Soft Comput (2006) 10: 230–237
DOI 10.1007/s00500-005-0476-x

ORIGINAL PAPER

Bilal Alataş · Erhan Akin

An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules

Published online: 27 April 2005

Abstract In this paper, a genetic algorithm (GA) is proposed as a search strategy for not only positive but also negative quantitative association rule (AR) mining within databases. Contrary to the methods used as usual, ARs are directly mined without generating frequent itemsets. The proposed GA performs a database-independent approach that does not rely upon the minimum support and the minimum confidence thresholds that are hard to determine for each database. Instead of randomly generated initial population, uniform population that forces the initial population to be not far away from the solutions and distributes it in the feasible region uniformly is used. An adaptive mutation probability, a new operator called uniform operator that ensures the genetic diversity, and an efficient adjusted fitness function are used for mining all interesting ARs from the last population in only single run of GA. The efficiency of the proposed GA is validated upon synthetic and real databases.

Keywords Data mining · Quantitative association rules · Negative association rules · Genetic algorithm

1 Introduction

Data mining is the extraction of implicit, valid, and potentially useful knowledge from large volumes of raw data [1]. The extracted knowledge must be not only accurate but also readable, comprehensible and ease of understanding.

Association rule (AR) mining is one of the important research problems in data mining field where the goal is to derive multi-feature (attribute) correlations from databases. In this study, the goal is to find positive and negative ARs in quantitative databases without the necessity of previously preparing the data. Itemsets are considered within the antecedent or the consequent being negated. Then the rules to be considered are of the form:

$$B \in \neg[b_1, b_2] \;\Rightarrow\; C \in [c_1, c_2] \wedge D \in [d_1, d_2]$$
$$A \in \neg[a_1, a_2] \wedge B \in [b_1, b_2] \;\Rightarrow\; D \in \neg[d_1, d_2]$$
$$A \in [a_1, a_2] \wedge C \in [c_1, c_2] \;\Rightarrow\; B \in \neg[a_1, a_2] \wedge D \in [d_1, d_2]$$

Here A, B, C, and D show the quantitative attributes, $\neg[l_1, l_2]$ shows the intervals not in $[l_1, l_2]$, and, l_1 and l_2 show the lower and upper bound of the interval.

In order to get this objective, an efficient genetic algorithm (GA) has been designed to simultaneously search for intervals of quantitative attributes and the discovery of ARs that these intervals conform in only single run. Furthermore, a database-independent approach that does not rely upon the minimum support and the minimum confidence thresholds that are hard to determine for each database has been performed. Contrary to the methods used as usual, ARs have directly been mined without generating frequent itemsets. Uniform population (UP) for initial population [2]; adaptive mutation, a new operator called uniform operator (UO), and an adjusted fitness function that will be described in Sect. 3 have been used for effectively mining all interesting positive and negative quantitative association rules in the last population of GA.

This paper is organized as follows. Section 2 describes the quantitative and negative ARs with related works. Section 3 explains the details of the algorithm. Section 4 briefly describes the used databases and discusses the experimental results. Finally Sect. 5 concludes the paper.

2 Quantitative and negative ARs

The Boolean AR mining problem over basket data was introduced in [3]. In this and many algorithms proposed afterwards for mining the ARs are divided into two stages: the first is to find the frequent itemsets; the second is to use the frequent itemsets to generate ARs. The mined rules have certain support and confidence values. Though Boolean ARs are

B. Alataş (✉) · E. Akin
Department of Computer Engineering, Faculty of Engineering,
Fırat University, 23119 Elazğ, Turkey
E-mail: balatas@firat.edu.tr; eakin@firat.edu.tr
Tel.:+90-424-2370000
Fax : +90-424-2415526

meaningful, there are many other situations where data items concerned are usually categorical or quantitative. That is why, quantitative AR mining algorithms have been proposed in [4] by first partitioning the attributes domains into small intervals and combining adjacent intervals into larger one such that the combined intervals will have enough supports. In fact, quantitative problem has been transformed to a Boolean one. Some researchers used geometric means to find numeric intervals for quantitative values [5]. The found association rules had no more than two numeric variables in the antecedent and a single Boolean variable in the consequent. Aumann and Lindell used the distribution of a numerical value as the criteria for inclusion in the association rule [6]. Their contention was that an association rule could be thought of as a population subset (the rule consequent) exhibiting some interesting behavior (the rule antecedent). They investigated two types of quantitative rules: categorical⇒quantitative rules and quantitative⇒quantitative rules.

Limitations on these quantitative association rule algorithms, in general, were the numbers of variables allowed in either the consequent or antecedent of the rules. In addition, it is not allowed for both Boolean and multiple quantitative values to be in both the consequent and/or in the antecedent of the rule.

Diverse researchers afterwards have used clustering techniques, partitioning by means of fuzzy sets, however all of them have in common the fact that they need information a priori from the user.

The main problem of all these approaches is preparation of the data before applying the algorithm. This preparation, either by means of the user or by means of an automatic process, conveys a loss of information because the rules will only be generated departing from the partitions previously created. Furthermore, except fuzzy sets these approaches have some drawbacks. The first problem is caused by sharp boundary between intervals that is not intuitive with respect to human perception. The algorithms either ignore or over-emphasize the elements near the boundary of the intervals. Furthermore, distinguishing the degree of membership for the interval method is not easy. The idea of using GA for mining only positive frequent sets was first applied in [7]. However, the encoding used in this work is not much effective for genetic operators to be performed because of variable size. Furthermore, only positive frequent itemsets were mined by running the GA as many times as frequent itemsets that have been wanted to obtain and this has a big computational cost.

Negative ARs are ARs between the antecedent and consequent of the rule. Either the antecedent or consequent or both have to be negated in order for the rule to be a negative rule. There has been no work for mining negative quantitative ARs, however these rules further complete associated relationships among attributes as a system in science and technology and, they offer more information that might be of use in supporting decisions for applications. Furthermore, sometimes intervals for attributes that conform rules may be in $[L, x_1] \cup [x_2, U]$ where $x_2 > x_1$, L is the lower bound and U is upper bound of the attribute as shown in gray colored area of Fig. 1.

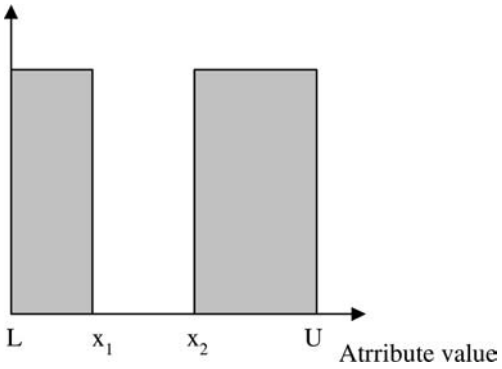

Fig. 1 Negative interval

3 The proposed GA

In this work, a high-quality population, UP, in the beginning has been generated and then the search of positive and negative ARs has been started. The chromosomes of initial population are far away from each others and dispersed the feasible region uniformly. For mining all rules in a single GA run, efficient techniques have been used. The flowchart of the proposed GA is shown in Fig. 2. In each generation of GA, two fittest chromosomes are selected for new operator called UO and two genetic operators, crossover and mutation, are performed to stochastically selected chromosomes for evolution. In each iteration, the number of new chromosomes is equal to the population size. Thus, the number of chromosomes in the population is doubled. At this stage, the rules are evaluated and adjusted fitness value is computed. One half of the high-quality chromosomes are retained and passed to the next iteration. The whole process is performed iteratively until the maximum number of generations is reached. The main characteristics of this algorithm have been described in the following subsections.

3.1 Encoding

In this work, the chromosomes that are being produced and modified along the genetic process represent rules. Each chromosome consists of genes that represent the items and intervals. A positional encoding, where the i-th item is encoded in the i-th gene has been used. Each gene has four parts. The first part of each gene represents the antecedent or consequent of the rule and can take three values: '0', '1' or '*'. If the first part of the gene is '0', it means that this item will be in the antecedent of the rule and if it is '1', this item will be in the consequent of the rule. If it is '*', it means that this item will not be involved in the rule. All genes that have '0' on their first parts will form the antecedent of the rule while genes that have '1' on their first part will form the consequent of the rule. Second part of the gene represents the positive or negative ARs. This part can take two values: '0' or '1'. If the second part of the gene is '0', this means that the interval of this item will be negated in the rule, that is, it forms a negative rule. If it is '1' it will be used for mining positive ARs. While the third part represents the lower bound, the

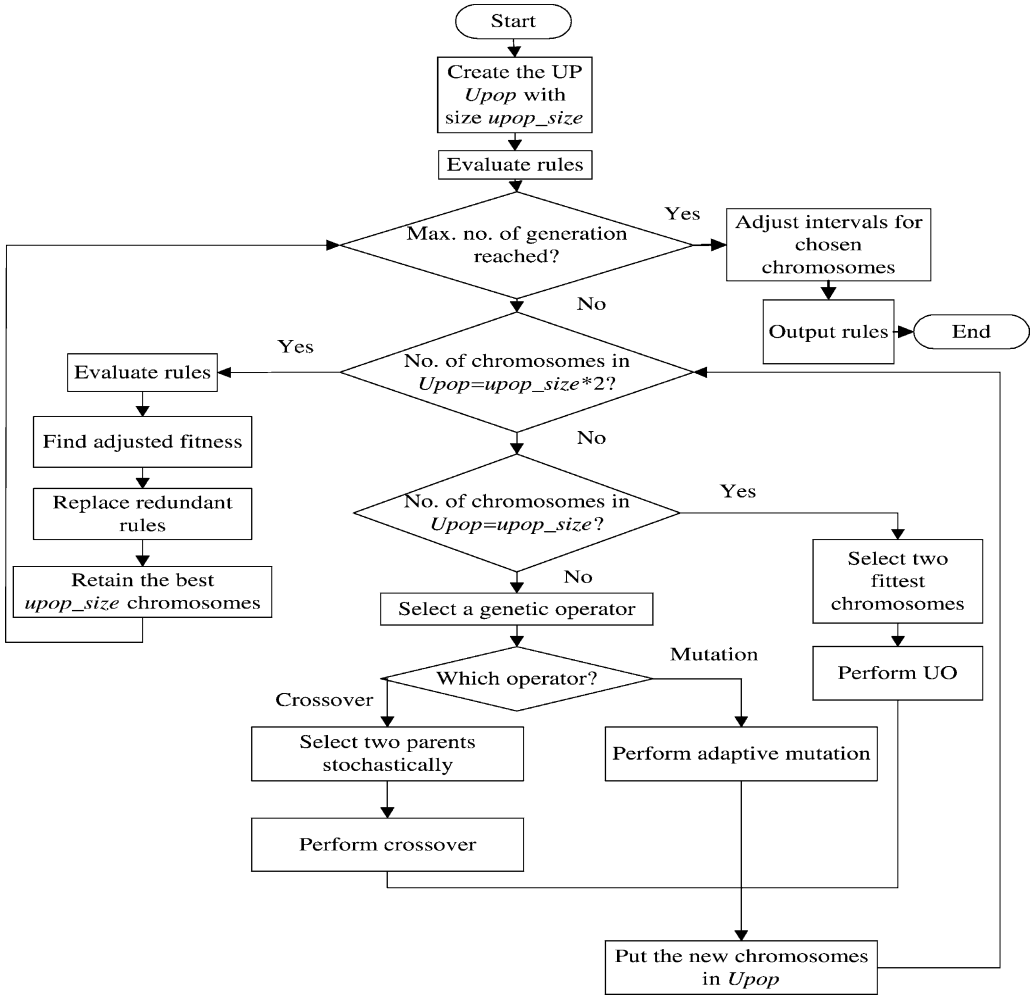

Fig. 2 The flowchart of the proposed GA

$Gene_1$				$Gene_2$				...			$Gene_m$			
AC_1	PN_1	LB_1	UB_1	AC_2	PN_2	LB_2	UB_2				AC_m	PN_m	LB_m	UB_m

Fig. 3 Chromosome representation

fourth part represents the upper bound of the item interval. The structure of a chromosome has been illustrated in Fig. 3, where m is the number of attributes of data being mined.

3.2 Initial population

All genetic solutions for any search problem have been done by means of creating initial population randomly. However, this kind of method has some drawbacks. Initial population may be created in the infeasible region, or all chromosomes in population may be in the nearest neighborhood and far to solution, or search of solution may get a local solution and this local solution can not be get rid of. In this work, UP method was used to create initial population avoiding the drawbacks of random initial population method.

For binary encoding, let $x = (x_1, x_2, \ldots, x_n)$ be a row vector (chromosome) and $x_i \in \{0, 1\}, 1 \leqslant i \leqslant n$. There is a parameter, r, for this method. If $r = 1$, then initially, a chromosome is randomly created and then, inversion of this chromosome is also selected as another chromosome. If $r = 2$, then randomly created chromosome is divided into two equal parts: First, the inversion of the first part is taken to yield new chromosome. Taking inversion of the second part will yield another new chromosome, and inversion of all genes of randomly generated chromosome is also another chromosome. Therefore, three extra chromosomes are derived from randomly created chromosome. For example, a population of size $4 \times p$ is created from p randomly created chromosomes (p is a positive integer) in case of $r = 2$. If x is a randomly created chromosome, then the derived chromosomes from x for $r = 2$ are shown in Fig. 4.

If there are r-dividing points, then the number of derived chromosomes from randomly generated chromosome is $2^r - 1$. Thus, the number of chromosomes in the initial

$\vec{x} = (x_1, x_2, ..., x_n)$ $\qquad \vec{y} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)$
$\vec{z} = (\bar{x}_1, \bar{x}_2, ...\bar{x}_{\lfloor \frac{n}{2} \rfloor}, x_{\lfloor \frac{n}{2} \rfloor + 1}, ..., x_n)$
$\vec{t} = (x_1, x_2, ..., x_{\lfloor \frac{n}{2} \rfloor}, \bar{x}_{\lfloor \frac{n}{2} \rfloor + 1}, ..., \bar{x}_n)$

Fig. 4 Derived chromosomes

population will be

$$(2^r - 1) \times p + p = p \times 2^r \tag{1}$$

Here, p is the number of randomly generated chromosomes.

Inversion of the parts of the genes can be performed by the rules shown in Table 1. "P" means the parts of the genes and "V" means the possible values of these parts. $k \in [0, 1]$ is a random number. "L" is the lower bound and "U" is the upper bound of the relevant interval of attribute. LB_m and UB_m are the current values of the genes. From this table, for example, it can be seen that inversion of "0" is "1", inversion of "1" is "*", and inversion of "*" is "0" for AC part of the gene.

3.3 Fitness function

The mined rules have to acquire large support and confidence. GA has been designed to find the intervals in each of the attributes that conform an interesting rule, in such a way that the fitness function itself is the one that decides the amplitude of the intervals. That is why, the fitness value has to appropriately shelter these and it has been shown in (2)

$$\begin{aligned}\text{Fitness} = {} & \alpha_1 \times \text{cover (Ant+Cons)} + \alpha_2 \\ & \times \frac{\text{cover(Ant+Cons)}}{\text{cover(Ant)}} - \alpha_3 \times (NA - N(*)) \\ & -\alpha_4 \times \text{Int}\end{aligned} \tag{2}$$

This fitness function has four parts. Here, Ant and Cons are distinct itemsets that are involved in the antecedent and consequent part of the rule respectively. cover (Ant+Cons) is ratio of the records that contain Ant+Cons to the total number of records in the database. The first part can be considered as support of the rule that is statistical significance of an AR. In fact, the second part can be considered as confidence value. The third part is used for number of attributes in the chromosome. This parameter rewards the shorter rules with a smaller number of attributes. NA is number of attributes in the database and $N(*)$ is number of attributes that has '*' in first parts of genes of chromosomes. The motivation behind this term is to bias the system towards slightly shorter rules. By this term, readability, comprehensibility, and ease of understanding that are important in data mining are increased. It is known that larger rules are more likely to contain redundant or unimportant information, and this can obscure the basic components that make the rule successful and efficiently processable. The last part of the fitness is used to penalize the amplitude of the intervals that conform the itemset and rule. In this way, between two chromosomes that cover the same number of records and have the same number of attributes, the one whose intervals are smaller gives the best information. Int has been computed in different ways for positive and negative ARs as shown in (3) where amp_m is the amplitude factor determined for each attribute for balancing the effect of Int to the fitness.

$$\text{Int} = \begin{cases} \frac{UB_m - LB_m}{\text{amp}_m} & \text{if second part of the gene is 1} \\ \frac{(UB - LB - (UB_m - LB_m))}{\text{amp}_m} & \text{otherwise} \end{cases} \tag{3}$$

α_1, α_2, α_3, and α_4 are user specified parameters and one might increase or decrease the effects of parts of fitness function by means of these parameters.

For finding all interesting rules in the last generation of GA, an adjusted fitness value has been computed after genetic operators described in Sect. 3.4 have been performed. In the computation of adjusted fitness, first, attributes of the records covered by the best rules before genetic operators are marked as A or C where A represents this attribute of the record is covered by antecedent while C represents this is covered by consequent of the rule. Then, chromosomes that cover the attributes of the records covered by the obtained best rules are penalized. This adjusted fitness factor affects negatively to the fitness and indicates that attributes of the record have been covered previously by the best rules. It has been achieved with that the algorithm tends to discover different rules in later generations. To penalize the chromosomes, a penalization factor (α_5) is used. Thus, the fitness value is computed as

$$\text{Adjusted Fitness} = \text{Fitness} - \alpha_5 \times \sum_{i=1}^{N} \text{marked}_i \tag{4}$$

where N is the number of the records. Here, the value marked_i that one chromosome will take for one record is:

$$\text{marked}_i = \begin{cases} 1 & \text{if } A_j = A \text{ and } C_j = C,\ A_j \in \text{Ant and } C_j \in \text{ Cons} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

marked_i is a binary value that indicates if the record has already been covered by the best rule. If the marked attributes of one record covered by both antecedent and consequent of one chromosome, the fitness will be decreased.

Table 1 Inversion rules

Part of genes	*AC*			*PN*		*LB*	*UB*
Values	0	1	*	0	1	LB_m	UB_m
Inversion of values	1	*	0	1	0	$k \times (LB_m - L) + L$	$k \times (U - UB_m) + UB_m$

3.4 Genetic operators

In this work, tournament selection with tournament size of 10 and uniform crossover are used. However, the probability of genes at division points of UP method to be exchanged is zero for the first generation. Otherwise, same chromosomes might be generated. With the mutation operator, parts of the genes of the chromosomes that are *AC*, *PN*, *LB*, and *UB* are altered. For the limit of the *LB* and *UB*, four different mutations have been performed: shifting the whole interval to the left or to the right and increasing or decreasing its size.

The value of mutation probability is increased as the chromosomes become similar to each other. Namely, the higher similarities of the fittest chromosome in the current population, the higher the mutation probability used in that generation. This adaptive mutation probability avoids the convergence of the population to a single highly fit chromosome and forces the last generation to contain many, high-quality chromosomes. It should be in mind that this mutation should be used with an elitist strategy to avoid the loss of the fittest chromosomes. This mutation approach allows that last generation contains many high-quality chromosomes and achieves an effect similar to niching but with less computational cost. By this operator and adjusted fitness approach, a good rule does not replicate itself and dominate the population as in conventional GA. Rather, several good rules have been found and the population has been diversified.

To increase the quality of rules, finally adjusting the intervals for the chosen chromosomes has been performed after genetic search. This has been done by decreasing the size of their intervals until the number of covered records be smaller than the records covered by the original rules.

3.4.1 Uniform operator

In each generation, based on the problem, four or more high-quality chromosomes from two different best chromosomes are generated; genetic diversity is ensured for mining all interesting rules by preventing early convergence. For the encoding proposed in this study, the positions of different values of the parts of the genes in these chromosomes are saved and array the size of which is the number of positions that have different values is generated. While *AC* and *PN* parts are randomly generated, *LB* and *UB* parts are generated by arithmetic mean. Then, from this array, four different arrays are generated similar to UP method for $r = 2$ and the values in these arrays are distributed to the relevant saved positions in the best chromosomes.

For example, let C_0 and C_1 be the best two chromosomes expressed with four genes in a generation as shown in Table 2. Ten positions are different and are marked as "*D*" in the fourth row of Table 1. Assume the lower bound as 0, upper bound as 100, and k as 0.5. Ten values with respect to the relevant parts of the genes are generated. Let this be $\left[\,0\ 25\ 55\ *\ 25\ 52\ 1\ 20\ 1\ 13\,\right]$. For $r = 2$, other generated values are shown in Table 3 as Values_2, Values_3, and Values_4. These first generated values and other Values_i $(i = 1, 2, 3)$ that are derived from these values are distributed to the positions of the saved gene parts of one of the best chromosomes, and generated new chromosomes are shown in Table 4.

3.5 Algorithm complexity

The comparison of the chromosome to the data records is the most critical part of the algorithm complexity. This part of the algorithm complexity strongly depends on four parameters: the number of records in the database (*NR*), the number of attributes in the database (*NA*), the population size (PS), and number of generations (*NG*) within a given evolution process. In each generation, the proposed algorithm scans the data only once and matches all the chromosomes to the scanned records. The population size is usually much smaller than data sample size, that is why, it is much more efficient to perform several scans on the population than to go through the data many times. Here, the database is usually stored on a disk and the population resides on computer memory. Since the matching is nothing more but a value-to-gene comparison of each chromosome in the population against each record in the database, the total number of such comparisons (*CN*) is given by:

$$CN = NR \times NA \times PS \times NG \tag{6}$$

The total complexity of this approach might seem quite high at first. However, mining all interesting negative and positive quantitative ARs involves confronting search spaces of an exponential size. The execution of adjusted fitness computation is faster than that of fitness sharing used for multi-objective GA problems. To calculate the fitness of one chromosome in fitness sharing, the similarity scores of *all* other chromosomes with respect to this chromosome have to be calculated. If a similarity score can be computed in time $O(t)$, and the population size is *upop_size,* each chromosome needs a time $O(upop\_size \times t)$ to calculate the similarity score, and the time needed to complete fitness sharing in each generation is $O(upop\_size^2 \times t)$. On the other hand, calculations of similarity are not needed in adjusted fitness approach. The required information of adjusted fitness approach is the list of records that each chromosome covered. This information is already stored during the evaluation process. If a chromosome covers *nr* records, a time of $O(nr)$ is needed for signing the records as covered, and adjusted fitness computation in each generation can be completed in $O(E \times upop\_size)$, where E is average value of *nr*. This computation is straight forward and can be faster than fitness sharing if $O(nr) < O(upop\_size \times t)$. This computation is clearly more effective than the GA proposed for mining only frequent itemsets by executing the GA as many times as frequent itemsets those have been wanted to obtain [5].

The proposed method directly mines the rules without generating frequent itemsets, and one must keep in mind that automatically finding the intervals in each of the attributes that conforms interesting positive and negative ARs is a difficult problem and this approach is very efficient for this goal.

Table 2 Best two chromosomes and differences

	$Gene_1$				$Gene_2$				$Gene_3$				$Gene_4$			
C_0	0	0	20	70	1	1	30	60	*	1	25	45	0	1	10	60
C_1	0	1	30	40	*	1	20	45	0	1	15	45	1	1	17	60
Y	0	D	D	D	D	1	D	D	D	1	D	45	D	1	D	60

Table 3 Generated values for $r = 2$

First generated values	0	25	55	*	25	52	1	20	1	13
Values2	1	12	77	0	12	80	*	30	*	19
Values3	0	25	55	*	25	80	*	30	*	19
Values4	1	12	77	0	12	52	1	20	1	13

Table 4 New chromosomes

C_{00}	0	0	25	55	*	1	25	52	1	1	20	45	1	1	13	60
C_{01}	0	1	12	77	0	1	12	80	*	1	30	45	*	1	19	60
C_{02}	0	0	25	55	*	1	25	80	*	1	30	45	*	1	19	60
C_{03}	0	1	12	77	0	1	12	52	1	1	20	45	1	1	13	60

4 Experimental results

To verify that the proposed GA correctly finds positive and negative quantitative ARs, a synthetic database was created in which certain rules, previously fixed, were fulfilled in an adequate number of records as to consider them interesting ones. The goal was to find, in an accurate way, the rules. Furthermore the GA was tested on records of the students of Fırat University Electrical and Electronics Engineering and databases on Bilkent University Function Approximation Repository [8]. The obtained results were commented on following subsections.

4.1 Synthetic database

The created database has four quantitative attributes. 1000 records were distributed to best two positive and three negative ARs with 70% support and 100% confidence. The rules pretended to be found are shown in Table 5. Note that all attributes are in the interval [0, 100]. GA was executed with a UP of 48 chromosomes (6 random chromosomes with $r = 3$) and 1,000 generations. The value of the mutation probability varied from 5% (when the fittest chromosome has a single copy in the current population) to 90% (when all chromosomes of the current population are copies of the fittest chromosome). Crossover probability was selected as 60%. $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, and α_5 that have been used in fitness values were selected as 5, 20, 0.05, 0.02, and 0.01 respectively.

A characteristic of GA is that it is stochastic. Thus, the used GA was larger fluctuations in different runs. In order to get a better result, the user may execute several trials of the algorithm to get the result with the best fitness score. GA was executed five times and the average values of such execution were presented. Table 6 shows the best rules found by GA. Other chromosomes also represent different rules, however here only the best five rules are shown. It can be seen that support and confidence of the rules are very close to predetermined values.

Table 5 The rules pretended to be found

$A \in [10, 20] \Rightarrow C \in [15, 40]$
$C \in [25, 45] \Rightarrow B \in [40, 70]$
$\neg D \in [15, 45] \Rightarrow C \in [20, 45]$
$\neg C \in [45, 85] \Rightarrow \neg A \in [30, 95] \wedge B \in [50, 75]$
$\neg C \in [10, 45] \Rightarrow D \in [25, 40]$

4.2 Real databases

To test the proposed method, records of the Electrical and Electronic Engineering students' class grades of Fırat University in Turkey were selected as sample database. More information can be found at http://www.firat.edu.tr. Furthermore, some experiments were performed using the Bilkent University Function Approximation Repository [8]. The used student database was partitioned to eight parts such as "first semester of first year", "second semester of first year", etc. The number of records in each semester is shown in Table 7.

Table 6 Obtained best rules

Obtained Best Rules	Supp (%)	Conf (%)
$A \in [10, 22] \Rightarrow C \in [16, 40]$	68	91
$C \in [19, 48] \Rightarrow B \in [44, 70]$	63	88
$\neg D \in [15, 45] \Rightarrow C \in [20, 45]$	70	100
$\neg C \in [45, 85] \Rightarrow \neg A \in [32, 95] \wedge B \in [50, 75]$	69	87
$\neg C \in [12, 45] \Rightarrow D \in [25, 40]$	68	98

Table 7 Number of records in each semester

Semester	Number of records	Semester	Number of records
1. Year 1. Semester	279	3. Year 1. Semester	170
1. Year 2. Semester	272	3. Year 2. Semester	166
2. Year 1. Semester	278	4. Year 1. Semester	157
2. Year 2. Semester	150	4. Year 2. Semester	161

Table 8 Mined best ARs for two semesters

Semester	Best Rules	Support (%)	Confidence (%)
1	B_comp_techn∈[72,88]⇒Turkish_lang1∈[50,74]	64	75
	Turkish_lang1∈[55,76]⇒B_comp_techn∈[69,88]	56	71
	Foreign_lang1∈[70,89]⇒B_comp_techn∈[52,79]	50	70
	B_comp_techn∈ ¬[64, 84] ⇒ Foreign_lang1∈ ¬ [54,80]	42	68
2	Turkish_lang2 ∈[60, 72]⇒ G_chemistry∈[38,76]	42	69
	G_chemistry∈[37,71] ⇒ Turkish_lang2 ∈[60, 75]	49	68
	Turkish_lang2 ∈ ¬ [43, 70] ⇒Physics2∈ ¬ [38,61]	62	67
	Turkish_lang2 ∈ ¬ [50, 75] ⇒ G_chemistry∈ ¬ [41,71]	60	60

Table 9 The used databases, number of rules, and confidence values

Database	No. of Records	No. of Attributes	No. of Rules	Confidence
Basketball	96	5	24	59
Bolts	40	8	30	65
Pollution	60	16	32	68
Quake	2178	4	34	62
Sleep	62	8	29	63

Thus, rules were found for each semester. The same parameters used for synthetic database were used for GA to mine interesting rules. Some of the best mined rules for two semesters are shown in Table 8. Rules have both high support and confidence values.

The proposed algorithm was also evaluated in five public domain databases: Basketball, Bolts, Pollution, Quake, Sleep. These databases are available from Bilkent University Function Approximation Repository. Table 9 shows the number of records and number of numeric attributes for each database as well as the mean number of found different high-quality rules and the mean of confidence value of these rules. Table 10 shows the comparison of obtained results from the proposed method and the GAR algorithm proposed in [7]. Note that GAR discovers only the positive frequent itemsets, not negative rules. The value of the column Support(%) indicates the mean of support, while the value of the column Size shows the mean number of attributes contained in the itemsets or rules. The column Amplitude (%) indicates the mean size of the intervals that conforms the set. The proposed GA has found rules with high values of support in two out of five databases and the difference is not significant due to finding both positive and negative rules. However, it found the rules without expanding the intervals in excess. The size and amplitude values obtained from this method are smaller than GAR. Thus, the discovered rules are more readable and comprehensible.

5 Conclusions

In this paper, an efficient GA for quantitative AR mining problem has been proposed. Not only positive but also negative quantitative ARs within databases have been mined. GA has been designed to simultaneously search for intervals of quantitative attributes that conforms a rule, in such a way that the fitness function itself is the one that decides the amplitude of the intervals. In this way, the problem of finding rules only with the intervals created before starting the process that have been used in the literature has been avoided. Contrary to the methods used as usual, ARs with high support and confidence have directly been mined without generating frequent itemsets and relying upon the minimum support and the minimum confidence thresholds that are hard to determine for each database has been performed.

Instead of randomly generated initial population, uniform population that forces the initial population to be not far away from the solutions and distributes it in the feasible region uniformly has been used. An adaptive mutation probability, uniform operator, and an efficient adjusted fitness function have been used for mining all interesting ARs from the last population in only single run of GA. The execution of adjusted fitness computation is faster than that of fitness sharing used for multi-objective GA problems. This approach is also more efficient than executing the GA as many times as frequent itemsets or rules those have been wanted to obtain. Several tests have been performed to check the GA in synthetic and real databases and satisfactory results have been obtained.

Table 10 Comparison of the obtained results

Database	Support (%)		Size		Amplitude (%)	
	This work	GAR	This work	GAR	This work	GAR
Basketball	31,59	36,69	3,21	3,38	20	25
Bolts	27,18	25,97	5,14	5,29	27	34
Pollution	30,08	46,55	6,21	7,32	14	15
Quake	34,91	38,65	2,1	2,33	19	25
Sleep	37,32	35,91	4,19	4,21	4	5

The proposed method has a very appropriate structure for parallel or distributed architectures and we plan a parallel implementation of this method with more elaborated experiments by using optimized parameters and embedding new interestingness measures into fitness function for different task of data mining such as sequential patterns, classification, and clustering.

Appendix

Algorithm GAR
nItemset = 0
while (nItemset < N) **do**
    nGen = 0
    generate first population P(nGen)
    while (nGen < NGENERATIONS) **do**
        process P(nGen)
        P(nGen+1) = select chromosomes of P(nGen)
        complete P(nGen+1) by crossover
        make mutations in P(nGen+1)
        nGen++
    end_while
    I[nItemset] = choose the best of P(nGen)
    penalize records covered by I[nItemset]
    nItemset++
end_while

Algorithm GAR [7] finds only positive frequent itemsets. Positive rules are built departing from them. First, initial population is generated and the fitness of each chromosome is computed. Then selection, crossover, and mutation are carried out. The highest-quality chromosome that corresponds to one of the positive frequent itemsets is selected. The process is repeated until the desired number of frequent itemsets N is obtained. Records covered by the obtained itemsets in the previous steps are penalized to prevent finding the same itemsets in later executions of GA. Elitist strategy, uniform crossover, and mutation operator that alters one or more genes of chromosome are used.

References

1. Han J, Kamber M (2001) Data mining: concepts and techniques. Morgan Kaufmann, Academic, San Fransisco, New York
2. Karcı A (2004) Novelty in the generation of initial population for genetic algorithms. Knowledge-based intelligent information and engineering systems. Part II In: 8th International conference KES 2004 (LNAI), Wellington, New Zealand, 3214: 268–276
3. Agrawal R, Imielinski T, Swami AN (1993) Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD, Washington, D.C. 207–216
4. Srikant R, Agrawal R (1996) Mining quantitative association rules in large relational tables. In: Proceedings ACMSIGMOD 1–12
5. Fukada T, Yasuhiko M, Sinichi M, Tokuyama T (1996) Mining optimized association rules for numeric attributes. In: Proceedings of the ACM SIGMOD International conference on management of data, Montreal Canada, 13–23
6. Aumann Y, Lindell Y (1999) A statistical theory for quantitative association rules. In: Proceedings of the 5th ACM SIGKDD International conference on knowledge discovery and data mining, 261–270.
7. Mata J, Alvarez JL, Riquelme JC (2002) Discovering numeric association rules via evolutionary algorithm. In: 6th Pacific-Asia conference on knowledge discovery and data mining PAKDD-02 (LNAI) Taiwan 2336: 40–51
8. Guvenir HA, Uysal I (2000) Bilkent University Function Approximation Repository. http://funapp.cs.bilkent.edu.tr