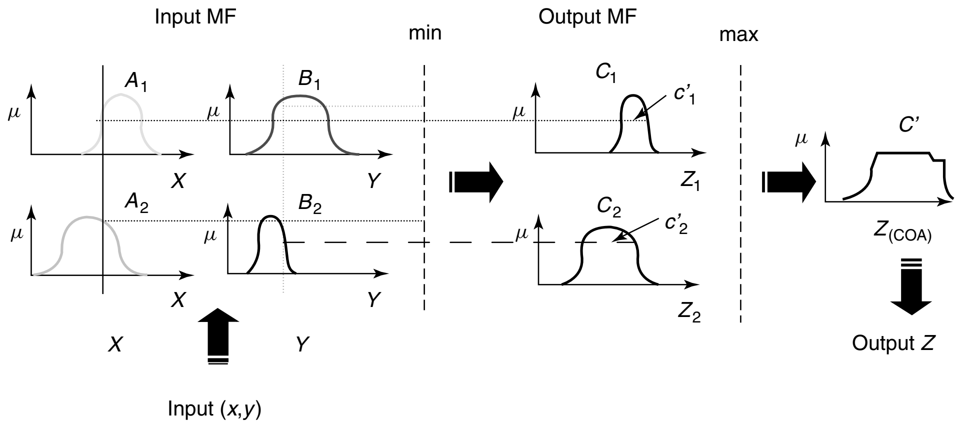


Figure 3. Mamdani fuzzy inference system using min and max for T-norm and T-conorm operators.

important modeling tools based on fuzzy set theory. The fuzzy rule base is characterized in the form of *if-then* rules in which the antecedents and consequents involve linguistic variables. The collection of these fuzzy rules forms the rule base for the fuzzy logic system. Using suitable inference procedure, the truth value for the antecedent of each rule is computed, and applied to the consequent part of each rule. This results in one fuzzy subset to be assigned to each output variable for each rule. Again, by using suitable composition procedure, all the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. Finally, defuzzification is applied to convert the fuzzy output set to a crisp output.

The basic fuzzy inference system can take either fuzzy inputs or crisp inputs, but the outputs it produces are always fuzzy sets. The defuzzification task extracts the crisp output that best represents the fuzzy set. With crisp inputs and outputs, a fuzzy inference system implements a nonlinear mapping from its input space to output space through a number of fuzzy *if-then* rules.

In what follows, the two most popular fuzzy inference systems are introduced that have been widely deployed in various applications. The differences between these two fuzzy inference systems lie in the consequents of their fuzzy rules, and thus their aggregation and defuzzification procedures differ accordingly.

According to Mamdani, fuzzy inference system (Mamdani and Assilian, 1975) – see Figure 3 – the rule antecedents and consequents are defined by fuzzy sets and has the following structure:

$$\textit{if } x \textit{ is } A_1 \textit{ and } y \textit{ is } B_1 \textit{ then } z_1 = C_1' \qquad (12)$$

There are several defuzzification techniques. The most widely used defuzzification technique uses the centroid of area method as follows

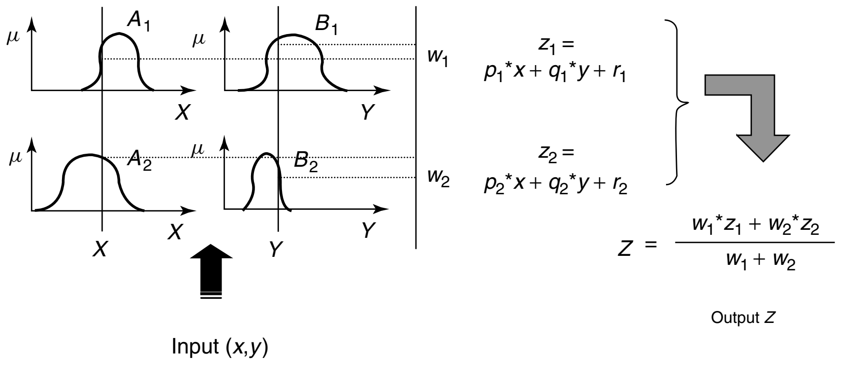
$$\text{Centroid of area } Z_{\text{COA}} = \frac{\int_Z \mu_A(z)\, z\, \mathrm{d}z}{\int_Z \mu_A(z)\, \mathrm{d}z} \qquad (13)$$

where $\mu_A(z)$ is the aggregated output MF.

Takagi and Sugeno (1985) proposed an inference scheme in which the conclusion of a fuzzy rule is constituted by a weighted linear combination of the crisp inputs rather than a fuzzy set. A basic Takagi–Sugeno fuzzy inference system is illustrated in Figure 4 and the rule has the following structure

$$\textit{if } x \textit{ is } A_1 \textit{ and } y \textit{ is } B_1, \textit{ then } z_1 = p_1x + q_1y + r_1 \qquad (14)$$

where p_1, q_1, and r_1 are linear parameters. TSK Takagi–Sugeno Kang fuzzy controller usually needs a smaller



μ
A_1
B_1
w_1
$z_1 = p_1{*}x + q_1{*}y + r_1$
X
Y
A_2
B_2
w_2
$z_2 = p_2{*}x + q_2{*}y + r_2$
X
Y
$Z = \frac{w_1{*}z_1 + w_2{*}z_2}{w_1 + w_2}$
Output Z
Input (x,y)

Figure 4. Takagi–Sugeno fuzzy inference system using a min or product as T-norm operator.

number of rules, because their output is already a linear function of the inputs rather than a constant fuzzy set.

6 MODELING FUZZY EXPERT SYSTEMS

Fuzzy expert system modeling can be pursued using the following steps.

- Select relevant input and output variables. Determine the number of linguistic terms associated with each input/output variable. Also, choose the appropriate family of membership functions, fuzzy operators, reasoning mechanism, and so on.
- Choose a specific type of fuzzy inference system (for example, Mamdani, Takagi–Sugeno etc.). In most cases, the inference of the fuzzy rules is carried out using the 'min' and 'max' operators for fuzzy intersection and union.
- Design a collection of fuzzy *if-then* rules (knowledge base). To formulate the initial rule base, the input space is divided into multidimensional partitions and then actions are assigned to each of the partitions.

In most applications, the partitioning is achieved using one-dimensional membership functions using fuzzy *if-then* rules as illustrated in Figure 5. The consequent parts of the rule represent the actions associated with each partition. It is evident that the MFs and the number of rules are tightly related to the partitioning.

7 ILLUSTRATION OF FUZZY EXPERT SYSTEM DESIGN

This section illustrates the development of a reactive power prediction model using Mamdani and Takagi–Sugeno fuzzy inference expert systems. The MatLab™ fuzzy logic toolbox was used to simulate the various experiments (Fuzzy Logic Tool Box, 2004).

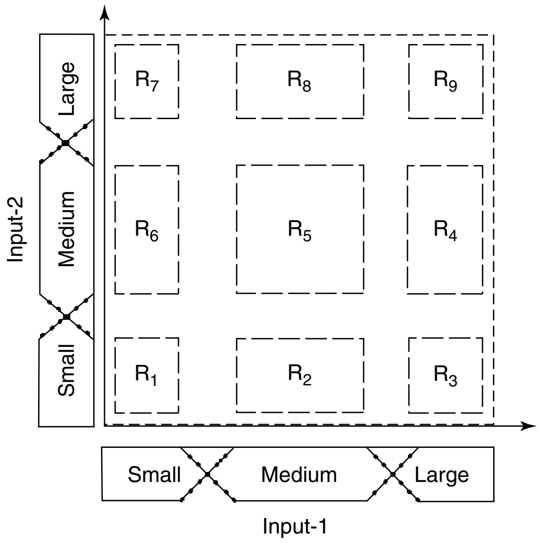


Figure 5. Example showing how the two-dimensional spaces are partitioned using three trapezoidal membership functions per input dimension. A simple if-then rule will appear as *If input-1 is medium and input 2 is large, then rule* R_8 *is fired*.

The task is to develop a fuzzy expert system to forecast the reactive power (P) at time $t + 1$ by knowing the load current (I) and voltage (V) at time t. The experiment system consists of two stages: developing the fuzzy expert system, and performance evaluation using the test data. The model has two in–out variables (V and I) and one output variable (P). Training and testing data sets were extracted randomly from the master dataset. Sixty percent of data was used for training and the remaining 40% for testing (Abraham and Khan, 2003).

7.1 Design and experiments: fuzzy expert systems

First, the effects of (a) shape and quantity of membership functions (b) T-norm and T-conorm operators (c) defuzzification methods and (d) inference method for designing the fuzzy expert system is analyzed. Experiments were carried out using four different settings using the same rule base.

Experiment 1 (To evaluate the effect on the number of membership functions) The following settings were used for designing the expert system

1. Two triangular membership functions (MFs) for each input variable and four triangular MFs for the output variable (power). Using the grid partitioning method (Figure 5), four *if-then* rules were developed.
2. Three triangular MFs for each input variable and nine triangular MFs for the output variable (power). The rule base consisted of nine *if-then* rules.

'min' and 'max' were used as T-norm and T-conorm operators and the centroid method of defuzzification for Mamdani inference method and weighted average defuzzification method for Takagi–Sugeno Fuzzy Inference System (FIS). The developed fuzzy inference systems using Mamdani and Takagi–Sugeno models are depicted in Figures 6 to 9. Table 1 summarizes the training and testing Root Mean Squared Error (RMSE) values.

Experiment 2 (To evaluate the effect of shape of membership functions) For the Mamdani FIS, three Gaussian MFs for each input variable and nine Gaussian MFs for the output variable were used. The rule base consisted of nine *if-then* rules. 'min' and 'max' as T-norm and T-conorm operators, and the centroid method of defuzzification for Mamdani FIS and the weighted average defuzzification method for Takagi–Sugeno FIS were also used. The developed fuzzy inference systems using Mamdani and Takagi–Sugeno models are depicted in Figures 10 and 11. Table 2 summarizes the training and testing RMSE values.

Experiment 3 (To evaluate the effect of fuzzy operators) For Mamdani FIS, three Gaussian MFs for each input variable and nine Gaussian MFs for the output variable were used. The rule base consisted of nine *if-then* rules. T-norm and T-conorm operators were 'product' and 'sum' and the centroid method of defuzzification for Mamdani FIS, and weighted average defuzzification method for Takagi–Sugeno FIS were used. Table 3 summarizes the training and testing RMSE values.

Experiment 4 (To evaluate the effect of defuzzification operators) For the Mamdani FIS, three Gaussian MFs for each input variable and nine Gaussian MFs for the output variable were used. The rule base consisted of nine *if-then* rules. T-norm and T-conorm operators were 'product' and 'sum' and the following defuzzification operators were tested for Mamdani FIS.

Table 1. Empirical comparison of fuzzy inference systems and quantity of Membership Functions (MFs).

No. of MFs	Mamdani FIS		Takagi – Sugeno FIS	
	Root mean squared error			
	Training	Test	Training	Test
2	0.401	0.397	0.024	0.023
3	0.348	0.334	0.017	0.016

Table 2. Empirical comparison of fuzzy inference systems using Gaussian MFs.

Mamdani FIS		Takagi – Sugeno FIS	
Root mean squared error			
Training	Test	Training	Test
0.243	0.240	0.021	0.019

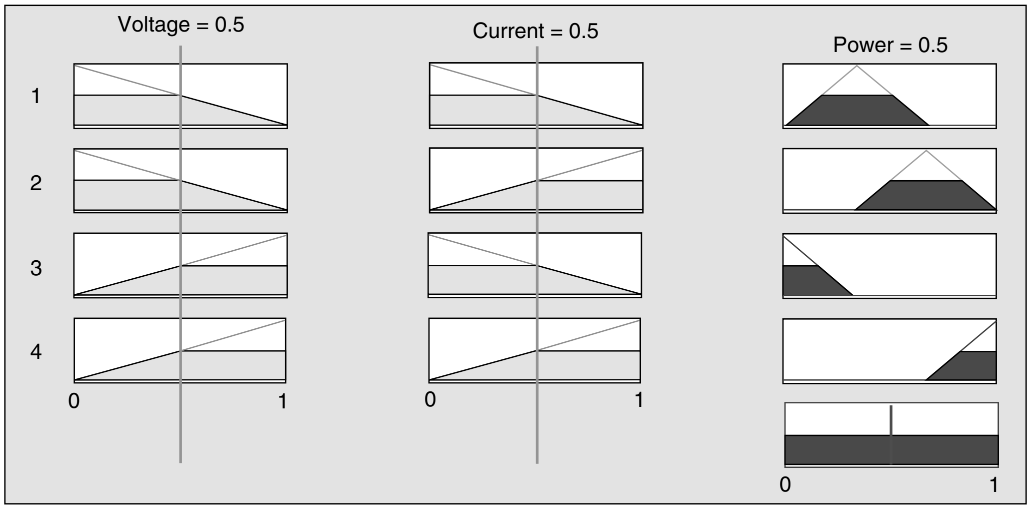


Figure 6. Mamdani fuzzy inference system using two triangular MFs for input variables.

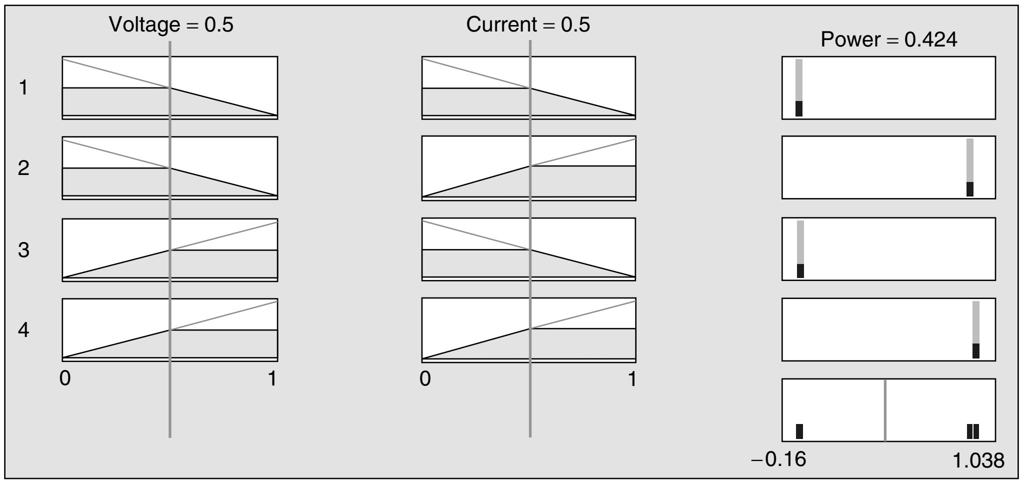


Figure 7. Takagi–Sugeno fuzzy inference system using two triangular MFs for input variables.

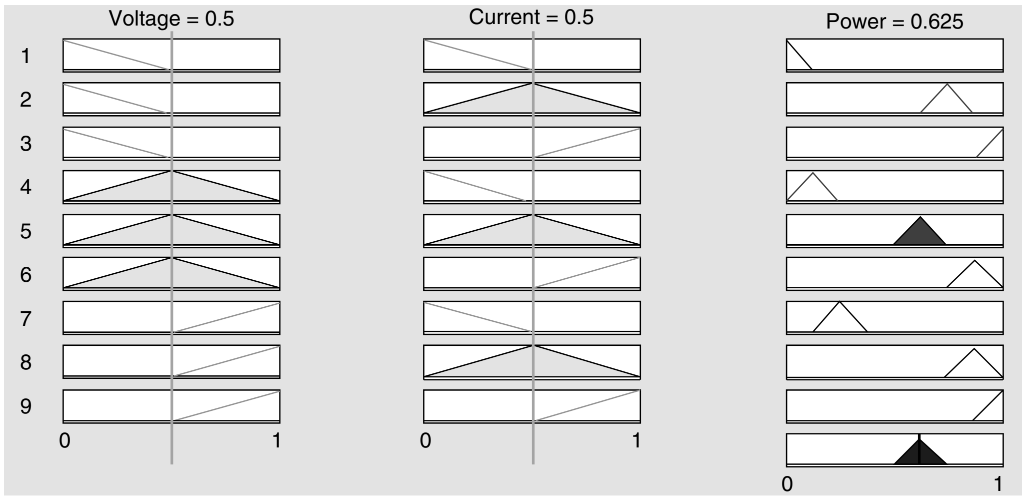


Figure 8. Mamdani fuzzy inference system using three triangular MFs for input variables.

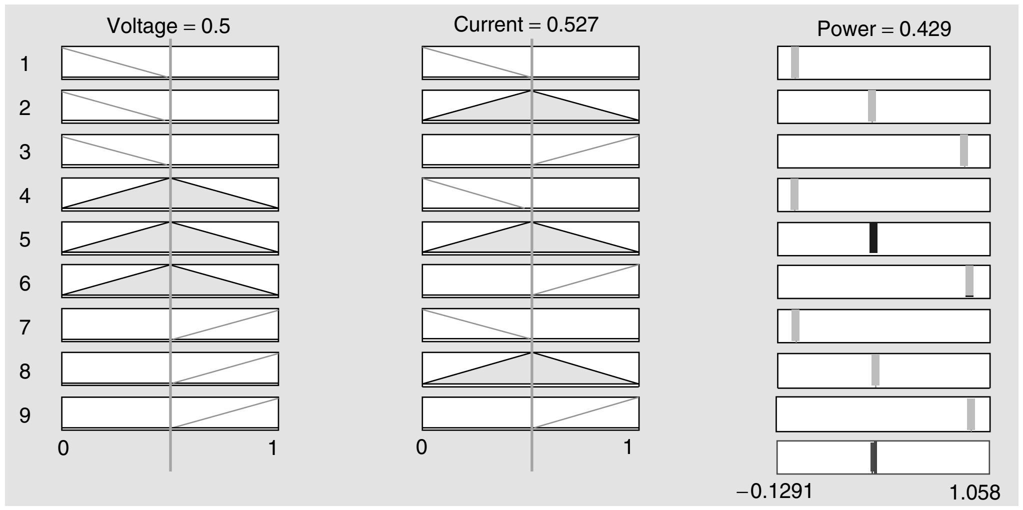


Figure 9. Takagi–Sugeno fuzzy inference system using three triangular MFs for input variables.

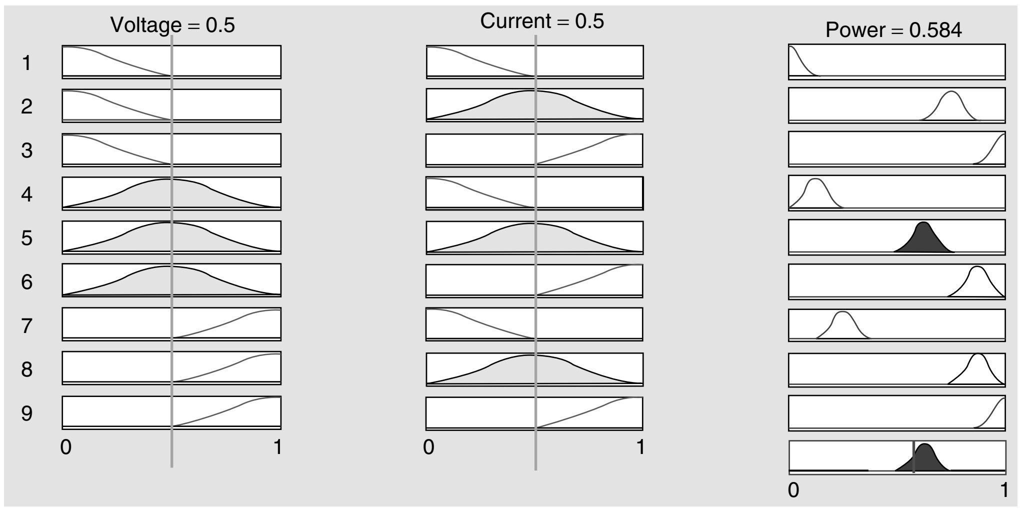


Figure 10. Mamdani fuzzy inference system using three Gaussian MFs for input variables.

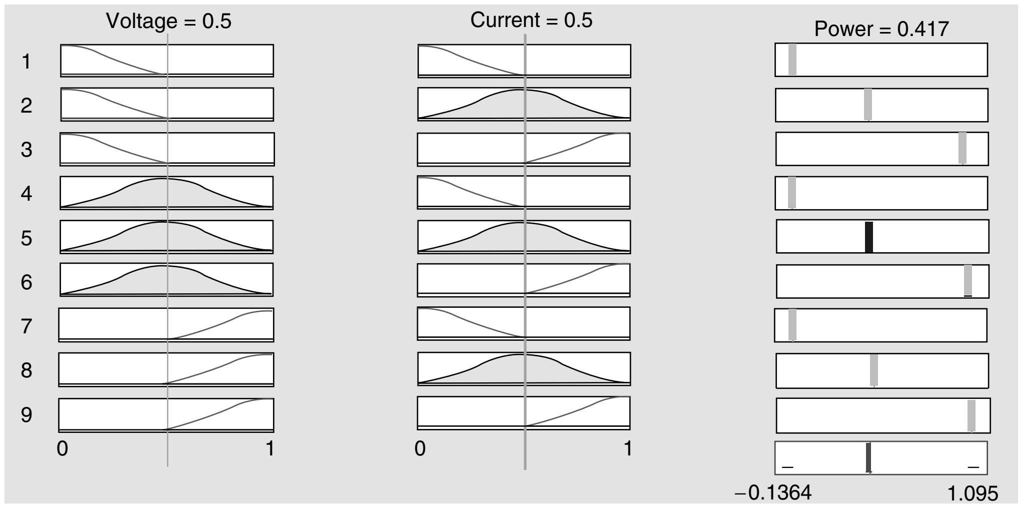


Figure 11. Takagi–Sugeno fuzzy inference system using three Gaussian MFs for input variables.

Table 3. Empirical comparison of fuzzy inference systems for different fuzzy operators.

Mamdani FIS		Takagi – Sugeno FIS	
Root mean squared error			
Training	Test	Training	Test
0.221	0.219	0.019	0.018

- Centroid
- Bisector of Area (BOA)
- Mean of Maximum (MOM)
- Smallest of Maximum (SOM).

For the Takagi–Sugeno FIS, the weighted sum and weighted average defuzzification methods were used. Table 4 summarizes the training and testing of RMSE values.

Discussions of Results and Problem Solution As depicted in Table 1, when the number of input MFs were increased from two to three, the RMSE values reduced regardless of the inference system used. However, when the shape of the MF was changed to Gaussian, RMSE values for Mamdani FIS decreased but the RMSE values for Takagi–Sugeno FIS increased (Table 2). Using Gaussian MFs, when the T-norm and T-conorm operators were changed to ‘product’ and ‘sum’ (instead of ‘min’ and ‘max’) both the inference methods performed better (Table 3). Finally, the selection of an ideal defuzzification operator also has a direct influence in the performance of FIS as shown in Table 4.

Table 4. Empirical comparison of fuzzy inference systems for different defuzzification operators.

Mamdani FIS			Takagi–Sugeno FIS		
Defuzzification	RMSE		Defuzzification	RMSE	
	Training	Test		Training	Test
Centroid	0.221	0.0219	Weighted sum	0.019	0.018
MOM	0.230	0.232	Weighted average	0.085	0.084
BOA	0.218	0.216			
SOM	0.229	0.232			

The design of the rule base (number of rules and how the inputs and outputs are related) is also very important for the good performance of FIS. The role of weighting factors emphasizing the importance of certain rules also bears a prominent role for the overall performance. When the input/output dimensions become larger, manual design becomes tedious and sometimes could even lead to poor design and implementation.

8 ADAPTATION OF FUZZY INFERENCE SYSTEMS

Expert knowledge is often the main source to design the fuzzy expert systems. Figure 12 illustrates the various parameters and components that need to be adapted for controlling a process. According to the performance measure of the problem environment, the membership functions, rule bases, and the inference mechanism are to be adapted (Abraham, 2002).

Neural network learning, self-organizing maps and clustering methods could be used to generate rules. Gradient descent and its variants could be applied to fine-tune the parameters of parameterized input/output membership functions and fuzzy operators (Abraham, 2001). Adaptation of fuzzy inference systems using evolutionary computation techniques has been widely explored. Evolutionary Computation (EC) is a population based adaptive method, which may be used to solve optimization problems, based on the genetic processes of biological organisms (Michalewicz and Fogel, 1999).

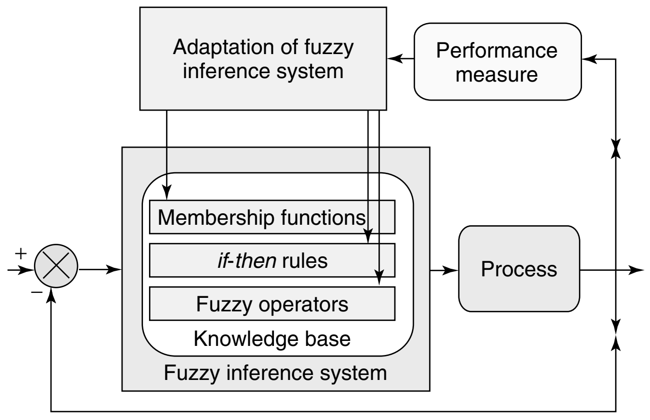


Figure 12. Adaptation of fuzzy inference systems.

Over many generations, natural populations evolve according to the principles of natural selection and 'survival of the fittest', first clearly stated by Charles Darwin in '*On the Origin of Species*'. By mimicking this process, EC could 'evolve' solutions to real-world problems, if they have been suitably encoded (problem representation is called *chromosome*). Automatic adaptation of membership functions is popularly known as *self tuning* and the chromosome encodes parameters of trapezoidal, triangle, logistic, hyperbolic-tangent, Gaussian membership functions, and so on. Evolutionary search of fuzzy rules can be carried out using three approaches. In the first method (Michigan approach), the fuzzy knowledge base is adapted as a result of antagonistic roles of competition and cooperation of fuzzy rules.

The second method (Pittsburgh approach), evolves a population of knowledge bases rather than individual fuzzy rules. Reproduction operators serve to provide a new combination of rules and new rules.

The third method (iterative rule learning approach), is very much similar to the first method with each chromosome representing a single rule, but contrary to the Michigan approach, only the best individual is considered to form part of the solution, discarding the remaining chromosomes of the population. The evolutionary learning process builds up the complete rule base through an iterative learning process (Cordón *et al.*, 2001).

9 SUMMARY

Rule-based expert systems have been applied in a vast number of application areas. An important advantage of the fuzzy expert system is that the knowledge is expressed as easy-to-understand linguistic rules. If we have data, the fuzzy expert system can be taught using neural network learning, EC, or other adaptation techniques. It is to be expected that the number of applications will grow considerably in the future now that success is clearly proven for these methods.

REFERENCES

Abraham, A. (2001) *Neuro-Fuzzy Systems: State-of-the-Art Modeling Techniques, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, in Lecture Notes in Computer Science, Vol. 2084, (eds. Mira., Jose and Prieto., Alberto) Springer Verlag, Germany (pp. 269–276).

Abraham, A. (2002) Intelligent Systems: Architectures and Perspectives, Recent Advances in Intelligent Paradigms and Applications, in *Studies in Fuzziness and Soft Computing*, Chapter 1, (eds A., Abraham, L., Jain and J., Kacprzyk), Springer Verlag, Germany (pp. 1–35).

Abraham, A. and Khan, M.R. (2003) Neuro-Fuzzy Paradigms for Intelligent Energy Management, Innovations in Intelligent Systems: Design, Management and Applications, in *Studies in Fuzziness and Soft Computing*, Chapter 12, (eds A., Abraham, L., Jain and B., Jan van der Zwaag), Springer Verlag, Germany (pp. 285–314).

CLIPS (2004) *Expert System Shell* <http://www.ghg.net/clips/CLIPS.html>.

Cordón, O., Herrera, F., Hoffmann, F. and Magdalena, L. (2001) *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, World Scientific Publishing Company, Singapore.

Donald, W.A. (1986) *A Guide to Expert Systems*, Addison-Wesley, Boston, MA.

Fuzzy Logic Toolbox, *The MathWorks* (2004) http://www.mathworks.com/products/fuzzylogic/.

Giarratano, J. and Riley, G. (1989) *Expert Systems: Principles and Programming*, PWS-Kent Publishing Co, Boston, MA.

Ignizio, J.P. (1991) *Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems*, McGraw-Hill, Inc, USA.

Jang, J.S.R., Sun, C.T. and Mizutani, E. (1997) *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall Inc, USA.

Mamdani, E.H. and Assilian, S. (1975) An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies*, **7**(1), 1–13.

Michalewicz, Z. and Fogel, D.B. (1999) *How to Solve It: Modern Heuristics*, Springer Verlag, Germany.

Militallo, L.G., Hutton, R.J.B. (1998) Applied Cognitive Task Analysis (ACTA): A Practitioner's Toolkit for Understanding Cognitive. *Ergonomics*, **41**(11), 1618–1642.

Niwa, K., Sasaki, K. and Ihara, H. (1988) An Experimental Comparison of Knowledge Representation Schemes, in *Principles of Expert Systems*, (Eds A., Gupta and E.B., Prasad), IEEE Press, New York (pp. 133–140).

Schneider, M., Langholz, G., Kandel, A. and Chew, G. (1996) *Fuzzy Expert System Tools*, John Wiley & Sons, USA.

Takagi, T. and Sugeno, M. (1985) Fuzzy identification of systems and its applications of modeling and control, *IEEE Transactions of Systems. Man and Cybernetics*, USA (pp. 116–132).

Vicente, K.J. (1999) *Cognitive Work Analysis:∼ Towards Safe, Productive, and Healthy Computer-Based Work*, Lawrence Erlbaum Associates, Inc. Press, USA.

Zadeh, L.A. (1965) Fuzzy Sets. *Information and Control*, **8**, 338–353.