Time complexity analysis of genetic- fuzzy system for disease diagnosis.

Ephzibah.E.P.

School of Information Technology and Engineering,

VIT University,Vellore, Tamilnadu, India.

ep.ephzibah@vit.ac.in

ABSTRACT

A new generation of tools and techniques are needed for finding interesting patterns in the data and discovering useful knowledge. Especially, Medical knowledge consists of a combination of structural information about known biological facts and probabilistic or actuarial information about exposures to hazards and recovery rates. Probabilistic information is especially difficult to use, as it requires constant maintenance and it usually comes in the form of study results which are not ideally suited for making individual predictions. Patterns summarizing mutual associations between class decisions and attribute values in a pre-classified database provide insight into the significance of attributes and are also useful in classificatory knowledge. The proposed work is an efficient method to extract significant attributes from a database. Reducing the features or attributes enhances the quality of knowledge extracted and also the speed of computation. In this paper the design of a hybrid algorithm for heart disease diagnosis using effective and efficient genetic algorithm and fuzzy logic is implemented. The proposed work analyses the time complexity of genetic- fuzzy system.

KEYWORDS

Genetic Algorithms, Fuzzy Logic, Disease diagnosis, Time complexity.

1. INTRODUCTION

Fuzzy logic is viewed as an extension of classical logical systems that provides an effective conceptual framework for dealing with the problem of knowledge representation in an environment of uncertainty and vagueness. It is a powerful tool for knowledge representation in computational intelligence. Among the most successful applications of fuzzy logic is the disease diagnosis. Genetic algorithms are search algorithms that implements principles inspired by natural population genetics to evolve solutions to problems. [1] A population of knowledge structure is maintained and each one represents a candidate solution and has an associated fitness to determine in the process of competition to produce the new ones. The new ones are created using the genetic operators like crossover and mutation. Genetic algorithms have a grate measure of success in optimization problems. The method of getting the output from the genetic algorithm and sending it as input to the fuzzy system is done in this work.

The description about the genetic algorithms and fuzzy logic are given in section 2 and 3. The section 4 gives an overview of the algorithms involved in this work. Section 5 gives the details about the experimental results obtained using the proposed method. The conclusion section 6 summarizes the work.

DOI : 10.5121/acij.2011.2403

2. RELATED WORK

In the past few years, there were many problems solved using Genetic algorithms, but the related theoretical study on the time complexity of these algorithms is relatively few.Thierens, Goldberg and Pereira (1998) analyzed the time complexity of genetic algorithms on exponentially scaled problems. For exponentially scaled problems the time complexity of Genetic algorithms is a quadratic function. According to Hariek et.al (1999), the linear time complexity for uniformly scaled problems grows with √m for $O(m)$ and $O(m^2)$. He and Yao (2002) showed how a population can be beneficial over single individual in genetic algorithms and the results proved that a well chosen population size and crossover and mutation operations can provide a better performance in Time complexity. In the case of fuzzy zyztems the time complexity is calculated based on the number of mathematical and move operations performed. Yong Ho Kim et. al (2000) investigated the computation time of fuzzy logic systems and found it to be more performance oriented when compared to other techniques.

3. GENETIC ALGORITHMS

Genetic algorithms are one of the stochastic search algorithms that attempt to contain the combinatorial explosion problem at the price of completeness. Genetic algorithms are special from other stochastic search algorithms in which it maintains several solutions in parallel. In addition to progressing space to exploitation, genetic algorithms offer exploration of the search space, large and undetermined jumps in the search space. Exploration is achieved by combining elements from two existing solutions to find the third one, which may represent a new point in the search space. Genetic algorithms are able to mine the search space in several locations at the same time. This gives robustness to the algorithm. Robustness is defined as the consistency in finding optimal solutions. Genetic algorithms maintain a population pool of candidate solutions called strings or chromosomes. Each chromosome p is a collection of building blocks known as genes, which are instantiated with values from a finite domain. Associated with each chromosome is a fitness value which is determined by a user defined function, called the fitness function. The function returns a magnitude that is proportional to the candidate solution's suitability and optimality. Fig. 1 shows the control and data flow of GA. At the start of the algorithm, an initial population is generated. Initial members of the population may be randomly generated, or generated according to some rules. The reproduction operator selects chromosomes from the population to be parents for a new chromosome and enters them into the mating pool. Selection of a chromosome for parenthood can range from a totally random process to one that is biased by the chromosome's fitness. The cross-over operator oversees the mating process of two chromosomes. A child chromosome that is created inherits complementing genetic material from its parents. The new chromosome produced is entered into the offspring pool. The mutation operator takes the chromosome and randomly changes its contents. The process of reproduction, cross-over mutation and formation of a new population completes one generation cycle.[2]

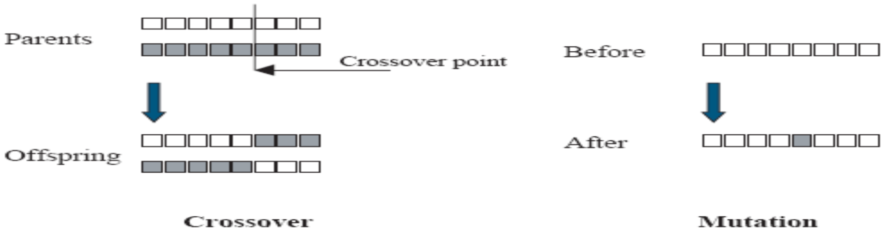


Fig 1. Cross over and mutation operations of Genetic Algorithms.

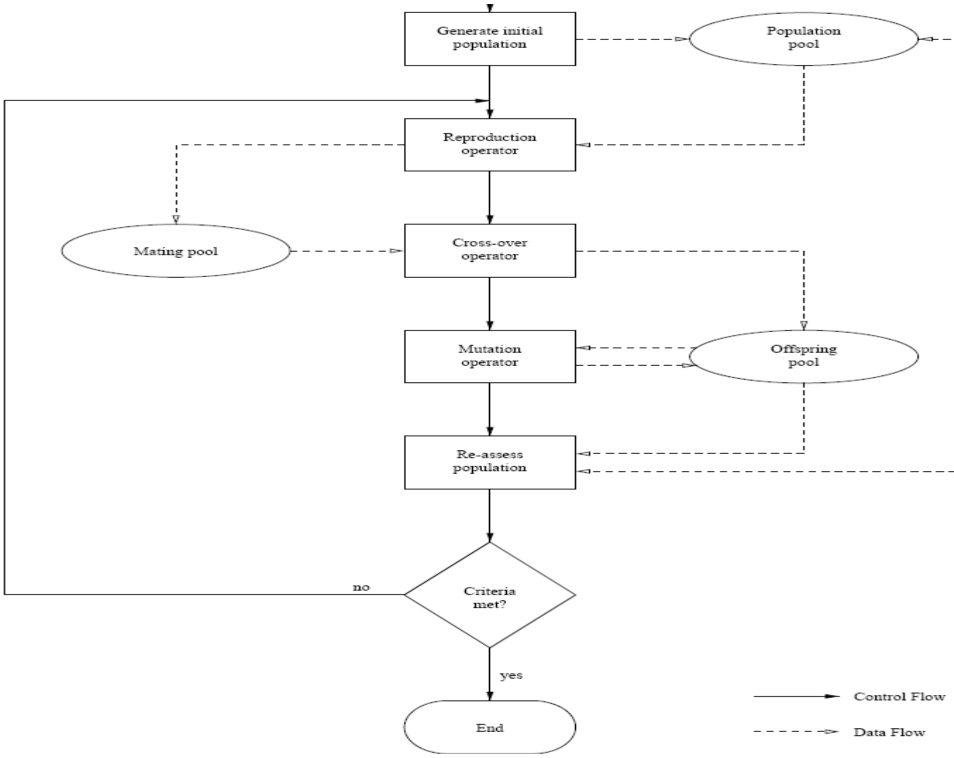


Fig1. Control and Data flow of Genetic Algorithms.

4. FUZZY LOGIC

Fuzzy logic poses the ability to mimic the human mind to effectively employ modes of reasoning that are approximate rather than exact. It can model nonlinear functions of arbitrary complexity to a desired degree of accuracy. It has the property of relativity and variability. Fuzzy logic allows decision making with estimated values under incomplete or uncertain information. **F**uzzy set: Let X be a nonempty set. A fuzzy set A in X is characterized by its membership function μ_A: X → [0, 1] and μ_A (x) is interpreted as the degree of membership of element x in fuzzy set A for each x ϵ X. It is clear that *A* is completely determined by the set of tuples $A = \{(u, \mu_A (u))|u \in X\}$.

The family of all fuzzy sets in *X* is denoted by *F*(*X*). If $X = \{x_1, . . . , x_n\}$ is a finite set and *A* is a fuzzy set in *X* then we often use the notation $A = \mu_1 / x_1 + \cdot \cdot \cdot + \mu_n / x_n$
where the term μ_i/x_i, $i = 1, . . . , n$ signifies that μ_i is the grade of membership of x_i in *A* and the plus sign represents the union. Every element in the universe of discourse is a member of the fuzzy set to some grade, may be even zero. The set of elements that have a non-zero membership is called the *support* of the fuzzy set. The function that ties a number to each element of the universe is called the *membership function μ(x).*

The knowledge base encodes the expert knowledge by means of a set of fuzzy control rules. A fuzzy control rule is a conditional statement with a form IF THEN in which the antecedent is a condition in its application domain, the consequent is a control action to be applied. The knowledge is composed of database and rule base.
An example of a fuzzy 'if–then' rule is
e.g. If food is 'Delicious' and *service* is 'good',

then *tip* is ‘more’.

Such a typical fuzzy ‘if–then’ rule is mathematically represented as follows:

If x_1 is A_1 and x_2 is A_2 and x_m is A_m then y is *B*.

Each *x*i, in the antecedent (or premise) of such a fuzzy rule, represents a fuzzy input variable having a linguistic value *Ai* and, similarly, in the consequent (or conclusion) ‘*y* is *B*’, *y* is a fuzzy output value having linguistic value *B*. The qualitative fuzzy linguistic values associated to each variable are quantified using so called membership functions that partition the domain (or ‘universe of discourse’) of each input variable into several overlapping classes.

Fuzzy inference system is an integrated computing framework based on the concepts of fuzzy set theory. The rule base is the collection of fuzzy rules describing the system. The fuzzy inference system consists of the following four basic components as pictorially represented in fig 2.:

- Fuzzification module,
- A rule-base with if-then rules and related membership functions,
- An inference engine applying algorithms on the rule base and
- A defuzzification module.

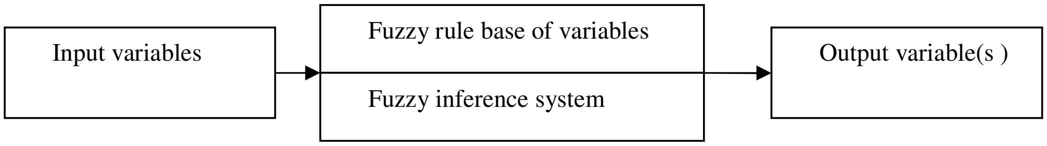


Fig 2: General architecture of a fuzzy inference system.

The general fuzzy inference process proceeds with the following steps:

1. The first step is called the FUZZIFICATION. In this step the membership functions defined on the input variables are applied to their actual values, to determine the degree of truth for each rule premise.

2. The second step is called the INFERENCE. The truth value for the premise of each rule is computed in this step and applied to the conclusion part of each rule. This results in one fuzzy subset to be assigned to each output variable for each rule. Usually minimum or the product is used as inference rules. In the case of minimum, the output membership function is clipped off at a height corresponding to the rule premise's computed degree of truth. While selected the product, the output membership function is scaled by the rule premise's computed degree of truth.

3. The third step is the COMPOSITION, in which all of the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable.

4. Finally DEFUZZIFICATION is performed to convert the fuzzy output set to a crisp number. There are more defuzzification methods available from which this work is done using the CENTROID method. This method gives the crisp value of the output variable that is computed by finding the variable value of the center of gravity of the membership function for the fuzzy value.

5. ALGORITHMIC VIEW

Genetic Algorithm for feature subset selection:

The gatool from MATLAB R2006b had been used for GA implementation. The selection operator is roulette wheel, and the crossover operator is double one-point crossover, and the mutation operator is binary mutation. The setting of GA parameters is very important. For GA parameters, if the population size is too small, it is difficult to get the best resolution and too big a population size will require a long convergence time. Thus, the size is normally 40–60. If the crossover *Pc* is too low, it is difficult to search forward and a *Pc* value too big will damage individuals with high adapting value. Therefore, the *Pc* is normally 0.3–0.9. If the mutation rate *Pm* is too low, the new individual is hard to produce and too high a *Pm* would make the GA simple search at random. Thus, the *Pm* is normally 0.01–0.2.[3]

The ga function in matalab:

```
x = ga(fitnessfcn,nvars)
x = ga(fitnessfcn,nvars,A,b)
x = ga(fitnessfcn,nvars,A,b,Aeq,beq)
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB)
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB,nonlcon)
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB,nonlcon,options)
x = ga(problem)
[x,fval] = ga(...)
[x,fval,exitflag] = ga(...)
[x,fval,exitflag,output] = ga(...)
[x,fval,exitflag,output,population] = ga(...)
[x,fval,exitflag,output,population,scores] = ga(...)
```

ga implements the genetic algorithm at the command line to minimize an objective function.

x = ga(fitnessfcn,nvars) finds a local unconstrained minimum, x, to the objective function, fitnessfcn. nvars is the dimension (number of design variables) of fitnessfcn. The objective function, fitnessfcn, accepts a vector x of size 1-by-nvars, and returns a scalar evaluated at x.

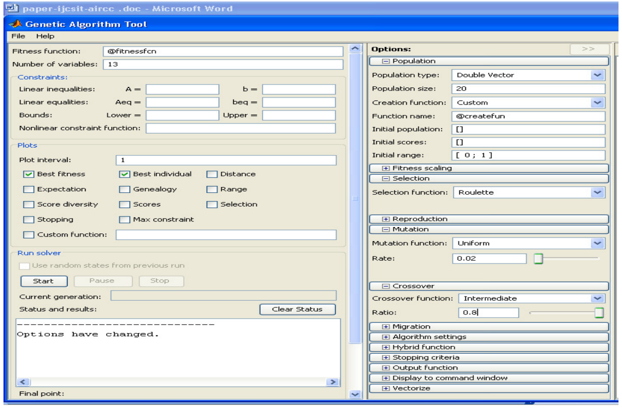
The program gatoolfile is used for the generation of feature subset. The fitness function “fitnessfcn” is assigned. The number of variables is 13 as there are 13 features available in the Heart disease dataset. The population size is 50. “selectionroulette” is invoked for the selection function. “crossoverintermediate” is invoked for the crossover function and “mutationgaussian” is invoked for the mutation function with values 0.8 and 0.05 respectively. The “createfun” is the function that initializes the population.

```
function [X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =  gatoolfile

fitnessFunction = @fitnessfcn;

%%Number of Variables
nvars = 13;
options = gaoptimset;
options = gaoptimset(options,'PopulationSize' ,50);
options = gaoptimset(options,'SelectionFcn' ,@selectionroulette);
options = gaoptimset(options,'CrossoverFcn' ,{  @crossoverintermediate 0.8 });
options = gaoptimset(options,'MutationFcn' ,{  @mutationgaussian 0.05 1  });
options = gaoptimset(options,'Display' ,'off');
options = gaoptimset(options,'OutputFcns' , @gaoutputfcn );
options = gaoptimset(options,'CreationFcn' ,@createfun);
options = gaoptimset(options,'Vectorized' ,'on');
%%Run GA
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES]=
ga(fitnessFunction,nvars,Aineq,Bineq,Aeq,Beq,LB,UB,nonlconFunction,options);
```

Fuzzy Classification system for effective diagnosis of the disease:

The designed system is based on the Cleveland clinic foundation dataset [13]. This dataset is a part of the collection of datasets at the university of California, Irvine collected by David Aha. The purpose of this dataset is to diagnose the presence or absence of heart disease given the results of various medical tests carried out on a patient. This database contains 76 attributes and 303 examples of patient, but all published experiments refer to using a subset of 14 of them and this work concentrates on only 6 of them. The feature subset of attributes obtained by the evaluation of genetic algorithms is given to the fuzzy system with a set of fuzzy inference rules and the diagnosis of the disease is done.

```
readfis('heartdisease.fis')
ans =
          name: 'heartdisease'
          type: 'mamdani'
          andMethod: 'min'
          orMethod: 'max'
          defuzzMethod: 'centroid'
           impMethod: 'min'
           aggMethod: 'max'
           input: [1x6 struct]
           output: [1x1 struct]
            rule: [1x24 struct]
```



Fitness function: @fitnessfcn
Number of variables: 13

Constraints:
Linear inequalities: A = b =
Linear equalities: Aeq = beq =
Bounds: Lower = Upper =
Nonlinear constraint function:

Plots
Plot interval: 1
☑ Best fitness ☑ Best individual ☐ Distance
☐ Expectation ☐ Genealogy ☐ Range
☐ Score diversity ☐ Scores ☐ Selection
☐ Stopping ☐ Max constraint
☐ Custom function:

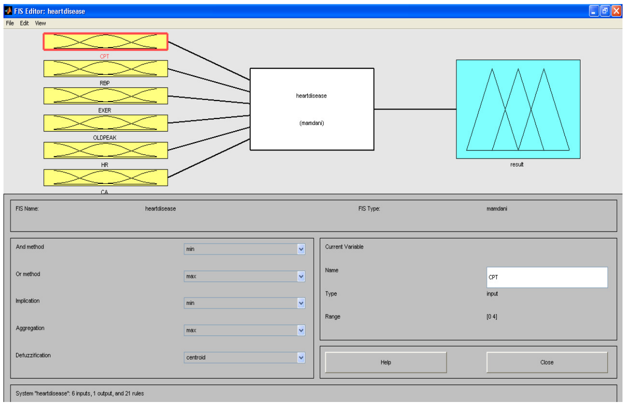
Run solver
☐ Use random states from previous run
Current generation:
Status and results:

```
------------------------------
Options have changed.
```

Final point:

Options:
⊟ Population
Population type: Double Vector
Population size: 20
Creation function: Custom
Function name: @createfun
Initial population: []
Initial scores: []
Initial range: [0 ; 1]
⊞ Fitness scaling
⊟ Selection
Selection function: Roulette
⊞ Reproduction
⊟ Mutation
Mutation function: Uniform
Rate: 0.02
⊟ Crossover
Crossover function: Intermediate
Ratio: 0.8
⊞ Migration
⊞ Algorithm settings
⊞ Hybrid function
⊞ Stopping criteria
⊞ Output function
⊞ Display to command window
⊞ Vectorize

Fig. 2 Genetic algorithm tool from Matlab for feature Selction.



FIS Name: heartdisease
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: CPT
Type: input
Range: [0 4]

System "heartdisease": 6 inputs, 1 output, and 21 rules

Fig 3.FIS Editor: Heart disease.

6.EXPERIMENTAL RESULTS

Time Complexity Analysis:

It is a part of the computational complexity theory that is used to describe an algorithm's use of computational resources. The big O notation is used to express the upper bound of the growth rate of a function and is used to describe asymptotic behavior. [7]

The big-*O* notation is described using set notation as follows:

$$O(g(n)) = \{f \mid \exists\ c>0, \exists n0 > 0, \forall n \geq n0 : 0 \leq f \leq cg(n)\} \qquad (1)$$

In other words, $f \in (g(n))$ if and only if there exist positive constants c and $n0$ such that for all $n \geq n0$, the inequality $0 \leq f \leq cg(n)$ is satisfied. We say that f is big O of $g(n)$, or that $g(n)$ is an asymptotic upper bound for f [9].

The set of potential solutions to the problem is represented as a population of chromosomes. Initially, a random population is created, which represents different points in the search space of potential solutions [4, 6, 7]. A fitness function assigns a score (fitness) to each chromosome in the current population, which will determine its survival into the next generation. The fitness of a chromosome depends on how well that chromosome can solve the problem at hand [8].The selection of chromosomes is done on the current population based on the fitness values – chromosomes with higher fitness are more likely to be selected than those with low fitness values. This is mostly done using probabilistic methods; in evolutionary computing researches, the common methods of selection are the roulette wheel, tournament, and rank selection [1, 10, 12]. The proposed work uses the roulette wheel selection method.

The genetic algorithm performance is usually measured by the number of fitness function evaluations done during the course of a run. In this work the data is taken from the UCI Machine Learning repository in which the population size is fixed. The number of fitness function evaluation is given by the product of population size by the number of generations.
We can see that the algorithm runs in polynomial time of some degree, and the fitted models estimate the order of polynomial time the algorithm runs in for each dataset. For the given disease dataset the time complexity is O (n^2). This means that the algorithm runs in polynomial time of degree 2. This tells us that since the population size grows unboundedly, the problem size also grows unboundedly and therefore a much larger computing resource is needed for next iteration of the GA algorithm. Consequently, the algorithm's running time also increases in the order of polynomial degree of 2 depending on the data.

The genetic algorithms are used to find the feature subset from the original set and with respect to the obtained feature subset the fuzzy input variables and fuzzy rules are generated. The fuzzy inference system helps in diagnosing the presence of the disease. The time complexity is calculated to be θ(ncp), where n is the number of patterns, c is the number of clusters and p is the dimension of the data points. The proposed model is analyzed and found to execute in lesser time period.

7.CONCLUSIONS

In this paper a hybrid approach on feature selection using genetic algorithm and classification using fuzzy inference system is proposed. The time complexity of genetic algorithms and fuzzy inference system is found to taken lesser values when compared to other combinations of feature selection and classification systems. The proposed work has used genetic algorithms for effective selection of feature subset and fuzzy Logic system for efficient classification of the disease. It is also proved that this combination reduces the time complexity as well as improves the cost- effectiveness of the diagnosis model. The time complexity analysis shows that the time taken to compute a problem of size n is in the set of functions described by $O(g(n))$. Time complexity analysis can be used to predict the growth behavior of an algorithm and is useful for analyzing and optimizing the real time efficiency of the algorithm [9].

REFERENCES

[1] Holland .J.H,(1975), Adaptation in neural and artificial systems. Ann arbor: The university of Michigan press.

[2] O.cordon et.al , (1995): Genetic Algorithms and Fuzzy logic in control processes, Tehnical report, #DECSAI-95109.

[3] S.N. Sivanandam, and S.N. Deepa. (2008) : *Introduction to Genetic Algorithms*, New York: Springer-Verlag Berlin Heidelberg.

[4] E.P.Ephzibah, (2010): Cost-Effective approach on Feature selection using Genetic Algorithms and LS-SVM classifier, IJCA Special Issue on "Evolutionary Computation for optimization Techniques" ECOT.

[5] M. Gen and R. Cheng, (2000), *Genetic Algorithms and Engineering Optimization*, New York: Wiley-Interscience.

[6] Z. Michalewicz, (1996) : *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd rev. ext. ed. New York: Springer-Verlag.

[7] Knuth, D. E., (1976) Big Omicron and big Omega and big Theta. *SIGACT News* Vol. 8, No. 2), pp. 18-24.

[8 N.H. Park, C.W. Ahn and R.S. Ramakrishna, (2005) Adaptive clustering technique using genetic algorithms. *IEICE Transactions on Information and Systems*, Vol. E88-D (12), pp. 2880- 2882.

[9] Black, P. E., (2008), big-O notation, *Dictionary of Algorithms and Data Structures* [online], U.S. National Institute of Standards and Technology.

[10] M. Gen and R. Cheng, (2000), *Genetic Algorithms and Engineering Optimization*, New York: Wiley-Interscience .

[11] Z. Michalewicz, (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd rev. ext. ed.New York: Springer-Verlag .

[12] M. Mitchell, (1999)*An Introduction to Genetic Algorithms*, 5th print, Cambridge, MA: The MIT Press.

[13] Robert Detrano & M.D & PhD, V.A. Medical Center, Long Each and Cleveland Clinic Foundation. Available: www.archive.ics.uci.edu/ml/datasets/Heart+Disease.

[14] Kosko, Bart, "Fuzzy Thinking: The New Science of Fuzzy Logic", Warner, 1993 [For technical details, see Kosko, Bart, "Fuzzy cognitive maps", International Journal of Man-Machine Studies 24:65-75, 1986.]

[15] McNeill, Daniel, and Freiberger, Paul, "Fuzzy Logic: The Discovery of a Revolutionary Computer Technology", Simon and Schuster, 1992. ISBN 0-671-73843-7. [Mostly history, but many examples of applications.]

[16] Negoita, C.V., "Fuzzy Systems", Abacus Press, Tunbridge-Wells, 1981.

[17] F. Martin McNeill and Ellen Thro, "Fuzzy Logic: A practical approach", Academic Press, 1994. 350 pages, ISBN 0-12-485965.

[18] Dubois, Didier, and Prade, H., "Fuzzy Sets and Systems: Theory and Applications", Academic Press, New York, 1980.

[19] Kandel, Abraham, "Fuzzy Mathematical Techniques with Applications", Addison-Wesley, 1986.

[20] R. Kruse, J. Gebhardt, and F. Klawonn, "Foundations of Fuzzy Systems" John Wiley and Sons Ltd., Chichester, 1994. ISBN 0471-94243-X ($47.95). [Theory of fuzzy sets.]

[21] Zadeh, L.A., "Fuzzy sets," *Information and Control*, Vol. 8, pp. 338-353, 1965.

[22] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 3, No. 1, pp. 28-44, Jan. 1973.

[23] Zadeh, L.A., "The concept of a linguistic variable and its application to approximate reasoning, Parts 1, 2, and 3," *Information Sciences*, 1975, 8:199-249, 8:301-357, 9:43-80.

[24] Zadeh, L.A., "Fuzzy Logic," *Computer*, Vol. 1, No. 4, pp. 83-93, 1988.

[25] Zadeh, L.A., "Knowledge representation in fuzzy logic," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, pp. 89-100, 1989.

Author

Mrs.E.P.EPHZIBAH received the M.Sc degree (2001) and M.Phil degree (2004) in Computer science from Madurai Kamaraj University and Bharathidasan University Trichy, India. She is pursuing her Ph.D. programme in the Mother Teresa Women's University, Kodaikanal, India. She is currently working as Assistant Professor in the School of Information Technology and Engineering at VIT University, Vellore, India. Her research interests include DataMining, Feature Selection, Genetic Algorithms, Fuzzy Logic.