Using the Master-Slave Parallel Architecture for Genetic-Fuzzy Data Mining

Tzung-Pei Hong
Department of Electrical Engineering
National University of Kaohsiung
Kaohsiung, 811, Taiwan, R.O.C.
tphong@nuk.edu.tw

Yeong-Chyi Lee
Institute of Information Engineering
I-Shou University
Kaohsiung, 840, Taiwan, R.O.C.
d9003007@stmail.isu.edu.tw

Min-Thai Wu
Department of Electrical Engineering
National University of Kaohsiung
Kaohsiung, 811, Taiwan, R.O.C.
m0935104@mail.nuk.edu.tw

Abstract - *Data mining is most commonly used in attempts to induce association rules from transaction data. In the past, we used the fuzzy and GA concepts to discover both useful fuzzy association rules and suitable membership functions from quantitative values. The evaluation for fitness values was, however, quite time-consuming. In this paper, we thus propose a parallel genetic-fuzzy mining algorithm based on the master-slave architecture to extract both association rules and membership functions from quantitative transactions. The master processor uses a single population as a simple genetic algorithm does, and distributes the tasks of fitness evaluation to slave processors. The evolutionary processes, such as crossover, mutation and production are performed by the master processor. Both the theoretic analysis and the experimental results show that the speed-up of the proposed parallel algorithm can increase nearly linear along with the number of individuals to be evaluated.*

Keywords: data mining, fuzzy set, genetic algorithm, parallel processing, association rule.

1 Introduction

Recently, the fuzzy set theory [26] has been used more and more frequently in intelligent systems because of its simplicity and similarity to human reasoning [19]. Several fuzzy mining algorithms for managing quantitative data have also been proposed [8, 15, 20], where the membership functions were assumed to be known in advance. The given membership functions may, however, have a critical influence on the final mining results. Genetic algorithms (GAs) [11, 12] have also recently been used in the field of data mining since they are powerful search techniques in solving difficult problems and can provide feasible solutions in a limited amount of time. Hong et al. thus proposed a GA-based fuzzy data-mining method [17] for extracting both association rules and membership functions from quantitative transactions. In that method, the fitness evaluation is based on the suitability of derived membership functions and the number of large itemsets. The evaluation for fitness values is, however, quite time-consuming.

Due to dramatic increases in available computing power and concomitant decreases in computing costs over the last decade, learning or mining by applying parallel processing techniques has become a feasible way of overcoming the problem of slow learning. Several parallel approaches to speed up the process of data-mining were proposed [5, 18, 23]. In addition, some parallel architectures for genetic algorithms were also suggested [1][7]. They have been applied to solving timetable scheduling and discovering classification rules.

Among the parallel architectures, the master-slave architecture is particularly easy to implement. It also usually promises substantial gains in performance [9]. The master processor allocates the tasks to the slave processors and collects the results from them. It can also do its own work if necessary. As mentioned before, the fitness evaluation in genetic-fuzzy data mining is usually very time-consuming. In this paper, we thus extend our previous work [17] by using the master-slave parallel architecture to dynamically adapt membership functions and to use them to deal with quantitative transactions in fuzzy data mining. It is very natural and efficient to design a parallel GA-fuzzy mining algorithm based on the master-slave architecture. The master processor uses a single population as a simple genetic algorithm does, and distributes the tasks of fitness evaluation for suitability of membership functions and large itemsets to slave processors. The evolutionary processes, such as crossover, mutation and production are performed by the master processor. We expect that by appropriately allocating the tasks among the different types of processors, the efficiency of the proposed genetic-fuzzy mining algorithm can greatly be raised.

2 Review of related works

Some related works about data mining and genetic algorithms are first reviewed below.

2.1 Data Mining

The goal of data mining is to discover important associations among items such that the presence of some items in a transaction will imply the presence of some other items. To achieve this purpose, Agrawal and his co-workers proposed several mining algorithms based on the concept of large itemsets to find association rules in transaction data [2, 3, 4, 6]. Srikant and Agrawal then

proposed a mining method [22] to handle quantitative transactions by partitioning the possible values of each attribute. Hong et al. proposed a fuzzy mining algorithm to mine fuzzy rules from quantitative data [15]. They transformed each quantitative item into a fuzzy set and used fuzzy operations to find fuzzy rules. Cai et al. proposed weighted mining to reflect different importance to different items [8]. Each item was attached a numerical weight given by users. Weighted supports and weighted confidences were then defined to determine interesting association rules. Kaya et al. [20] proposed a GA-based clustering method to derive a predefined number of membership functions for getting a maximum profit within an interval of user specified minimum support values.

2.2 Genetic algorithm

Genetic algorithms (GAs) [11][12] have become increasingly important for researchers in solving difficult problems since they could provide feasible solutions in a limited amount of time [14]. They were first proposed by Holland in 1975 [12] and have been successfully applied to many fields. On applying genetic algorithms to solving a problem, the first step is to define a representation that describes the problem states. The most common way used is the bit string representation. An initial population of individuals, called chromosomes, is then defined and the three genetic operations (crossover, mutation, and selection) are performed to generate the next generation. Each chromosome in the population is evaluated by a fitness function to determine its goodness. This procedure is repeated until a user-specified termination criterion is satisfied. As to parallel GA, three types were proposed as single-population master-slave Gas, single-population fine-grained Gas and Multiple-population coarse-grained GAs [9]. Among the above three types, the type of master-slave parallel GAs consists of a simple structure and uses less parameter to control the process of evolution. This type of parallel GAs has been successfully applied for solving timetable scheduling and discovering classification rules [1][7]. In this paper, we will use this parallel architecture to fuzzy data mining due to its suitability.

3 A parallel genetic-fuzzy mining framework

In [17], we used the fuzzy and GA concepts to discover both useful association rules and suitable membership functions from quantitative values. The proposed approach in that paper is shown in Figure 1, where a genetic algorithm was proposed for searching membership functions suitable for mining problems and then the final best set of membership functions was used to mine association rules.

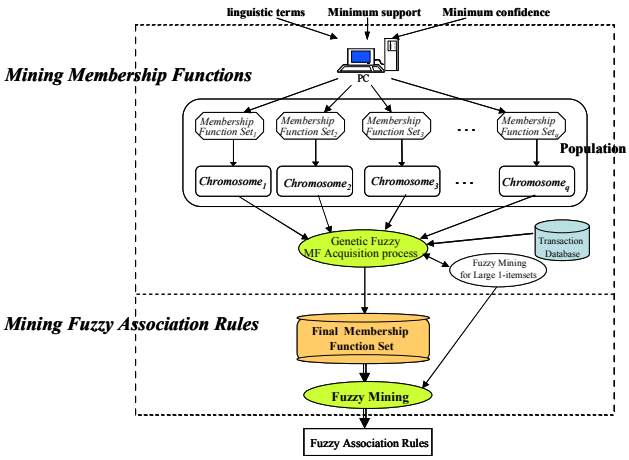

Figure 1: A GA-based approach for fuzzy data mining

In this section, a parallel genetic-fuzzy framework based on the master-slave architecture is thus proposed to speed up the mining process. The proposed framework is shown in Figure 2.

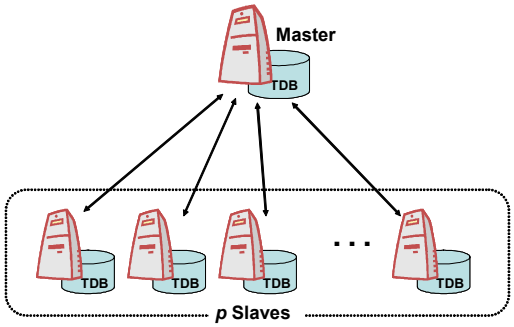

Figure 2: The proposed parallel genetic-fuzzy mining framework

In Figure 2, there are $p + 1$ processors composed of one master and p slaves. Each processor has an identical transaction database for mining. Since the fitness evaluation process is the most time-consuming part in the entire genetic-fuzzy mining process, it is thus processed in parallel by the slave processors. The other part is processed by the master processor. The parallel genetic-fuzzy data mining framework includes two phases, mining membership function and mining fuzzy association rules. In the first phase, the master and the slaves cooperate to gradually discover a set of adaptive membership functions. The master processor maintains a population of sets of membership functions and performs the genetic operators to select, mate and evolve chromosomes. The master processor distributes the tasks of fitness evaluation for suitability of membership functions and large itemsets to slave processors. Each slave processor with an identical transaction database then calculates the fitness value of a chromosome that is assigned by the master. The evaluation results from the slaves are then sent back to the master. The master processor then performs the evolutionary processes,

such as crossover, mutation and production according to the evaluation results collected. After undergoing recursive evolutions, a good set of membership functions can be obtained. The good set of membership functions is then used in the second phase by the master processor to mine fuzzy association rules. The second phase is not necessary to be processed in parallel since the most time-critical part is phase 1. The time complexity analyzed later will show this.

4 Chromosome representation and fitness evaluation

It is important to encode membership functions as string representation for GAs to be applied. Several possible encoding approaches have been described in [10, 21, 24, 25]. In this paper, each set of membership functions is encoded as a chromosome and handled as an individual with real-number schema.

In order to effectively encode the associated membership functions, we use two parameters to represent each membership function, as Parodi and Bonelli [21] did. Membership functions applied to a fuzzy rule set are then assumed to be isosceles-triangle functions. Note that other types of membership functions (e.g. non-isosceles trapezes) can also be adopted in our method. For coding non-isosceles triangles and trapezes, three and four points are needed instead of two for isosceles triangles.

In order to develop a good set of membership functions from an initial population, the genetic algorithm selects *parent* membership function sets with high fitness values for mating. Note that the selection of membership function sets is performed by the master processor, and the evaluation of each membership function set is processed by the slave processors. An evaluation function is defined to qualify the derived membership function sets. The evaluation results are then sent back to the master processor to control how the solution space is searched to promote the quality of the membership functions. The fitness value of a chromosome C_q is defined as follows:

$$f(C_q) = \frac{|L_1|}{suitabilit\ y(C_q)},$$

where suitability(C_q) is the suitability of the membership functions in a chromosome C_q and $|L_1|$ is the number of large 1-itemsets obtained by using the set of membership functions. The suitability is defined according to the two factors – overlap ratio and coverage ratio. The overlap ratio of two membership functions is defined as the overlap length divided by half the minimum span of the two functions. The coverage ratio of a set of membership functions for an item is defined as the coverage range of the functions divided by the maximum quantity of that item in the transactions. The suitability factor used in the fitness function can reduce the occurrence of the two bad kinds of membership functions shown in Figure 3, where the first one is too redundant, and the second one is too separate. Details can be referred to in [17].

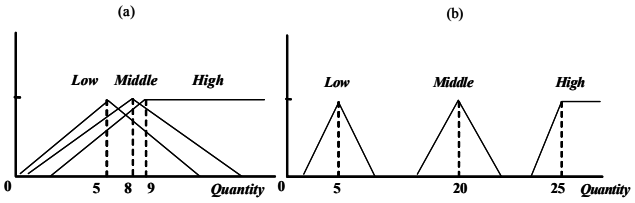

Figure 3: Two bad membership functions

Besides, using the number of large 1-itemsets can achieve a trade-off between execution time and rule interestingness. Usually, a larger number of 1-itemsets will result in a larger number of all itemsets with a higher probability, which will thus usually imply more interesting association rules. The evaluation by 1-itemsets is, however, faster than that by all itemsets or interesting association rules. It can be further speeded up by the parallel processing approach proposed in this paper.

Two genetic operators, the *max-min-arithmetical (MMA) crossover* proposed in [13] and the *one-point mutation*, are used in the genetic fuzzy mining framework. Note that the genetic operations are performed by the master processor.

5 The proposed parallel genetic-fuzzy mining algorithm

According to the above description, the proposed parallel algorithm for mining both fuzzy association rules and membership functions is described below.

The proposed parallel genetic-fuzzy mining algorithm:

INPUT: One master processor, p slave processors (p is the maximum number of individuals to be evaluated in each generation), a body of n quantitative transaction data stored in each processor, a set of m items, a support threshold α, and a confidence threshold λ.

OUTPUT: A set of fuzzy association rules with its associated set of membership functions.

STEP 1: Randomly generate a population of individuals by the master processor; each individual is a set of membership functions for all the m items.

STEP 2: Encode each set of membership functions into a string representation by the master processor.

STEP 3: Distribute the individuals from the master processor to the slave processors.

STEP 4: Calculate the fitness value of each chromosome by each corresponding slave processor by the following substeps:

STEP 4.1: For each transaction datum D_i, i = 1 to n, and for each item I_j, j=1 to m, transfer the quantitative value $v_j^{(i)}$ into a fuzzy set $f_j^{(i)}$ represented as:

$$\left(\frac{f_{j1}^{(i)}}{R_{j1}} + \frac{f_{j2}^{(i)}}{R_{j2}} + ... + \frac{f_{jl}^{(i)}}{R_{jl}} \right),$$

using the corresponding membership functions represented by the chromosome, where R_{jk} is the k-th fuzzy region (term) of item I_j, $f_{jl}^{(i)}$ is $v_j^{(i)}$'s fuzzy membership value in region R_{jk}, and l (= $|I_j|$) is the number of linguistic terms for I_j.

STEP 4.2: For each item region R_{jk}, calculate its scalar cardinality $count_{jk}$ on the transactions as follows:

$$count_{jk} = \sum_{i=1}^{n} f_{jk}^{(i)} .$$

STEP 4.3: For each R_{jk}, $1 \le j \le m$ and $1 \le k \le |I_j|$, check whether its $count_{jk}$ is larger than or equal to the minimum support threshold α. If R_{jk} satisfies the above condition, put it in the set of large 1-itemsets (L_1). That is:

$$L_1 = \{ R_{jk} \mid count_{jk} \ge \alpha,\ 1 \le j \le m \text{ and } 1 \le k \le |I_j| \}.$$

STEP 4.4: Set the fitness value of the chromosome as the number of large itemsets in L_1 divided by s*uitability*(C_q). That is:

$$f(C_q) = \frac{|L_1|}{suitability(C_q)}.$$

STEP 5: Send the fitness value $f(C_q)$ of each chromosome C_q from each slave processor to the master processor.

STEP 6: Execute the crossover operations on the population by the master processor.

STEP 7: Execute the mutation operations on the population by the master processor.

STEP 8: Distribute the individuals to be evaluated from the master processor to the slave processors.

STEP 9: Calculate the fitness value of each chromosome by each corresponding slave processor as in STEP 4.

STEP 10: Send the fitness value $f(C_q)$ of each chromosome C_q from each slave processor to the master processor.

STEP 11: Use the defined selection criteria to choose suitable individuals for the next generation by the master slave.

STEP 12: If the termination criterion is not satisfied, go to Step 6; otherwise, do the next step.

STEP 13: Use the set of membership functions with the highest fitness value for finding all fuzzy large itemsets by the master processor.

STEP 14: Find the fuzzy association rules from the fuzzy large itemsets by the master processor.

In Steps 13 and 14, our fuzzy mining algorithm proposed in [16] can be used to find the results.

6 Time complexity analysis

The time complexities of both the sequential and the parallel genetic-fuzzy mining algorithms are first analyzed. The speed-up of the parallel mining algorithm is then derived. Some notation is first defined as follows:

p: the number of slave processors (the maximum number of individuals to be evaluated);
n: the number of generations;
f: *the average execution time of calculating the* fitness value of an individual in each generation;
g: the average execution time of processing all genetic operations during a generation;
c: the average communication time between a master processor and a slave processor during a generation;
s: the average execution time of mining the association rules in the second phase;
T_{ave}^{s}: the average execution time of the sequential GA-fuzzy mining algorithm;
T_{ave}^{p}: the average execution time of the parallel GA-fuzzy mining algorithm;
S_{ave}: the average speed-up calculated by T_{ave}^{s} over T_{ave}^{p}.

The average time complexity of the sequential mining algorithm is:

$$T_{ave}^{s} = n * (p * f + g) + s$$

The paralllel mining algorithm can distribute the fitness-evaluation tasks to the slave processors. Therefore, the execution time for the fitness evaluaion in each generation needs only *f*. The parallel algorithm, however, needs additional computation time *c* between a master

processor and slave processors. The average time complexity of the paralled mining algorithm is thus:

$$T_{ave}^{p} = n*(f+g+c)+s.$$

The average speed-up is thus:

$$S_{ave} = \frac{T_{ave}^{s}}{T_{ave}^{p}} = \frac{n(p*f+g)+s}{n(f+g+c)+s}.$$

In this paper, the fitness evaluation is based on the suitability of derived membership functions and the number of large 1-itemsets. The entire database must be scanned to find the large 1-itemsets. The average evaluation time f is thus much larger than the average execution time for genetic operations g, especially when the processed dataset is large. In addition, since the master only distributes the code of a chromosome to a slave and receives a number (the evaluation value) from the slave, the communication time is thus very little. Therefore, the speed-up can be further simplified as the following:

$$S_{ave} = \frac{n(p*f+g)+s}{n(f+g+c)+s} \approx \frac{n*p*f+s}{n*f+s}.$$

The average evaluation time f includes the time of finding large 1-itemsets. The average execution time s of mining association rules in the second phase includes finding all large itemsets and deriving rules from them. s is thus larger than f. However, because the number of items in the longest large itemsets is usually small, some pruning techniques may be used to reduce the mining time s, and the number of generations is usually set at more than one hundred, s may thus be much smaller than $n*f$, especially when n is large. In this case, the above speed-up can be further simplified as:

$$S_{ave} \approx \frac{n*p*f+s}{n*f+s} \approx \frac{n*p*f}{n*f} = p.$$

Thus, when the number of generations is large, the speed-up is nearly linear.

7 Experimental results

In this section, the experiments made to show the performance of the proposed approach are described. They were simulated in Java on a personal computer with Intel Pentium IV 3.2GHz and 512MB RAM. 64 items and 10000 transactions were used in the experiments. In each data set, the numbers of purchased items in transactions were first randomly generated. The purchased items and their quantities in each transaction were then generated. An item could not be generated twice in a transaction. The crossover rate p_c is set at 0.8, and the mutation rate p_m is set at 0.01. The minimum support α is set at 400. Experiments with different population sizes from 10 to 50 were made to show the speed-up trend of the proposed algorithm. The communication time was not considered. Note that by the genetic-fuzzy mining algorithm proposed in [17], the maximum number p of individuals to be evaluated in each generation was 2.6*r, where r is the number of individuals in a population. The experimental results are shown in Figure 4.

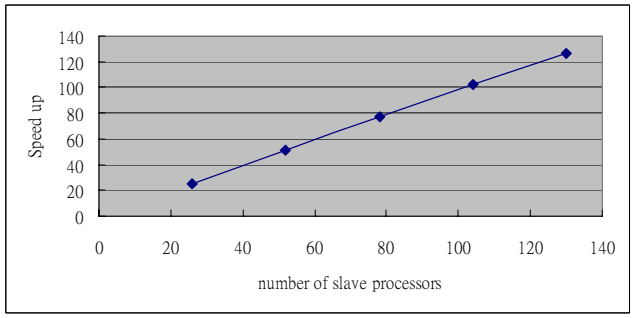

Figure 4: The experimental results for speed-up with transaction I/O time

It can be easily observed from the above figure that the speed-up increases nearly linearly along with the number of slave processors. The speed-up is also close to the number of slave processors. It is quite consistent with our analysis in Section 6.

8 Conclusions

In this paper, we have proposed a parallel genetic-fuzzy mining algorithm based on the master-slave architecture to extract both association rules and membership functions from quantitative transactions. The master and the slaves first cooperate to discover a set of suitable membership functions. The set of membership functions found is then used by the master processor to mine fuzzy association rules. The second phase is not necessary to be processed in parallel since the most time-critical part lies in phase 1. The time complexities for both sequential and parallel genetic-fuzzy mining algorithm have been analyzed, with results showing the good effect of the proposed approach. When the number of generations is large, the speed-up can be nearly linear. The experimental results have also shown this point. Applying the master-slave parallel architecture to speed up the genetic-fuzzy data mining algorithm is thus a feasible way to overcome the low-speed fitness evaluation problem of the original algorithm.

Acknowledgment

The authors would like to thank Mr. Chun-Hao Chen for his help in making the experiments. This research was supported by the National Science Council of the Republic of China under contract NSC93-2213–E -390-001.

References

[1] D. Abramson and J. Abela, “A parallel genetic algorithm for solving the school timetabling problem”. The Fifteenth Australian Computer Science Conference, pp. 1–11, 1992.

[2] R. Agrawal, T. Imielinksi and A. Swami, “Mining association rules between sets of items in large database,“ The 1993 ACM SIGMOD Conference, pp. 207-216, 1993.

[3] R. Agrawal, T. Imielinksi and A. Swami, “Database mining: a performance perspective,” IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 6, pp. 914-925, 1993.

[4] R. Agrawal and R. Srikant, “Fast algorithm for mining association rules,” The International Conference on Very Large Databases, pp. 487-499, 1994.

[5] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 962-969, 1996.

[6] R. Agrawal, R. Srikant and Q. Vu, “Mining association rules with item constraints,” The Third International Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California, pp. 67-73, 1997.

[7] D. L. A. Araujo, H. S. Lopes, and A. A. Freitas, “A parallel genetic algorithm for rule discovery in Large Databases,” The IEEE International Conference on Systems, Man and Cybernetics Conference, Vol. 3, pp. 940-945, 1999.

[8] C. H. Cai, W. C. Fu, C. H. Cheng and W. W. Kwong, “Mining association rules with weighted items,” The International Database Engineering and Applications Symposium, pp. 68-77, 1998.

[9] E. Cantu-Paz, “A survey of parallel genetic algorithms,” Calculateurs Paralleles, Reseaux et Systems Repartis. Vol. 10, No. 2, pp. 141-171, 1998.

[10] O. Cordón, F. Herrera and P. Villar, “Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base,” IEEE Transactions on Fuzzy Systems, Vol. 9, No. 4, pp. 667-674, 2001.

[11] D. E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison Wesley, 1989.

[12] J. H. Holland. Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[13] F. Herrera, M. Lozano and J. L. Verdegay, “Fuzzy connectives based crossover operators to model genetic algorithms population diversity,” Fuzzy Sets and Systems, Vol. 92, No. 1, pp. 21–30, 1997.

[14] A. Homaifar, S. Guan, and G. E. Liepins, “A new approach on the traveling salesman problem by genetic algorithms,” The Fifth International Conference on Genetic Algorithms, 1993.

[15] T. P. Hong, C. S. Kuo and S. C. Chi, "A data mining algorithm for transaction data with quantitative values," Intelligent Data Analysis, Vol. 3, No. 5, pp. 363-376, 1999.

[16] T. P. Hong, C. S. Kuo and S. C. Chi, "Trade-off between time complexity and number of rules for fuzzy mining from quantitative data," International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, Vol. 9, No. 5, pp. 587-604, 2001.

[17] T. P. Hong, C. H. Chen, Y. L. Wu and Y. C. Lee, "Mining membership functions and fuzzy association rules", The 2003 Joint Conference on AI, Fuzzy System, and Grey System, 2003.

[18] M. Joshi, E. H. Han, G. Karypis and V. Kumar. “Efficient parallel algorithms for mining associations.” Parallel and Distributed Systems, Vol. 1795, pp. 418-429, 2000.

[19] A. Kandel, Fuzzy Expert Systems, CRC Press, Boca Raton, pp.8-19, 1992.

[20] M. Kaya, and R. Alhajj, “A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining,” The IEEE International Conference on Fuzzy Systems, pp. 881 -886, 2003.

[21] A. Parodi and P. Bonelli, "A new approach of fuzzy classifier systems," Proceedings of Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 223-230, 1993.

[22] R. Srikant and R. Agrawal, “Mining quantitative association rules in large relational tables,” The 1996 ACM SIGMOD International Conference on Management of Data, pp. 1-12, 1996.

[23] A. Veloso, W. Meira Jr. and S. Parthasarathy, “New parallel algorithms for frequent itemset mining in very large databases”, The Fifteenth Symposium on Computer Architecture and High Performance Computing, pp. 158-166, 2003.

[24] C. H. Wang, T. P. Hong and S. S. Tseng, “Integrating fuzzy knowledge by genetic algorithms,” IEEE Transactions on Evolutionary Computation, Vol. 2, No. 4, pp. 138-149, 1998.

[25] C. H. Wang, T. P. Hong and S. S. Tseng, “Integrating membership functions and fuzzy rule sets from multiple knowledge sources,” Fuzzy Sets and Systems, Vol. 112, pp. 141-154, 2000.

[26] L. A. Zadeh, “Fuzzy sets,” Information and Control, Vol. 8, No. 3, pp. 338-353, 1965.