Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

A Comparison of Different Fitness Functions for Extracting Membership Functions Used in Fuzzy Data Mining

Chun-Hao Chen[1], Tzung-Pei Hong[2], Vincent S. Tseng[1]
[1]*Department of Computer Science and Information Engineering, National Cheng-Kung University*
{chchen, vincent}@idb.csie.ncku.edu.tw
[2]*Department of Electrical Engineering, National University of Kaohsiung*
tphong@nuk.edu.tw

***Abstract*—In this paper, a GA-based framework for finding membership functions suitable for fuzzy mining problems is proposed. Each individual represents a possible set of membership functions for the items and is divided into two parts, control genes and parametric genes. Control genes are encoded into binary strings and used to determine whether membership functions are active or not. Each set of membership functions for an item is encoded as parametric genes with real-number schema. Seven fitness functions are proposed, each of which is used to evaluate the goodness of the obtained membership functions and used as the evolutionary criteria in GA. Experiments are also made to show the effectiveness of the framework and to compare the seven fitness functions.**

I. INTRODUCTION

Data mining is commonly used to induce association rules from transaction data [1]. Most previous studies focused on binary valued transaction data. Transaction data in real-world applications, however, usually consist of quantitative values. Designing a sophisticated data-mining algorithm able to deal with various types of data presents a challenge to workers in this research field.

Recently, fuzzy set theory has been used more and more frequently in intelligent systems because of its simplicity and similarity to human reasoning [7]. Several fuzzy learning algorithms for inducing rules from given sets of data have been designed and used to good effect with specific domains [3]. As to fuzzy data mining, Hong *et al.* proposed an algorithm to mine fuzzy rules from quantitative data [6]. 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 [2]. Each item was attached a numerical weight given by users. Weighted supports and weighted confidences were then defined to determine interesting association rules. Yue *et al.* then extended their concepts to fuzzy item vectors [13]. In the above approaches, the membership functions were assumed to be known in advance. Although many approaches for learning membership functions were proposed [3, 12], most of them were usually used for classification or control problems.

In the fuzzy mining problem, the given membership functions may have a critical influence on the final mining results. Kaya *et al.* proposed a GA-based approach to derive a predefined number of membership functions for getting a maximum profit within an interval of user specified minimum support values [8]. Kaya *et al.* also proposed a multi-objective genetic algorithm to find a number of Pareto-optimal rules sets according to two objective functions, number of rules and execution time [9]. We also proposed a fuzzy data-mining framework for extracting both association rules and membership functions from quantitative transactions [5]. It maintained a population of sets of membership functions, and used the genetic algorithm to automatically derive the resulting one. The number of membership functions was, however, predefined.

This paper thus modifies the previous algorithm [5] and compares seven fitness evaluation functions for extracting an appropriate number of linguistic terms and their membership functions used in fuzzy data mining for the given items.

II. A GA-BASED MINING FRAMEWORK

The proposed framework is shown in Fig. 1.

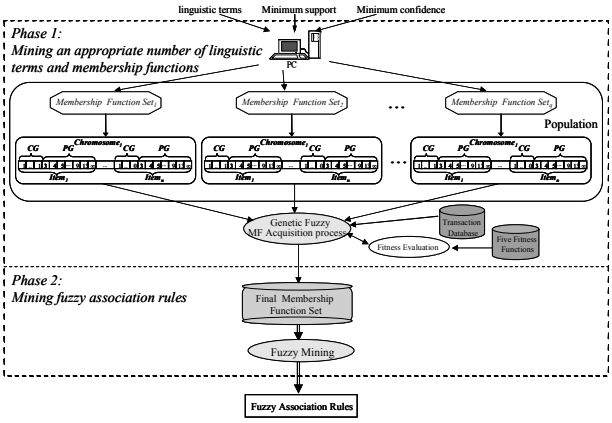

Fig. 1. A GA-based framework for searching for an appropriate number of linguistic terms and membership functions

The proposed framework is divided into two phases: mining

an appropriate number of linguistic terms and membership functions, and mining fuzzy association rules. In the first phase, it maintains a population of individuals, each including control genes (CG) and parametric genes (PG), and uses the genetic algorithm to automatically derive the resulting one. Control genes are encoded into bit strings and used to determine whether parametric genes are active or not. Each set of membership functions for an item is encoded as parametric genes with real-number schema. The proposed approach first transforms each set of membership functions into a fixed-length string. It then chooses appropriate strings for "mating", gradually creating good offspring membership function sets. The offspring membership function sets then undergo recursive "evolution" until a good set of membership functions has been obtained. Next, in the phase of mining fuzzy association rules, the sets of membership function for all the items are used to mine the interesting rules from the given quantitative database. The fuzzy mining algorithm proposed in [6] is adopted to achieve this purpose.

III. FUZZY-GENETIC DATA MINING

A. Chromosome Representation

It is important to encode membership functions as string representation for GAs to be applied. Several possible encoding approaches have been described in [3, 11]. In this paper, we adopt the Hierarchical Genetic Algorithm (HGA) to represent individuals [10], in order to find appropriate number of linguistic terms and membership functions of each item.

Each individual is thus divided into two parts, control genes and parametric genes. In the first part, control genes are encoded into binary strings and used to determine whether parametric genes are active or not. In the second part, each set of membership functions for an item is encoded as parametric genes with real-number schema. In order to effectively encode the associated membership functions, we assume the membership functions are isosceles-triangular and use two parameters to represent each membership function as Parodi and Bonelli [11] did. Figure 2 shows the membership functions for item I_j, where R_{jk} denotes the membership function of the k-th linguistic term of item I_j, c_{jk} indicates the center abscissa of fuzzy region R_{jk}, and w_{jk} represents half the spread of fuzzy region R_{jk}. As Parodi and Bonelli did, we then represent each membership function as a pair (c, w). Thus, all pairs of (c, w)'s for a certain item are concatenated to represent its membership functions.

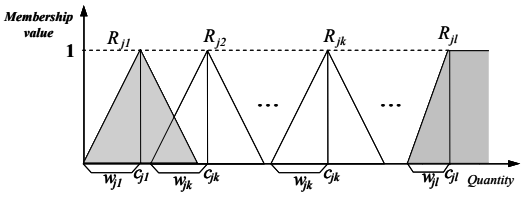

Fig. 2. Membership functions of item I_j

Thus the set of membership functions MF_j for the first item I_j is then represented as a substring of $a_{j1}a_{j2}\ldots a_{jl}$, $c_{j1}w_{j1}\ldots c_{jl}w_{jl}$, where a_{jk} represents the k-th control gene of I_j and l is the maximum possible number of linguistic terms of I_j. The entire set of membership functions is then encoded by concatenating substrings of MF_1, MF_2, ..., MF_m. An example is given below to demonstrate the process of encoding a set of membership functions.

Example 1: Assume there are four items in a transaction database: milk, bread, cookies and beverage. Also assume the maximum possible number of linguistic terms for each item is 3. If there is a chromosome shown in Fig. 3, then it represents the membership functions shown in Fig. 4. according to the encoding scheme mentioned above.

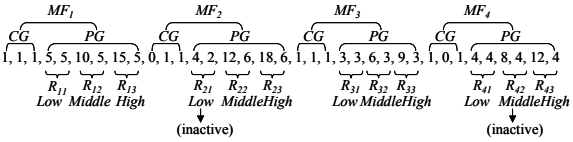

Fig. 3. A chromosome representation of membership functions

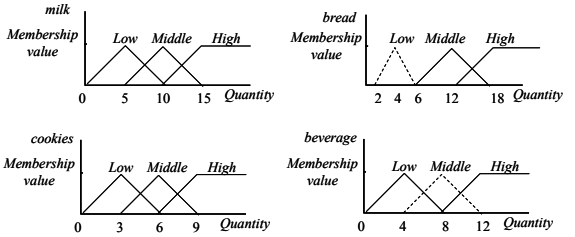

Fig. 4. The membership functions for the four items represented by the chromosome in Fig. 3

In Fig. 4, solid lines represent active membership functions and dash lines represent inactive ones. Since the membership functions of the item *milk* are encoded as (5, 5, 10, 5, 15, 5), *milk* thus has three possible linguistic terms. Let them be called *Low*, *Middle* and *High*. In Fig. 3, all the three bits in the control genes of MF_1 are 1, representing the three membership functions are active. The membership functions for *bread* are encoded as (4, 2, 12, 6, 18, 6). *Bread* thus also has three possible linguistic terms. The first bit in the control genes of MF_2 is 0, representing the first membership function (*Low*) is inactive. The membership functions for the other two items, *cookies* and *beverage*, can be similarly explained.

Note that other types of membership functions 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.

According to the proposed representation, each chromosome thus consists of a set of membership functions for all the items. This representation allows genetic operators to search for appropriate solutions.

B. Initial Population

A genetic algorithm requires a population of feasible solutions to be initialized and updated during the evolution process. As mentioned above, each individual within the population is a set of isosceles-triangular membership functions. Each membership function corresponds to a linguistic term of a certain item. The initial set of chromosomes is randomly generated with some constraints to form feasible

membership functions. A feasible set of membership functions must satisfy the condition that $c_{j1} < c_{j2} < \ldots < c_{jl}$, where c_{jk} is the center value of the k-th membership function in the j-th chromosome.

C. Genetic Operators

Genetic operators are important to the success of specific GA applications. In our approach, different crossover operators are performed for control genes and parametric genes. For control genes, the single-point crossover and the binary one-point mutation operators are used. For parametric genes, the max-min-arithmetical (MMA) crossover operator proposed in [4] and the one-point mutation for real numbers are used. Assume the parametric genes in two parent chromosomes are shown below:

$$C_1^t = (c_1, \ldots\ldots, c_h, \ldots\ldots, c_Z),$$
$$C_2^t = (c_1', \ldots\ldots, c_h', \ldots\ldots, c_Z'),$$

The max-min-arithmetical (MMA) crossover operator will generate the following four candidate chromosomes from them.

1. $C_1^{t+1} = (c_{11}^{t+1}, ..., c_{1h}^{t+1}, ..., c_{1Z}^{t+1})$, where
   $c_{1h}^{t+1} = dc_h + (1-d)c_h'$, $1 \le h \le Z$,
2. $C_2^{t+1} = (c_{21}^{t+1}, ..., c_{2h}^{t+1}, ..., c_{2Z}^{t+1})$, where
   $c_{2h}^{t+1} = dc_h' + (1-d)c_h$, $1 \le h \le Z$,
3. $C_3^{t+1} = (c_{31}^{t+1}, ..., c_{3h}^{t+1}, ..., c_{3Z}^{t+1})$, where
   $c_{3h}^{t+1} = \min\{c_h, c_h'\}$, $1 \le h \le Z$,
4. $C_4^{t+1} = (c_{41}^{t+1}, ..., c_{4h}^{t+1}, ..., c_{4Z}^{t+1})$, where
   $c_{4h}^{t+1} = \max\{c_h, c_h'\}$, $1 \le h \le Z$,

where the parameter d is either a constant or a variable whose value depends on the age of the population. The best two chromosomes of the four candidates are then chosen as the offspring.

The one-point mutation operator will create a new fuzzy membership function by adding a random value ε (between $-w_{jk}$ to $+ w_{jk}$) to the center or to the spread of an existing linguistic term, say R_{jk}. Assume that c and w represent the center and the spread of R_{jk}. The center or the spread of the newly derived membership function will be changed to $c + \varepsilon$ or $w + \varepsilon$ by the mutation operation. Mutation at the center of a fuzzy membership function may however disrupt the order of the resulting fuzzy membership functions. These fuzzy membership functions then need rearrangement according to their center values.

D. Fitness functions

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. An evaluation function is then needed to qualify the derived membership function sets. The performance of membership function sets is then fed back to the genetic algorithm to control how the solution space is searched to promote the quality of the membership functions. Before the proposed fitness functions are described, several related terms are first explained.

The overlap ratio of two membership functions R_{jk} and R_{ji} is defined as the overlap length divided by half the minimum span of the two functions. If the overlap length is larger than half the span, then these two membership functions are thought of as a little redundant. Appropriate punishment must then be considered in this case. Thus, the overlap factor of the membership functions for an item I_j in the chromosome C_q is defined as:

$$overlap\_factor(C_{qj}) = \sum_{\substack{k \ne i \\ R_{jk}, R_{ji} \text{ are active}}} [\max((\frac{overlap(R_{jk}, R_{ji})}{\min(w_{jk}, w_{ji})}), 1) - 1]$$

where $overlap(R_{jk}, R_{ji})$ is the overlap length of R_{jk} and R_{ji}. The coverage ratio of a set of membership functions for an item I_j is defined as the coverage range of the functions divided by the maximum quantity of that item in the transactions. The more the coverage ratio is, the better the derived membership functions are. Thus, the coverage factor of the membership functions for an item I_j in the chromosome C_q is defined as:

$$coverage\_factor(C_{qj}) = \frac{1}{\frac{range(R_{j1}, ..., R_{jl})}{max(I_j)}}$$

where $range(R_{j1}, R_{j2}, \ldots, R_{jl})$ is the coverage range of the membership functions, l is the number of membership functions for I_j, and $max(I_j)$ is the maximum quantity of I_j in the transactions.

The usage ratio of membership functions for an item I_j is defined as the number of large-1 itemsets for I_j divided by the number of active linguistic terms. Note that the maximum possible number of large-1 itemsets for an item is the number of its active linguistic terms. The more the usage ratio is, the better the derived membership functions are. Thus, the usage factor of the membership functions for an item I_j in the chromosome C_q is defined as:

$$usage\_factor(C_{qj}) = \frac{l_{C_{qj}}}{\max(|L_1^{C_{qj}}|, 1)}$$

where $l_{C_{qj}}$ is the active linguistic terms of chromosome C_{qj} and $max(|L_1^{C_{qj}}|, 1)$ is the maximum of the number of large-1 itemsets and 1.

The suitability of the membership functions in a chromosome C_q is thus defined as:

$$\sum_{j=1}^{m}[\, k_1 * overlap\_factor(C_{qj}) + k_2 * coverage\_factor(C_{qj}) + k_3 * usage\_factor(C_{qj})],$$

where m is the number of items and k_1, k_2, k_3 are weighting factors.

The suitability factor used in the fitness functions can reduce the occurrence of the two bad kinds of membership functions shown in Fig. 5, where the first one is too redundant, and the second one is too separate. The overlap factor in *suitability*(C_q) is designed for avoiding the first bad case, and the coverage factor is for the second one. The usage factor is used to avoid too many linguistic terms.

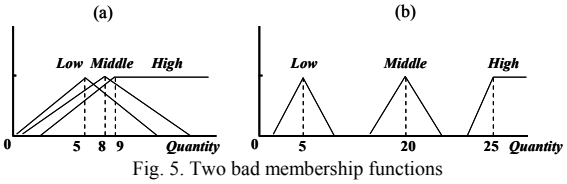

Fig. 5. Two bad membership functions

Several possible fitness functions to evaluate a chromosome C_q are defined as follows.

1. $f_1(C_q) = suitability(C_q)$. A smaller value means a better quality. It thus focuses only on the suitability of membership functions.

2. $f_2(C_q) = |L_1|$, where $|L_1|$ is the number of large 1-itemsets obtained by the set of membership functions in C_q. A larger fitness value means a better quality. It thus focuses only on the number of derived large-1 itemsets.

3. $f_3(C_q) = |L_1|/suitability(C_q)$. A larger value means a better quality. It is a compromise between the number of derived large-1 itemsets and the suitability of derived membership functions.

4. $f_4(C_q) = |L_{all}|$, where $|L_{all}|$ is the number of all large itemsets obtained by the set of membership functions in C_q. A larger fitness value means a better quality. It thus focuses only on the number of all derived large itemsets.

5. $f_5(C_q) = |L_{all}|/suitability(C_q)$. A larger value means a better quality. It is a compromise between the number of all derived large itemsets and the suitability of derived membership functions.

6. $f_6(C_q) = |R|$, where $|R|$ is the number of fuzzy association rules obtained by the set of membership functions in C_q. A larger fitness value means a better quality. It thus focuses only on the number of all derived association rules.

7. $f_7(C_q) = |R|/suitability(C_q)$. A larger value means a better quality. It is a compromise between the number of all derived association rules and the suitability of derived membership functions.

Functions 2 and 3 use the number of large 1-itemsets to evaluate chromosomes. They can save much execution time when the evaluation is done. 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.

IV. The Proposed Mining Algorithm

According to the above description, the proposed algorithm for finding an appropriate number of linguistic terms, their membership functions, and the corresponding association rules is described below.

The proposed mining algorithm:

INPUT: A body of n quantitative transaction data, a set of m items, a maximum possible number T of linguistic terms, 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 P individuals; each individual is a set of membership functions for all m items.

STEP 2: Encode each set of membership functions into a string representation.

STEP 3: Select the desired fitness function.

STEP 4: Calculate the fitness value of each chromosome according to the selected fitness function.

STEP 5: Execute crossover operations on the population.

STEP 6: Execute mutation operations on the population.

STEP 7: Use the selection criteria to choose individuals for the next generation.

STEP 8: If the termination criterion is not satisfied, go to Step 4; otherwise, do the next step.

STEP 9: Output the set of membership functions with the best fitness value.

STEP 10: Adopt the fuzzy mining algorithm proposed in [6] to mine fuzzy association rules from the given database.

V. Experimental Results

In this section, experiments made to show the performance of the proposed approach are described. They were implemented in Java on a personal computer with Intel Pentium 4 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 initial population size P was set at 50, the crossover rate p_c was set at 0.8, and the mutation rate p_m was set at 0.001. The parameter d of the crossover operator was set at 0.35 according to [3] and the minimum support α was set at 0.05 (5%). The third fitness function (f_3) was first used to show the performance of the proposed algorithm since it could make a good trade-off between execution time and accuracy. After 500 generations, the final membership functions were apparently much better than the original ones. For example, the initial membership functions of some four items among the 64 items are shown in Fig. 6, where the membership functions with dash lines mean they are inactive.

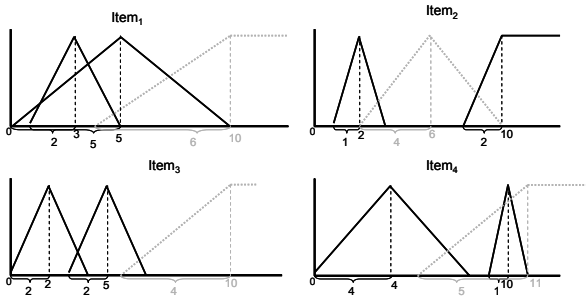

Fig. 6. The initial membership functions of some four items

In Fig. 6, the membership functions had the two bad types of shapes according to the definition in the previous section. For example, the membership functions for $Item_1$ overlap too much. After 500 generations, the final membership functions for the same four items are shown in Fig. 7. It is easily seen that the membership functions in Fig. 7 was better than those in Fig. 6. The two bad kinds of membership functions didn't appear in the final results.

The average fitness values of the chromosomes along with different numbers of generations are shown in Fig. 8. As expected, the curve gradually went upward, finally converging to a certain value.

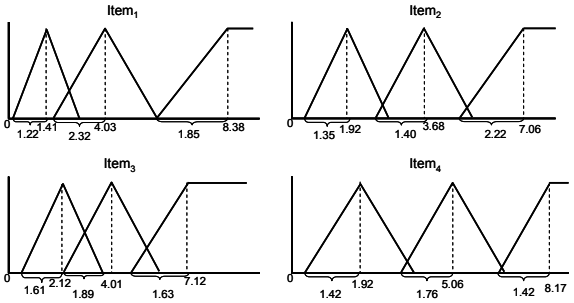

Fig. 7. The final membership functions of some four items after 500 generations

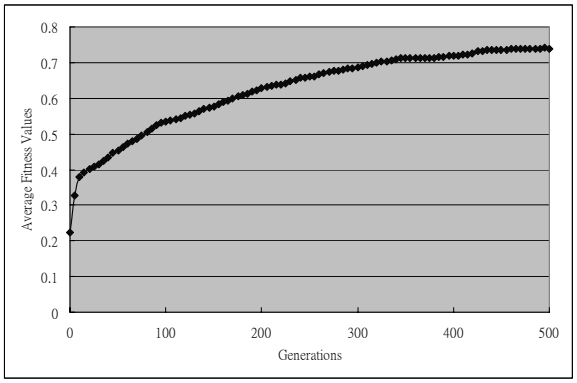

Fig. 8: The average fitness values along with different numbers of generations

Next, experiments were made to compare the results by using f_1 (only suitability), f_2 (only $|L_1|$) and f_3 as the fitness functions. For the same experimental environments and data, the membership functions of the above four items after 500 generations by using f_1 as the fitness function are shown in Fig. 9, and by using f_2 are shown in Fig. 10.

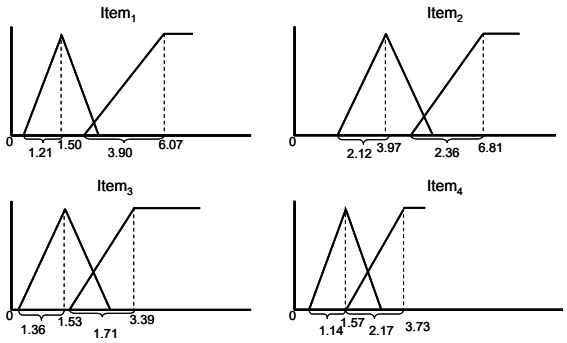

Fig. 9. The final membership functions when only the suitability is considered

It can be easily seen from Fig. 9 that the derived membership functions by considering only suitability were satisfactory because the suitability measure was designed for getting good shapes of membership functions. Its number of large 1-itemsets was, however, less than the original one (which will be shown later). On the contrary, it was very natural for the derived membership functions by considering only the number of large 1-itemsets to have a bad shape from Fig. 10. Their overlap degrees were quite high.

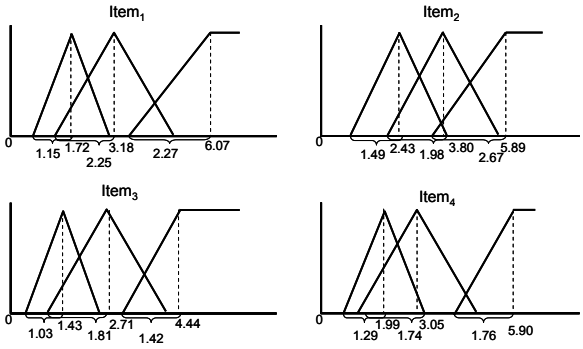

Fig. 10. The final membership functions when only $|L_1|$ is considered

The numbers of large 1-itemsets by the fitness functions f_1, f_2 and f_3, along with different generations were further compared, with results shown in Fig. 11.

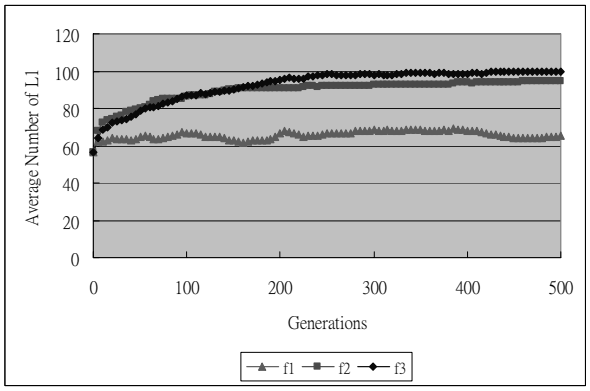

Fig. 11. The average number of large 1-itemsets by the three different fitness functions

It can be easily seen from Fig. 11 that the number of large 1-itemsets by f_1 (only the suitability) is the least among the three fitness functions. The suitability values by the three fitness functions along with different generations are shown in Fig. 12.

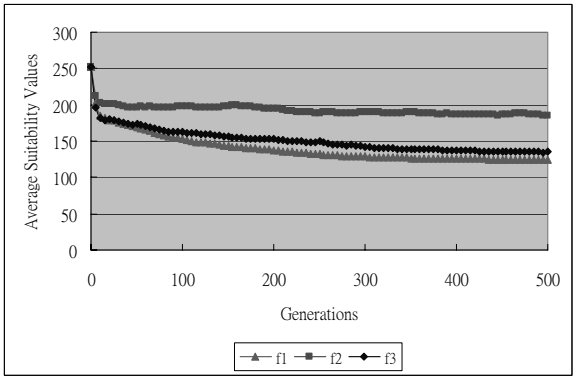

Fig. 12. The average value of suitability by the three different fitness functions

It can be easily seen from Fig. 12 that the suitability by only f_2 (the number of large 1-itemsets) was the worst among the three fitness functions. The average values of $|L_1|/suitability(C_q)$ calculated by the proposed algorithm with the three fitness functions f_1, f_2 and f_3, along with different generations are shown in Fig. 13.

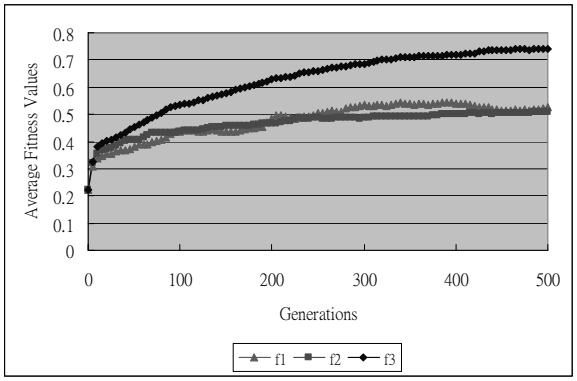

Fig. 13. The average value of fitness values by the three different fitness functions

It can be easily seen from Fig. 13. that the average values of $|L_1|/suitability(C_q)$ by f_3 was the best among the three fitness functions. f_3 could thus achieve a good trade-off between numbers of large-1 itemsets and suitability of membership functions.

A comparison for f_1, f_4, f_5 and for f_1, f_6, f_7 showed a similar result to the comparison for f_1, f_2, f_3. At last, the three fitness functions f_3, f_5, f_7 were compared. The execution time by f_3 was much less than that by f_5, which was much less than that by f_7. The number of association rules by f_3 was less than that by f_5, which was less than that by f_7. The rule numbers by the three fitness functions were quite close. The trade-off achieved by f_3 was thus acceptable.

VI. Conclusions and Future Works

This paper has described a fuzzy-genetic framework for extracting an appropriate number of linguistic terms, their membership functions and fuzzy association rules from quantitative transactions. Seven fitness evaluation functions have also been proposed to evaluate the goodness of the obtained membership functions and used as the evolutionary criteria in GA. The proposed algorithm can thus adjust membership functions by genetic algorithms and then uses them to fuzzify the quantitative transactions. After the GA process terminates, a better set of association rules can then be expected with a more suitable set of membership functions.

Experiments have also been made to show the effectiveness of the framework and to compare the seven fitness functions. From the experimental results, it can be inferred that f_3 ($|L_1|/suitability(C_q)$) has a good trade-off among the three fitness functions f_1, f_2, and f_3. Besides, f_5 ($|L_{all}|/suitability(C_q)$) and f_7 ($|R|/suitability(C_q)$) has a similar result among f_1, f_4, f_5 and among f_1, f_6, f_7. The execution time by f_3 is also less than f_5 and f_7, but the rule numbers by the three fitness functions are quite close. Using the number of large 1-itemsets and the suitability of membership functions in the fitness evaluation can thus achieve a good trade-off between execution time and rule interestingness. In the future, we will continuously attempt to enhance the GA-based mining framework for more complex problems.

Acknowledgment

This research was supported by the National Science Council of the Republic of China under contract NSC 95-2221-E-390-025.

References

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

[2] 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.

[3] 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, 2001.

[4] 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.

[5] T. P. Hong, C. H. Chen, Y. L. Wu and Y. C. Lee, "A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions", *Soft Computing*, Vol. 10, No. 11, pp. 1091-1101. 2006.

[6] 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.

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

[8] 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.

[9] M. Kaya, R. Alhajj, "Integrating multi-objective genetic algorithms into clustering for fuzzy association rules mining," *The Fourth IEEE International Conference on Data Mining*, pp. 431-434, 2004.

[10] K. F. Man, K. S. Tang, S. Kwong and W. A. Halang, "*Genetic Algorithms*," Springer-Verlag London Limited, 1999.

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

[12] H. Roubos and M. Setnes, "Compact and transparent fuzzy models and classifiers through iterative complexity reduction," *IEEE Transactions on Fuzzy Systems*, Vol. 9, No. 4, pp. 516-524, 2001.

[13] S. Yue, E. Tsang, D. Yeung and D. Shi, "Mining fuzzy association rules with weighted items," *The IEEE International Conference on Systems, Man and Cybernetics*, pp. 1906-1911, 2000