Soft Comput (2006) 10: 1091–1101
DOI 10.1007/s00500-006-0046-x

ORIGINAL PAPER

Tzung-Pei Hong · Chun-Hao Chen · Yu-Lung Wu
Yeong-Chyi Lee

A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions

Published online: 22 March 2006

Abstract Data mining is most commonly used in attempts to induce association rules from transaction data. Transactions in real-world applications, however, usually consist of quantitative values. This paper thus proposes a fuzzy data-mining algorithm for extracting both association rules and membership functions from quantitative transactions. We present a GA-based framework for finding membership functions suitable for mining problems and then use the final best set of membership functions to mine fuzzy association rules. The fitness of each chromosome is evaluated by the number of large 1-itemsets generated from part of the previously proposed fuzzy mining algorithm and by the suitability of the membership functions. Experimental results also show the effectiveness of the framework.

Keywords Data mining · Genetic algorithm · Fuzzy set · Membership function · Association rule

1 Introduction

Data mining is most commonly used in attempts to induce association rules from transaction data. An association rule is an expression $X \rightarrow Y$, where X is a set of items and Y is a single item [1]. It means in the set of transactions, if all the items in X exist in a transaction, then Y is also in the transaction with a high probability. For example, assume whenever customers in a supermarket buy bread and butter, they will also buy milk. From the transactions kept in the supermarkets, an association rule such as "*Bread* and *Butter*→*Milk*" will be mined out. 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 [13]. The theory has been applied in fields such as manufacturing, engineering, diagnosis, economics, among others [4,13,16,24]. Several fuzzy learning algorithms for inducing rules from given sets of data have been designed and used to good effect with specific domains [6–9,12].

As to fuzzy data mining, Hong et al. [10] proposed an algorithm to mine fuzzy rules from quantitative data. They transformed each quantitative item into a fuzzy set and used fuzzy operations to find fuzzy rules. Cai et al. [2] proposed weighted mining to reflect different importance to different items. 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. [25] then extended their concepts to fuzzy item vectors. Besides, Lee et al. [15] proposed a mining algorithm which used multiple minimum supports of different items to mine fuzzy association rules. In the above approaches, the membership functions were assumed to be known in advance. Although many approaches for learning membership functions were proposed [3,18,19,22,23], most of them were usually used for classification or control problems. Wang et al. [21] tuned membership functions for intrusion detection systems based on similarity of association rules. Kaya et al. [14] proposed a GA-based clustering method to derive

T.-P. Hong (✉)
Department of Electrical Engineering,
National University of Kaohsiung
Kaohsiung, 811, Taiwan, R.O.C.
E-mail: tphong@nuk.edu.tw

C.-H. Chen
Department of Computer Science and Information Engineering,
National Cheng-Kung University
Tainan, 701, Taiwan, R.O.C.

Y.-L. Wu
Institute of Information Management,
I-Shou University Kaohsiung, 840, Taiwan, R.O.C.
E-mail: wuyulung@isu.edu.tw

Y.-C. Lee
Institute of Information Engineering,
I-Shou University Kaohsiung, 840, Taiwan, R.O.C.
E-mail: d9003007@stmail.isu.edu.tw

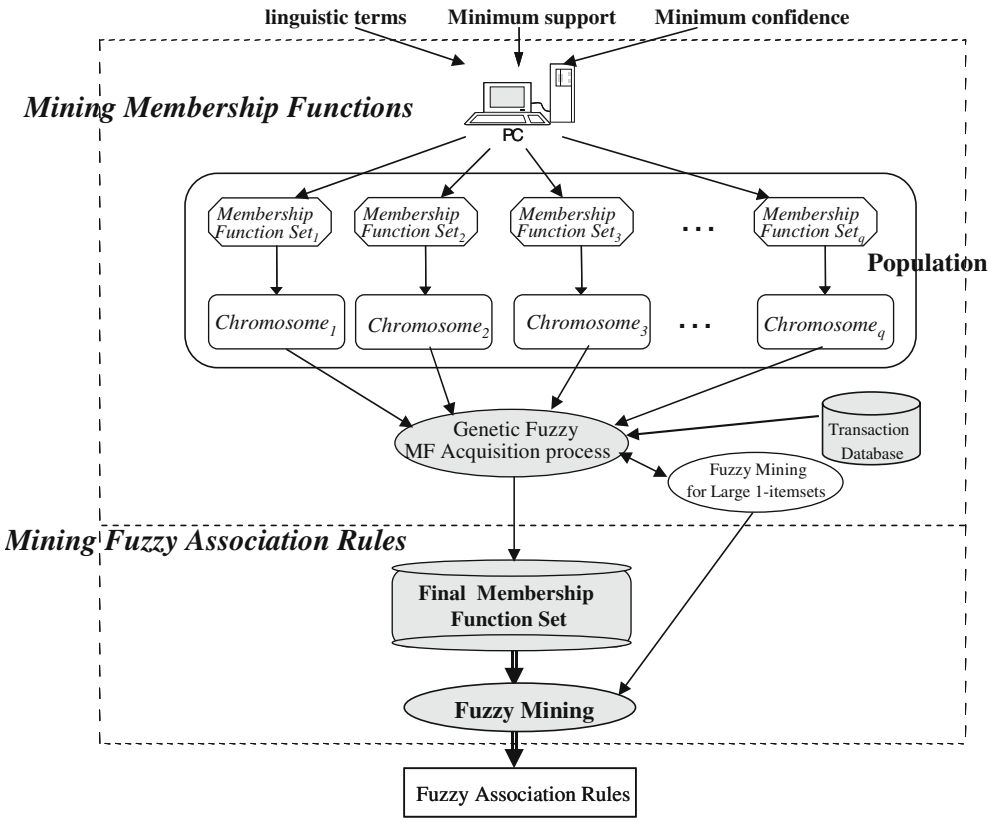

Fig. 1 GA-based framework for searching membership functions

a predefined number of membership functions for getting a maximum profit within an interval of user specified minimum support values.

We have proposed a fuzzy mining algorithm to mine fuzzy rules under a given set of membership functions [11]. The given membership functions may, however, have a critical influence on the final mining results. This paper thus modifies the previous algorithm and proposes a new fuzzy data-mining algorithm for extracting both association rules and membership functions from quantitative transactions. The proposed algorithm can dynamically adapt membership functions by genetic algorithms and uses them to fuzzify the quantitative transactions. Our previous fuzzy mining approach [11] can thus be easily used to find fuzzy association rules. The fitness of each set of membership functions is then evaluated from the mining results and used as the evolutionary criteria in GA. After the GA process terminates, a better set of association rules can then be expected with a more suitable set of membership functions.

The remaining parts of this paper are organized as follows. A GA-based mining framework is stated in Sect. 2. Chromosome representation is described in Sect. 3. The adjustment process of membership functions is explained in Sect. 4. The details of the proposed algorithm for mining both association rules and membership functions are described in Sect. 5. An example to illustrate the proposed algorithm is given in Sect. 6. Experiments to demonstrate the performance of the proposed algorithm are stated in Sect. 7. Conclusions and future works are given in Sect. 8.

2 A GA-based mining framework

In this section, the fuzzy and GA concepts are used to discover both useful association rules and suitable membership functions from quantitative values. We propose a GA-based framework for searching membership functions suitable for mining problems and then use the final best set of membership functions to mine association rules. The proposed framework is shown in Fig. 1.

The proposed framework maintains a population of sets of membership functions, and uses the genetic algorithm to automatically derive the resulting one. It 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.

3 Chromosome representation

It is important to encode membership functions as string representation for GAs to be applied. Several possible encoding

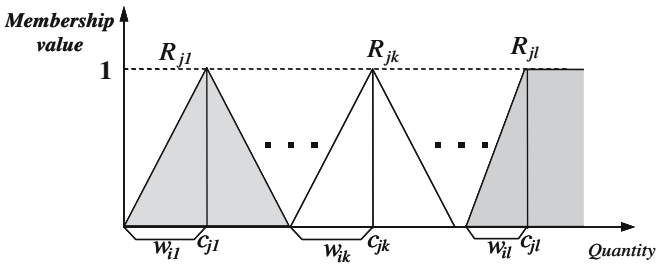

Fig. 2 Membership functions of item I_j

approaches have been described in [3,17,22,23]. 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 [17] did. Membership functions applied to a fuzzy rule set are then assumed to be isosceles-triangle functions as shown in Fig. 2, where R_jk denotes the membership function of the kth 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. Thus the set of membership functions MF_1 for the first item I_1 is then represented as a substring of $c_{11}w_{11}\ldots c_{1|I1|}w_{1|I1|}$, where $|I_1|$ is the number of terms of I_1. The entire set of membership functions is then encoded by concatenating substrings of $\mathrm{MF}_1, \mathrm{MF}_2, ..., \mathrm{MF}_j$. Since c and w are both numeric values, a chromosome is thus encoded as a fixed-length real-number string rather than a bit string. 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. Assume the membership functions for each *item* are given as shown in Fig. 3.

According to the proposed encoding scheme mentioned above, the chromosome for representing the membership functions in Fig. 3 is encoded as shown in Fig. 4.

Since the item *milk* has three possible linguistic terms, *Low*, *Middle* and *High*, the membership functions for *milk* are thus encoded as (5, 5, 10, 5, 15, 5) according to Fig. 2. *Bread* also has three possible linguistic terms, *Low*, *Middle* and *High*, and its associated membership functions are thus encoded as (6, 6, 12, 6, 18, 6). Similarly, the membership functions for *cookies* and *beverage* are, respectively, encoded as (3, 3, 6, 3, 9, 3) and (4, 4, 8, 4, 12, 4).

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.

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

4 Mining membership functions and association rules

4.1 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 in a certain item. The initial set of chromosomes is randomly generated with some constraints of forming feasible membership functions.

4.2 Fitness and selection

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 used 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 fitness of each set of membership functions is formally 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

$$\text{overlap\_factor}(C_{qj}) = \sum_{k\neq i}\left[\max\left(\left(\frac{\text{overlap}(R_{jk}, R_{ji})}{\min(w_{jk}, w_{ji})}\right), 1\right) - 1\right],$$

where $\text{overlap}(R_{jk}, R_{ji})$ 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

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

where $\text{range}(R_{j1}, R_{j2}, \ldots, R_{jl})$ 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.

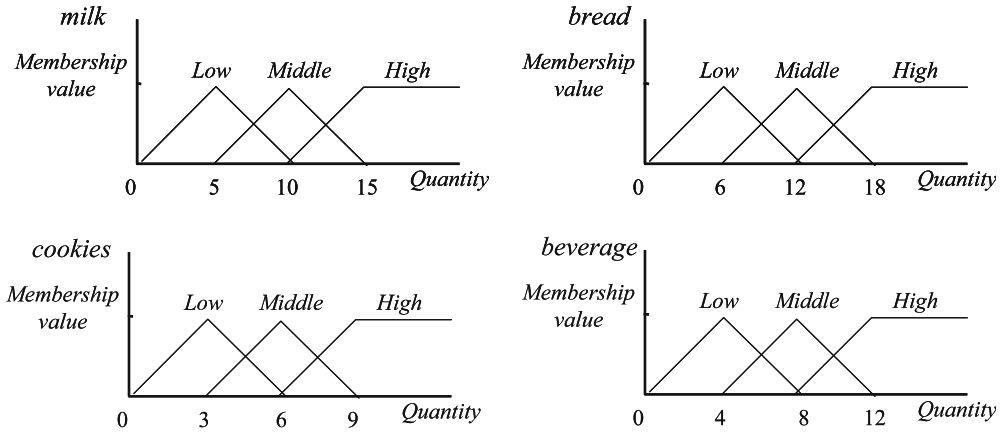

Fig. 3 An example of membership functions for four items

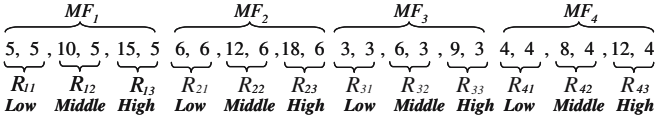

Fig. 4 The chromosome representation for the membership functions in Fig. 3

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

$$\sum_{j=1}^{m} \left[\text{overlap\_factor}(C_{qj}) + \text{coverage\_factor}(C_{qj})\right],$$

where m is the number of items. The fitness value of a chromosome C_q is then defined as

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

where $|L_1|$ is the number of large 1-itemsets obtained by using the set of membership functions in C_q. The suitability factor used in the fitness function 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 suitable(C_q) is designed for avoiding the first bad case, and the coverage factor is for the second one. Below, an example is given to illustrate the above idea.

Example 2 Continue Example 1, assume $\max(I_1) = 13$, $\max(I_2) = 14$, $\max(I_3) = 12$, $\max(I_4) = 7$. The suitability of the chromosome C_1 for item I_1 is computed as follows:

$$\text{Suitability}(C_{11}) = \sum_{k \neq i} \left[\max\left(\left(\frac{\text{overlap}(R_{jk}, R_{ji})}{\min(w_{jk}, w_{ji})} \right), 1 \right) - 1 \right] + \frac{1}{\frac{\text{range}(R_{j1}, R_{j2}, R_{j3})}{\max(I_j)}} = 0 + 0 + 0 + 1 = 1.$$

Similarly, Suitability(C_{12}) = 1, Suitability(C_{13}) = 1, and Suitability(C_{14}) = 1. The suitability of C_1 is then 1 + 1 +1 + 1 = 4.

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.

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 used 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.

In this paper, the fitness of each set of membership functions is evaluated by the number of large 1-itemsets generated by executing part of the previously proposed fuzzy mining algorithm [11]. 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.

4.3 Genetic operators

Genetic operators are very important to the success of specific GA applications. Two genetic operators, the *max–min–arithmetical* (MMA) *crossover* proposed in [5] and the *one-point mutation*, are used in the genetic fuzzy mining framework. Assume there are two parent chromosomes:

$$C_u^t = (c_1, \ldots\ldots, c_h, \ldots\ldots, c_Z),$$

$$C_w^t = (c_1', \ldots\ldots, c_h', \ldots\ldots, c_Z').$$

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

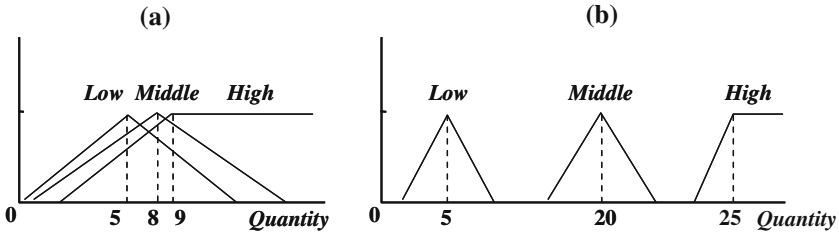

Fig. 5 Two bad membership functions

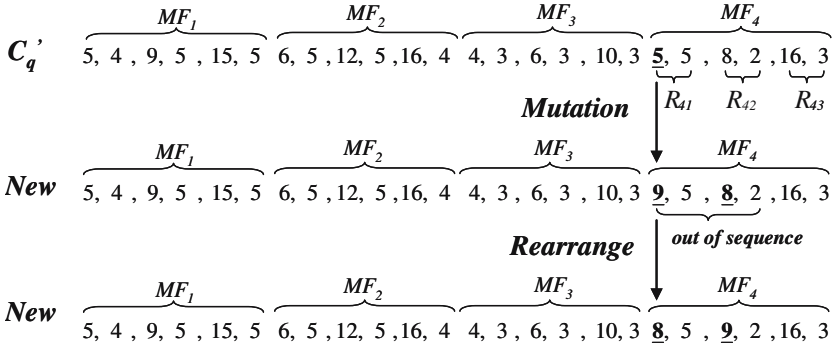

Fig. 6 A mutation operation

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

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. An example is given below to demonstrate the mutation operation.

Example 3 Continuing from Example 1, assume after several generations, we get a new chromosome C_q' as shown in Fig. 6. Also assume the mutation point is set at R_{41} and the random value ε is set at 4. The mutation process is shown in Fig. 6.

5 The proposed mining algorithm

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

The proposed mining algorithm:

INPUT: A body of n quantitative transaction data, a set of m items, each with a number 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: Calculate the fitness value of each chromosome by the following substeps:

STEP 3.1: 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

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

using the corresponding membership functions represented by the chromosome, where R_{jk} is the kth 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 3.2: For each item region R_{jk}, calculate its scalar cardinality on the transactions as follows:

$$\text{count}_{jk} = \sum_{i=1}^{n} f_{jk}^{(i)}.$$

STEP 3.3: 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,

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

STEP 3.4: Set the fitness value of the chromosome as the number of large itemsets in L_1.

STEP 4: Execute crossover operations on the population.

STEP 5: Execute mutation operations on the population.

STEP 6: Using the *selection* criteria to choose individuals for the next generation.

STEP 7: If the termination criterion is not satisfied, go to Step 3; otherwise, do the next step.

STEP 8: Output the set of membership functions with the highest fitness value.

The set of membership functions are then used to mine fuzzy association rules from the given database. Our fuzzy mining algorithm proposed in [11] is then adopted to achieve this purpose.

6 An Example

In this section, an example is given to illustrate the proposed mining algorithm. This is a simple example to show how the proposed algorithm can be used to mine membership functions and fuzzy association rules from data. Assume there are four items in a transaction database: milk, bread, cookies and beverage. The data set includes the six transactions shown in Table 1.

Assume each item has three fuzzy regions: *Low*, *Middle*, and *High*. Thus, three fuzzy membership functions must be derived for each item. For the data shown in Table 1, the proposed mining algorithm proceeds as follows:

STEP 1: P individuals are randomly generated as the initial population. In this example, P is set at 10. Each individual is a set of membership functions for all the four items including milk, bread, cookies, and beverage.

Table 1 Six transactions in this example

TID	Items
T1	(milk, 5); (bread, 10); (cookies, 7); (beverage, 7)
T2	(milk, 7); (bread, 14); (cookies, 12)
T3	(bread, 15); (cookies, 12)
T4	(milk, 2); (bread, 5); (cookies, 5)
T5	(bread, 9)
T6	(milk, 13); (beverage, 12)

STEP 2: Each set of membership functions is encoded into a chromosome according to the representation proposed in Sect. 3. Assume the ten individuals are generated as follows:

C_1: **5, 5, 10, 5, 15, 5**,6, 6, 12, 6, 18, 6, **3, 3, 6, 3, 9, 3**, 4, 4, 8, 4, 12, 4;

C_2: **5, 5, 10, 5, 15, 5**,4, 6, 10, 6, 16, 6, **4, 3, 7, 3, 10, 3**, 4, 4, 8, 4, 12, 4;

C_3: **4, 3, 7, 3, 10, 3**,6, 6, 12, 6, 18, 6, **6, 5, 11, 5, 16, 5**, 6, 4, 10, 4, 14, 4;

C_4: **5, 2, 7, 2, 9, 2**,5, 4, 9, 4, 13, 4, **6, 5, 11, 5, 16, 5**, 6, 4, 10, 4, 14, 4;

C_5: **4, 3, 7, 3, 10, 3**,6, 6, 12, 6, 18, 6, **5, 3, 8, 3, 11, 3**, 3, 4, 7, 4, 11, 4;

C_6: **6, 3, 9, 3, 12, 3**,5, 5, 10, 5, 15, 5, **4, 4, 8, 4, 12, 4**, 6, 4, 10, 4, 14, 4;

C_7: **3, 3, 6, 3, 9, 3**,6, 2, 8, 2, 10, 2, **6, 5, 11, 5, 16, 5**, 4, 4, 8, 4, 12, 4;

C_8: **4, 3, 7, 3, 10, 3**,6, 6, 12, 6, 18, 6, **5, 6, 11, 6, 17, 6**, 6, 4, 10, 4, 14, 4;

C_9: **3, 3, 6, 3, 9, 3**,6, 3, 9, 3, 12, 3, **6, 5, 11, 5, 16, 5**, 6, 2, 8, 2, 10, 2;

C_{10}: **4, 3, 7, 3, 10, 3**,6, 2, 8, 2, 10, 2, **6, 5, 11, 5, 16, 5**, 6, 3, 9, 3, 12, 3.

STEP 3: The fitness value of each chromosome is calculated by the following substeps:

STEP 3.1: The quantitative value of each transaction datum is transformed into a fuzzy set according the membership functions in each chromosome. Take the first item in transaction $T5$ using the membership functions in chromosome C_1 as an example. The membership functions for bread in C_1 are represented as (6, 6, 12, 6, 18, 6), which are shown in Fig. 7.

The amount "9" of item *bread* is then converted into the fuzzy set $\left(\frac{0.5}{bread.Low} + \frac{0.5}{bread.Middle}\right)$ using the membership functions for bread in C_1. The results for all the items are shown in Table 2, where the notation *item.term* is called a fuzzy region.

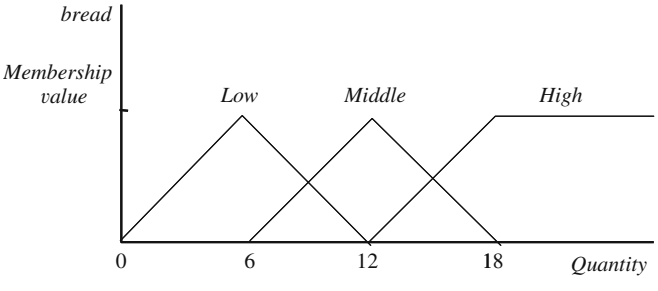

Fig. 7 The membership functions for bread in C_1

Table 2 The fuzzy sets transformed from the data in Table 1

TID	Fuzzy set
T1	$(\frac{1.0}{milk.Low})\,(\frac{0.33}{bread.Low}+\frac{0.67}{bread.Middle})\left(\frac{0.67}{cookies.Middle}+\frac{0.33}{cookies.High}\right)\left(\frac{0.25}{beverage.Low}+\frac{0.75}{beverage.Middle}\right)$
T2	$\left(\frac{0.6}{milk.Low}+\frac{0.4}{milk.Middle}\right)\left(\frac{0.67}{bread.Middle}+\frac{0.33}{bread.High}\right)\left(\frac{1}{cookies.High}\right)$
T3	$\left(\frac{0.5}{bread.Middle}+\frac{0.5}{bread.High}\right)\left(\frac{1}{cookies.High}\right)$
T4	$(\frac{0.4}{milk.Low})\,(\frac{0.83}{bread.Low})\,(\frac{0.33}{cokkies.Low}+\frac{0.67}{cokkies.Middle})$
T5	$\left(\frac{0.5}{bread.Low}+\frac{0.5}{bread.Middle}\right)$
T6	$\left(\frac{0.4}{milk.Middle}+\frac{0.6}{milk.High}\right)\left(\frac{1}{beverage.High}\right)$

STEP 3.2: The scalar cardinality of each fuzzy region in the transactions is calculated as the *count* value. Take the fuzzy region *milk.Low* as an example. Its scalar cardinality = (1.0 + 0.6 + 0.0 + 0.4 + 0.0 + 0.0) = 2.0. The counts for all the fuzzy regions are shown in Table 3.

Table 3 The counts of the fuzzy regions

Item	Count	Item	Count
milk.Low	2.00	*cookies.Low*	0.33
milk.Middle	0.80	*cookies.Middle*	1.33
milk.High	0.60	*cookies.High*	2.33
bread.Low	1.67	*beverage.Low*	0.25
bread.Middle	2.33	*beverage.Middle*	0.75
bread.High	0.83	*beverage.High*	1.00

STEP 3.3: The count of any fuzzy region is checked against the predefined minimum support value α. Assume in this example, α is set at 2.0. Since all the count values of *milk.Low*, *bread.Middle* and *cookies.High* are larger than 2.0, these items are put in L_1 (Table 4).

Table 4 The set of large 1-itemsets (L_1) in this example

Itemset	Count
milk.Low	2.0
bread.Middle	2.33
cookies.High	2.33

STEP 3.4: Since there are three large 1-itemsets for the membership functions of C_1 and its suitability is calculated as 4, the fitness value of C_1 is thus 3/4 (= 0.75). The fitness values of all the chromosomes are shown in Table 5.

Table 5 The fitness values of the chromosomes in the initial population

Chromosome	f	Chromosome	f
C_1	0.75	C_6	0.54
C_2	0.53	C_7	0.32
C_3	0.27	C_8	0.54
C_4	0.3	C_9	0.31
C_5	0.58	C_{10}	0.32

STEP 4: Execute crossover operators on the population. Assume d is set at 0.35. Taking the crossover of C_1 and C_2 as an example. The following four candidate offspring chromosomes are generated as follows:

C$_1$: **5**, **5**, **10**, **5**, **15**, **5**,6, 6, 12, 6, 18, 6, **3**, **3**, **6**, **3**, **9**, **3**,4, 4, 8, 4, 12, 4

C$_2$: **5**, **5**, **10**, **5**, **15**, **5**,4, 6, 10, 6, 16, 6, **4**, **3**, **7**, **3**, **10**, **3**,4, 4, 8, 4, 12, 4

1) C_1^{t+1}: **5**, **5**, **10**, **5**, **15**, **5**,4.7, 6, 10.7, 6, 16.7, 6, **3.65**, **3**, **6.65**, **3**, **9.65**, **3**,4, 4, 8, 4, 12, 4

2) C_2^{t+1}: **5**, **5**, **10**, **5**, **15**, **5**,5.3, 6, 11.3, 6, 17.3, 6, **3.35**, **3**, **6.35**, **3**, **9.35**, **3**,4, 4, 8, 4, 12, 4

3) C_3^{t+1}: **5**, **5**, **10**, **5**, **15**, **5**,4, 6, 10, 6, 16, 6, **3**, **3**, **6**, **3**, **9**, **3**,4, 4, 8, 4, 12, 4

4) C_4^{t+1}: **5**, **5**, **10**, **5**, **15**, **5**,6, 6, 12, 6, 18, 6, **4**, **3**, **7**, **3**, **10**, **3**,4, 4, 8, 4, 12, 4

The fitness value of the above four candidates are then evaluated, with results shown in Table 6.

Table 6 The fitness value of the four candidate offspring

Chromosome	f	Chromosome	f
C_1^{t+1}	0.8	C_3^{t+1}	0.8
C_2^{t+1}	0.8	C_4^{t+1}	0.8

Since all the four chromosomes have the same fitness value, any two of them can be chosen. Here assume C_1^{t+1} and C_3^{t+1} are chosen.

STEP 5: The *mutation operator* is executed to generate possible offspring. The operation is the same as the traditional one except that rearrangement may need to be done.

STEPs 6–8: The best ten chromosomes are selected as the next generation. The same procedure is then executed until the termination criterion is satisfied. The best chromosome (with the highest fitness value) is output as the membership functions for deriving fuzzy rules.

After the membership functions are derived, the fuzzy mining method proposed in [11] is then used to mine fuzzy association rules.

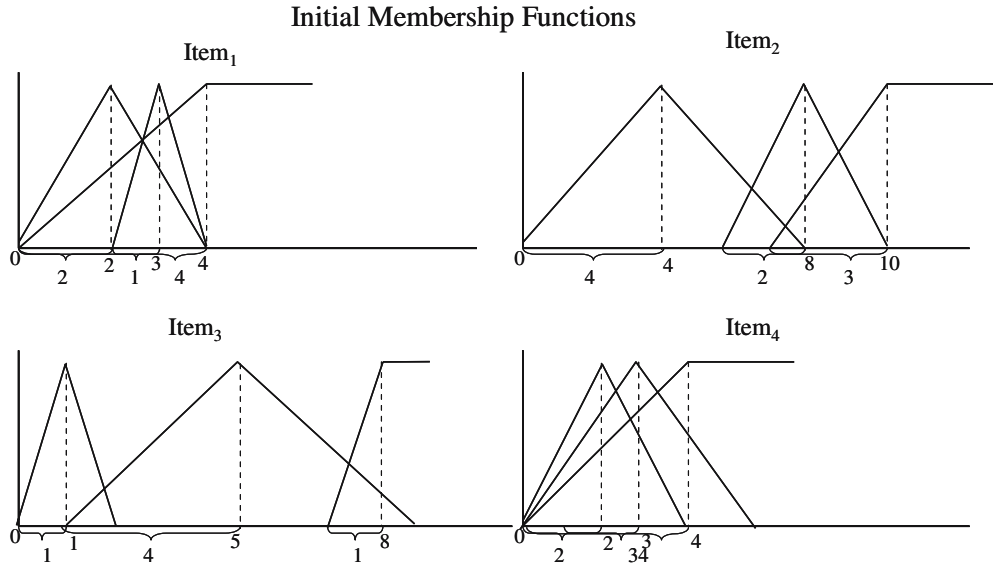

Fig. 8 The initial membership functions of some four items

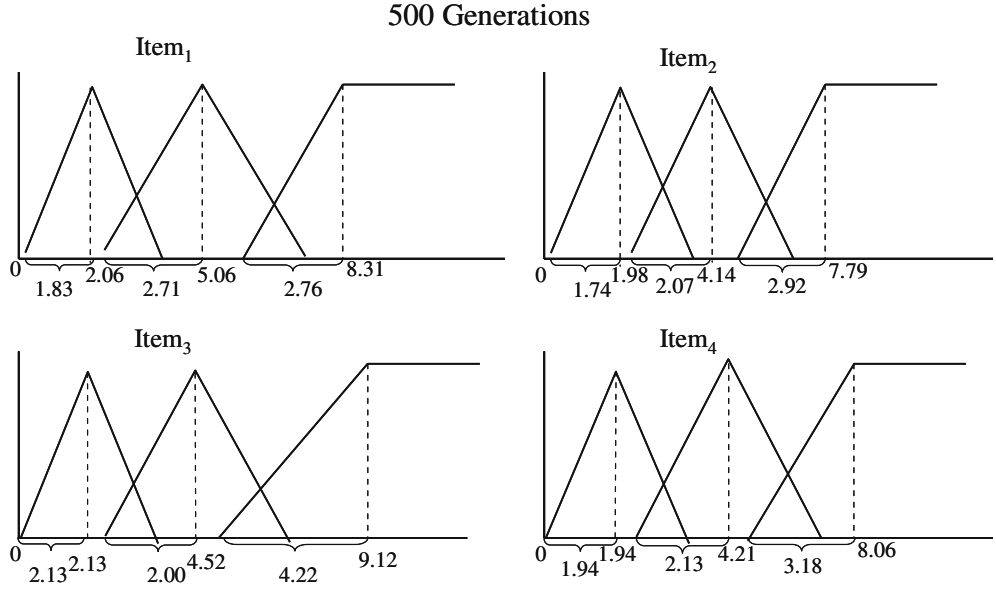

Fig. 9 The final membership functions of some four items after 500 generations

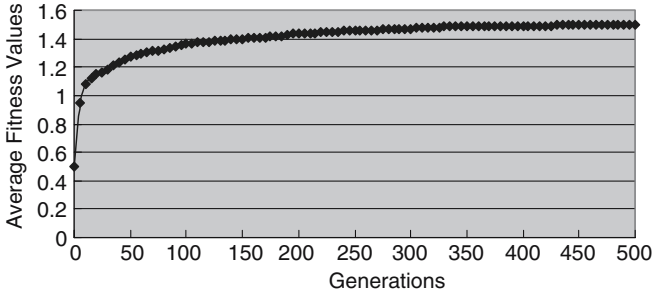

Fig. 10 The average fitness values along with different numbers of generations

7 Experiment 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 2.00 GHz and 256 MB RAM. A total of 64 items and 10,000 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 is set at 50, the crossover rate p_c is set at 0.8, and the

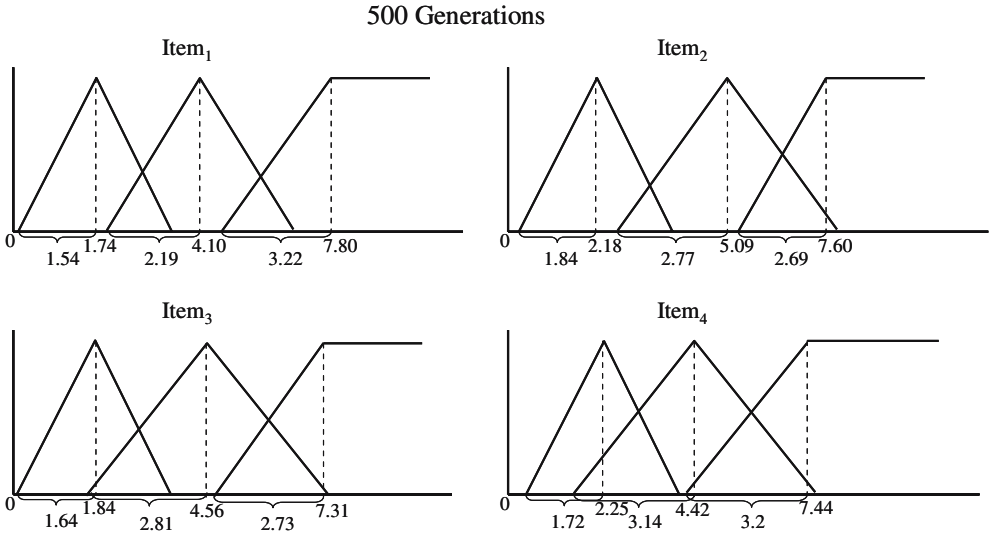

Fig. 11 The final membership functions when only the suitability is considered

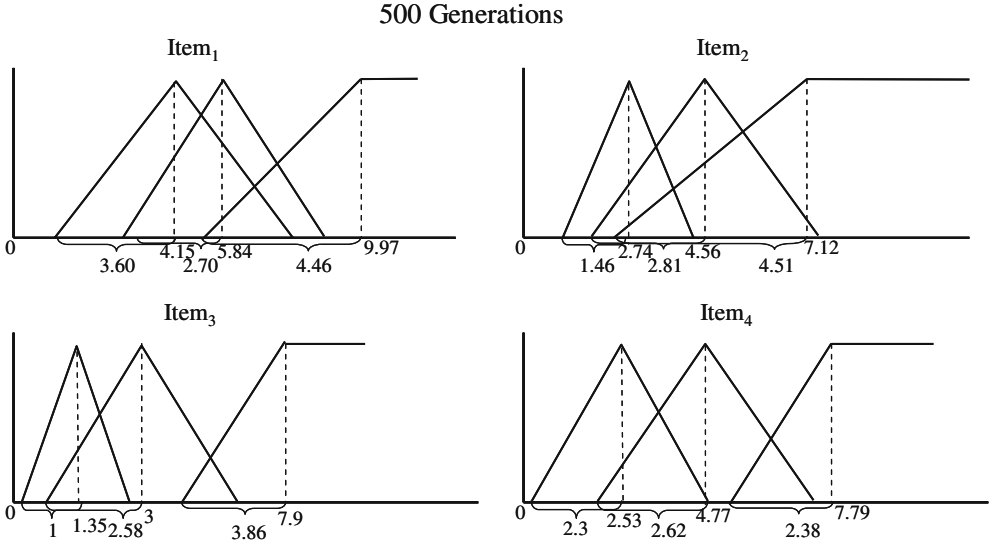

Fig. 12 The final membership functions when only $|L_1|$ is considered

mutation rate p_m is set at 0.01 [20]. The parameter d of the crossover operator is set at 0.35 according to [5] and the minimum support α is set at 0.04 (4%). The data generated are put in the website "http://www.nuk.edu.tw/tphong/data/".

After 500 generations, the final membership functions are 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. 8.

In Fig. 8, the membership functions have the two bad types of shapes according to the definition in the previous section. The membership functions for $Item_1$, $Item_2$, and $Item_4$ overlap too much. After 500 generations, the final membership functions for the same four items are shown in Fig. 9.

It is easily seen that the membership functions in Fig. 9 is better than those in Fig. 8. The two bad kinds of membership functions did not appear in the final results.

The average fitness values of the chromosomes along with different numbers of generations are shown in Fig. 10. As expected, the curve gradually goes upward, finally converging to a certain value.

Next, experiments were made by using only suitability(C_q) and only $|L_1|$ as the fitness functions to show the validity of the proposed one. For the same experimental environments and data, the membership functions of the above four items after 500 generations by using only suitability(C_q) as the fitness function are shown in Fig. 11, and by using only the number of large 1-itemsets are shown in Fig. 12.

It can be easily seen from Fig. 11 that the derived membership functions by considering only suitability are satisfactory because the suitability measure is designed for getting good shapes of membership functions. Its number of large 1-itemsets is, however, less than the original one (which will be shown later). On the contrary, it is very natural for the

derived membership functions by considering only the number of large 1-itemsets to have a bad shape from Fig. 12. Their overlap degrees are quite high.

The numbers of large 1-itemsets by the original fitness function, by only the suitability measure, and by only the number of large 1-itemsets, along with different generations are further compared, with results shown in Fig. 13.

It can be easily seen from Fig. 13 that the number of large 1-itemsets by only the suitability is the least among the three fitness functions. The suitability values by the three fitness functions along with different generations are then shown in Fig. 14.

From Fig. 14, the suitability by the original fitness function is even better than that by only the suitability measure. It is because the solutions by considering only suitability may be more easily trapped into local optimality than those by the original fitness functions. Our proposed fitness function can thus achieve a good trade-off between numbers of large itemsets and suitability of membership functions.

Next, experiments were made for providing a comparative analysis of the proposed approach with the fuzzy mining approach in [11] with uniform fuzzy partition. Since the range of the dataset in our experiments fell in the interval 1–12, the membership functions shown in Fig. 15 were thus used for uniform fuzzy partition.

The relationship between the numbers of large 1-itemsets and the minimum supports for these two approaches is shown in Fig. 16.

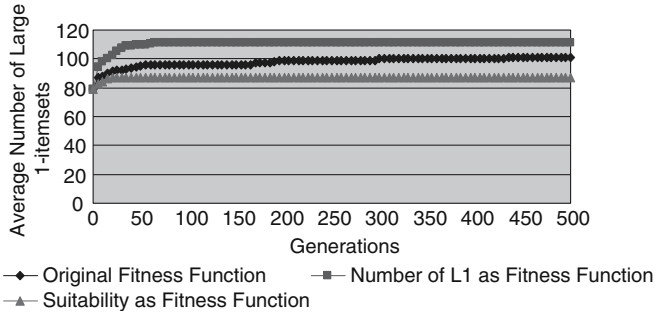

Fig. 13 The average number of large 1-itemsets by the three different fitness functions

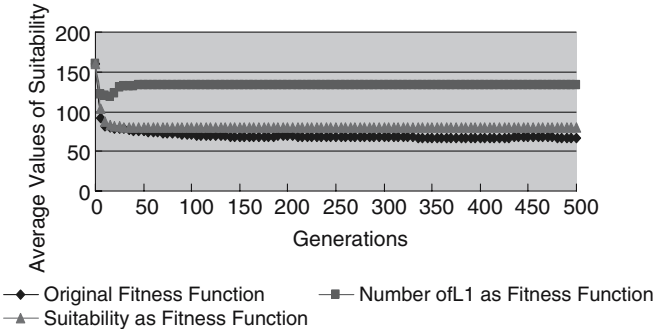

Fig. 14 The average value of suitability by the three different fitness functions

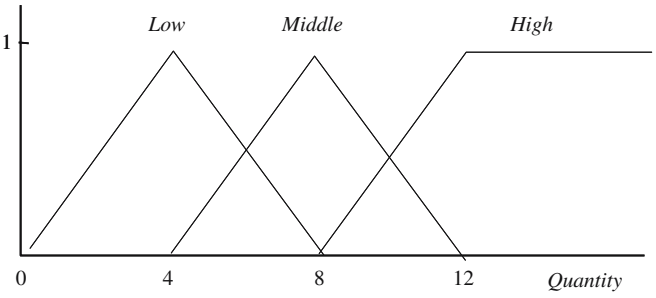

Fig. 15 The membership functions used for uniform fuzzy partition

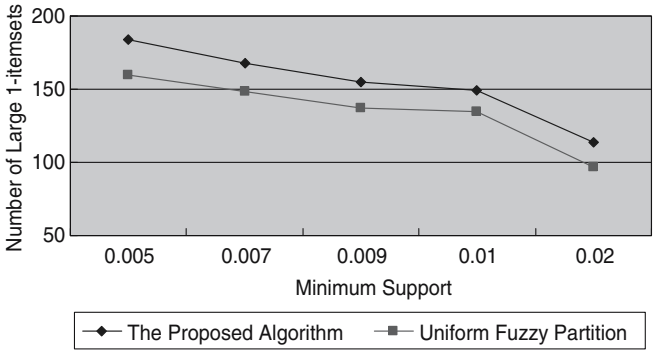

Fig. 16 The relationship between the numbers of large 1-itemsets and the minimum supports for the two approaches

It can be observed from Fig. 16 that the number of large 1-itemsets derived by the proposed algorithm was larger than the one with uniform fuzzy partition. It is consistent with the previous discussion since the fitness function used will help the proposed approach search for a larger number of large 1-itemsets.

Experiments were then made to provide an analysis of the association rules via supports and confidences. Fig. 17 shows the relationship between the numbers of association rules derived by the membership functions found in the first phase and the minimum supports along with different minimum confidences.

It can be observed from Fig. 17 that the number of rules decreased along with the increase of the minimum support

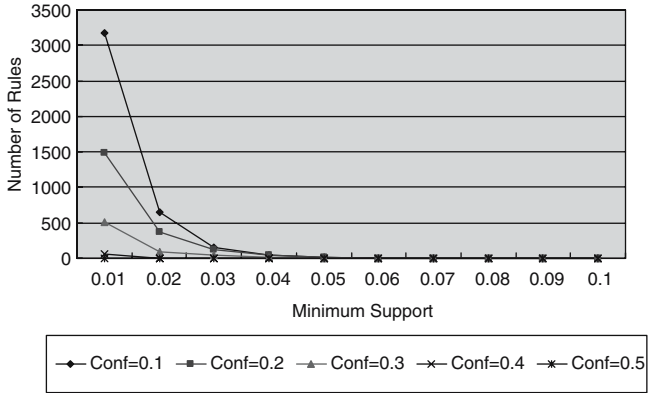

Fig. 17 The relationship between the numbers of association rules and the minimum supports along with different minimum confidences

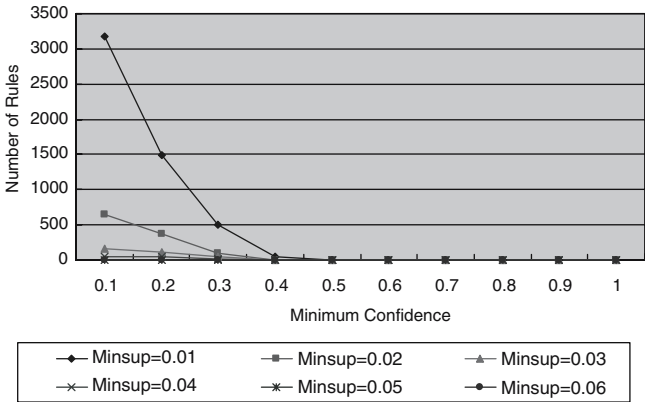

Fig. 18 The relationship between the numbers of association rules and the minimum confidences along with different minimum supports

values. Besides, the curve with a large minimum confidence value was smoother than those with a small value. It means that the effect is not apparent for a large minimum confidence since most of the association rules cannot satisfy the confidence condition.

The relationship between the numbers of association rules derived by the membership functions found in the first phase and the minimum confidences along with different minimum supports is shown in Fig. 18.

It can be observed from Fig. 18 that the number of association rules decreased along with the increase of the minimum confidence values. Besides, the curve with a large minimum support value was smoother than those with a small value, meaning that the minimum confidence value had a larger effect on the number of association rules when smaller minimum support values were used. Note that all of the curves approximated to 0 as the minimum confidence value approached 1.

8 Conclusion and future works

In this paper, we have proposed a fuzzy data-mining algorithm for extracting both association rules and membership functions from quantitative transactions. The proposed algorithm can dynamically adjust membership functions by genetic algorithms and uses them to fuzzify the quantitative transactions. Using the number of large 1-itemsets in the fitness function can achieve a trade-off between execution time and rule interestingness. The experimental results show that the designed fitness function is effective to avoiding the two bad kinds of membership functions in the mining process and to providing the most information to users. In the future, we will continuously attempt to enhance the GA-based mining framework for more complex problems.

Acknowledgements The authors would like to thank the anonymous reviewers for their constructive comments and thank Mr. Chien-Shing 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. Agrawal R, Srikant R (1994) Fast algorithm for mining association rules. In: The international conference on very large databases, pp 487–499
2. Cai CH, Fu WC, Cheng CH, Kwong WW (1998) Mining association rules with weighted items. In: The international database engineering and applications symposium, pp 68–77
3. Cordón O, Herrera F, Villar P (2001) Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. IEEE Trans Fuzzy Syst 9(4):667–674
4. Graham I, Jones PL (1998) Expert Systems – Knowledge, Uncertainty and Decision, Chapman and Computing, Boston, pp 117–158
5. Herrera F, Lozano M, Verdegay JL (1997) Fuzzy connectives based crossover operators to model genetic algorithms population diversity. Fuzzy Sets Syst 92(1):21–30
6. Hong TP, Chen JB (1999) Finding relevant attributes and membership functions. Fuzzy Sets Syst 103(3):389–404
7. Hong TP, Chen JB (2000) Processing individual fuzzy attributes for fuzzy rule induction. Fuzzy Sets Syst 112(1):127–140
8. Hong TP, Lee CY (1996) Induction of fuzzy rules and membership functions from training examples. Fuzzy Sets Syst 84:33–47
9. Hong TP, Tseng SS (1997) A generalized version space learning algorithm for noisy and uncertain data. IEEE Trans Knowledge Data Eng 9(2):336–340
10. Hong TP, Kuo CS, Chi SC (1999) Mining association rules from quantitative data. Intelligent Data Analysis 3(5):363–376
11. Hong TP, Kuo CS, Chi SC (2001) Trade-off between time complexity and number of rules for fuzzy mining from quantitative data. Int J Uncertain Fuzziness Knowledge-Based Syst 9(5): 587–604
12. Hou RH, Hong TP, Tseng SS, Kuo SY (1997) A new probabilistic induction method. J Autom Reason 18:5–24
13. Kandel A (1992) Fuzzy expert systems. CRC Press, Boca Raton, pp 8–19
14. Kaya M, Alhajj R (2003) A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining. The IEEE international conference on fuzzy systems, pp 881–886
15. Lee YC, Hong TP, Lin WY (2004) Mining fuzzy association rules with multiple minimum supports using maximum constraints. Lecture Notes Comput Sci 3214:1283–1290
16. Mamdani EH (1974) Applications of fuzzy algorithms for control of simple dynamic plants. In: IEEE proceedings, pp 1585–1588
17. Parodi A, Bonelli P (1993) A new approach of fuzzy classifier systems. In: Proceedings of fifth international conference on genetic algorithms. Morgan Kaufmann, Los Altos, pp 223–230
18. Roubos H, Setnes M (2001) Compact and transparent fuzzy models and classifiers through iterative complexity reduction. IEEE Trans Fuzzy Syst 9(4):516–524
19. Setnes M, Roubos H (2000) GA-fuzzy modeling and classification: complexity and performance. IEEE Trans Fuzzy Syst 8(5):509–522
20. Srinivas M, Patnaik LM (1994) Genetic algorithms: a survey. Computer 27(6):17–26
21. Wang W, Bridges SM (2000) Genetic algorithm optimization of membership functions for mining fuzzy association rules. The international joint conference on information systems, fuzzy theory and technology, pp 131–134
22. Wang CH, Hong TP, Tseng SS (2000) Integrating fuzzy knowledge by genetic algorithms. IEEE Trans Evol Comput 2(4):138–149
23. Wang CH, Hong TP, Tseng SS (2000) Integrating membership functions and fuzzy rule sets from multiple knowledge sources. Fuzzy Sets Syst 112:141–154
24. Weber R (1992) Fuzzy-ID3: a class of methods for automatic knowledge acquisition. The second international conference on fuzzy logic and neural networks, Iizuka, Japan, pp 265–268
25. Yue S, Tsang E, Yeung D, Shi D (2000) Mining fuzzy association rules with weighted items. The IEEE International Conference on Systems, Man and Cybernetics, pp 1906–1911