Efficient Generation of Association Rules from Numeric Data Using Genetic Algorithm for Smart Cities

Pardeep Kumar and Amit Kumar Singh

Abstract Machine learning plays an important role to develop smart cities by gathering real time information using several state of the art algorithms. In the recent past, association rule mining plays an important role in the discovery of accurate information from databases to satisfy the need of real time applications in smart cities ranging from healthcare to intelligent transport systems. It is used in various applications for decision making, detection and prediction etc. because of its robustness to derive associations among various attributes of datasets. This technique seems to be simple in case of categorical data but becomes quite complex in case of numeric data. In this work, we have mainly concentrated on the problem of generating association rules from numeric data in an efficient way. For accomplishing this task we have taken genetic algorithm as the base of the solution to this problem. Genetic algorithm is selected for this task because of its nature of self-improving and ability to handle large solution set. Here we have proposed genetic algorithm based association rule mining algorithm which generates random association rules on the basis of general property of datasets. The generated rule set is improved at each run of algorithm and filtered for more and more interesting and accurate rules.

Keywords Association rule mining · Decision making · Categorical data
Numeric data · Genetic algorithm · Smart cities

1 Introduction

Large amounts of data is being collected routinely in the task of day-to-day management and decision making in banking, business, administration, the delivery

P. Kumar (✉)
Department of Computer Science and Engineering, Jaypee University
of Information Technology, Solan 173234, Himachal Pradesh, India
e-mail: pardeepkumarkhokhar@gmail.com

A. K. Singh
Department of Computer Science and Engineering, National Institute of Technology Patna, Patna
University Campus, Patna 800005, Bihar, India
e-mail: amit_245singh@yahoo.com

A. E. Hassanien et al. (eds.), *Security in Smart Cities: Models, Applications, and Challenges*, Lecture Notes in Intelligent Transportation and Infrastructure,
https://doi.org/10.1007/978-3-030-01560-2_14

of social and health services, works related to environmental protection, handling security and in politics. Such data are primarily used for accounting service such as decision building, prediction, etc. to users of the system, later it can be used for maintenance in long term. Typically, data sets to be considered for this task are very large and constantly growing and contain a large number of complex dimensionality. While these data sets reflect properties of the managed subjects and relations between them, and are thus potentially of some use to their owner, but they often have relatively less useful information as compared to its volume. Hence, robust, efficient, simple and computationally efficient algorithm is required to extract information from such datasets.

One of the important data mining tasks is generating association rules, which deals with finding IF-THEN rules by analyzing similarity between combinations of attributes of data. As it can be seen that IF-THEN rules are easy to understand and can be used in multiple applications for deriving inferences for relevant tasks, therefore generating such kind of rules is considered as an important task in data mining.

To generate such algorithm in an efficient way, one of the important subjects of computer science also helps data mining, i.e. machine learning. Machine learning is a subject which keeps efficient solution of various general problems which can be used in any field of study such as data mining, optimization, information retrieval etc. We will also be using one of the machine learning techniques in our work i.e. genetic algorithm.

2 Basic Terminology

2.1 Numeric Association Rule Mining

Association rule mining is a well-known data mining technique that is used for extracting information from raw data in form of IF-THEN statement. These statements can be used in many fields for prediction and decision making. First introduced in Agrawal et al. [1], the process of extracting such rules was quite cumbersome where all the possible combinations are checked for co-relations. The co-relation between attributes was checked on basis of interestingness. Each rule is checked for interestingness using various interestingness parameters such as support, confidence, lift, conviction etc. Various such parameters are given in Tew et al. [2]. Later same author improved the procedure of extracting rules in Agrawal and Srikant [3] where the authors proposed a well-known Apriori algorithm in which only frequent occurring pairs are checked for further co-relation. After the introduction of Apriori, lots of other algorithms were proposed for extracting association rules. Like in Houtsma and Swami [4], association rule mining using sql query was explained but it requires multiple scans of database. In Hidber [5], an online technique for generation of association rule was proposed where the rules are generated at real time. A comparison between two well-known algorithms Apriori–Eclat was discussed in Borgelt [6]. But all these algorithms were designed for categorical data only. None of these algorithm works well for numeric dataset. The rules needed for numeric dataset are of form:

$$\text{attribute1[lb1, ub1], attribute2[lb2, ub2]} \Rightarrow \text{attribute3[lb3, ub3]}$$

where there is a lower and upper bound attached with every attribute. These kinds of rules are more general in nature as well as contains more information as compared to categorical association rules. By extracting numeric association rule, various inter disciplinary fields can be benefited such as medication, structural engineering, pollution analysis, image processing etc.

2.2 Genetic Algorithm

Genetic Algorithm (GA) explained in Goldberg [7] is a powerful algorithmic technique that can be used for generating solution for a search and optimization problem [8, 9]. It basically follows one of the nature's principle i.e. survival of the fittest as stated in Darwin's theory of evolution. It tells that the one that is stronger survives for longer. This algorithm finds better solution in each generation and keeps on improving the solution in each generation till it finds a near optimal solution.

The process of general genetic algorithm starts with the description of problem; on the basis of problem and the solution needed an encoding scheme is selected. Chromosomes are generated as initial population by using that encoding scheme. Fitness value is calculated for each chromosome and on the basis of that it is decided whether the solution is appropriate or not. Further the population is changed by applying crossover and mutation operators which is meant for introducing diversity in the algorithm. By implementing these operators, a new generation of population is evolved then again on that population fitness is checked. This is an iterative self-improving process which improves its own solution iteration by iteration. The iterations are stopped when the desirable results are generated. The population in the final iteration is considered to be the best solution set. As per the requirement of the problem results can be formulated from the population of last generation. A general work flow of genetic algorithm is illustrated in Fig. 1 given in Obitko [10].

3 Literature Survey

To derive association rules of form IF-THEN statement, we need to get an antecedent and a related consequent part which are co-related to each other. For applying GA for finding such rules as mentioned in Freitas [11], four main designing components are needed i.e. Chromosome encoding for representing rules in form of GA understandable chromosomes, Crossover operator for generating new child chromosomes, Mutation operators for introducing diversity in the population and fitness function

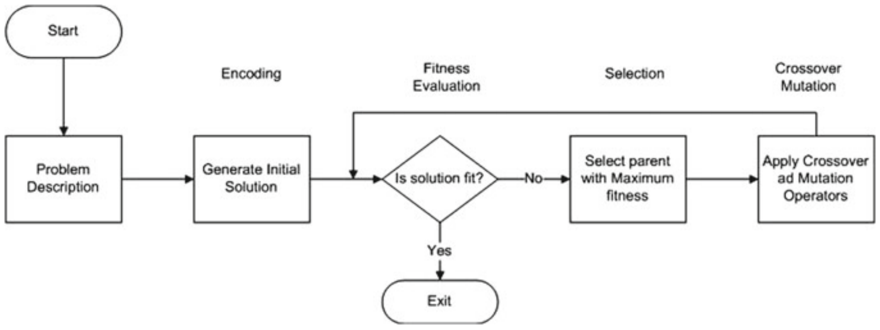


Fig. 1 Flowchart of basic genetic algorithm

which will evaluate fitness measure of each rule. After deciding these factors only thing left is to apply basic genetic algorithm which is defined in introduction part above, using all these components.

3.1 Chromosome Representation

Broadly there are two chromosome representation techniques are present in literature for ARM. In the first approach (Pittsburgh approach), a batch of possible association rules are encoded in a chromosome. This encoding approach is well suitable for classification rule mining, where the goal is to generate a good quality set of rules. However, in ARM, the main goal is to find a set of rules where each rule is good in itself. Therefore, for ARM case, the Michigan approach explained in Goldberg [7] is mostly used. Each chromosome represents exactly one rule in itself, is more suitable in Michigan approach. Most of the ARM techniques use Michigan chromosome representation.

In an early work by Ghosh and Nath [12], the authors use the Michigan approach as follows: each chromosome has length $2k$, where k is the number of items. The chromosomes in this case were binary strings where each attribute is given two bits. If these two bits are 00 or 11, then the attribute is present in the antecedent part or consequent part of the rule, respectively, else the attribute is not present in the rule.

In paper by Anand et al. [13], each chromosome has two parts. The first part indicates the location of the attribute in the rule and the other part indicates the categorical value it has. The prior part consists of two bits where the attribute appears in the antecedent part or the consequent part of the rule, if the bits are 10 and 11, respectively; else, it is meant to be absent from the rule. The other part represents the categorical values carried by attributes in binary form. However, the authors in the paper did not given any justification of how the binary value of the attribute in the second part will appear and how categorical state will be managed if the number of states for an attribute is not an exact power of two. The main demerit of choosing a binary encoding scheme is that the length of the chromosome is large when the number of attributes increases, because at least two bits are required for each attribute representation. An integer encoding can be used as a solution to this problem.

An integer encoding scheme has been used in association rule mining using multi objective genetic algorithm by (ARMGA) Yan et al. [14]. In this work the chromosomes encode the index of the attributes. A chromosome represented as encoding a k-rule, k is the total number of attributes in the antecedent part and the consequent part of the rule, and this chromosome will have $k+1$ genes. The first gene position represents the differentiating position of the chromosome where the attributes of antecedent and the consequent are separated. For example, if Ai represents the ith item, then the chromosome $\{3 \mid 5\ 4\ 2\ 1\ 3\}$ represents the rule $A2, A5, A4 \Rightarrow A1, A3$. This kind of representation reduces length of the chromosome significantly.

In Alatas et al. [15], encoding scheme used for chromosomes representation where each attributes has three parts. The first part tells whether the attribute is present or not in the rule, and if present, in which part of the rule (antecedent or consequent) it is present. The second and third part tells the lower and upper bounds of the value of attribute. The first part can have integer numbers such as 0, 1, or 2, which represents the occurrence of the attribute in the antecedent part of the rule, the occurrence of the attribute in the consequent of the rule, and the absence of the attribute from the given rule, respectively. Next, the second and the third part can take real numbers from the related attribute range. Further it is to be noted that as MODENAR uses differential evolution as an optimization technique and works on real-valued chromosomes, the authors considered a round off operator for handling the integer value part of the chromosome. In Martin et al. [16], authors use positional encoding scheme. Metawa et al. [17] used binary chromosome encoding scheme in their genetic algorithm based proposed approach for optimizing bank lending decisions to specify the selected customer in the lending decision.

3.2 Objective Functions

Even though support count and confidence are two well-known objectives that are generally to be maximized yet there are several other parameters available to measure the interestingness or strength of association rules as mentioned in Berzal et al. [18]. Some of the parameters, which can be used by different algorithms for optimization in a multi-objective scenario, are comprehensibility, conviction, interestingness, performance, lift, coverage, precision etc. Each function has its own significance and combination of best of these functions gives best result in the end. In Alatas et al. [15], objective function tries to optimize four criteria of the rules: support, confidence, amplitude of the intervals and comprehensibility, which make up the subparts of the rule. In Martin et al. [16], authors use comprehensibility, interestingness, and performance for fitness calculation. Anand and Vinodchandra [19] proposed association rule mining algorithm using treap data structure where they focused on running time rather than rule set quality. Umit and Bilal [20] proposed exploration of association rules within numerical databases with Gravitational Search Algorithm (GSA) with flexible fitness function. In Uroš et al. [21], authors proposed a modified single-objective binary cuckoo search for association rule mining (MBCS-ARM) where objective function composed of support and confidence only. The quality of rule sets was still an issue in literature.

3.3 Evolutionary Operators

Mostly, when binary encoding scheme is used, some of the standard crossover and mutation operators are used. For example, in Ghosh and Nath [12], bit-flip mutation and multipoint crossover have been used. In Anand et al. [13], mutation operator, bit flip has been adopted. However, the authors did not specifically mention which crossover operator should be used.

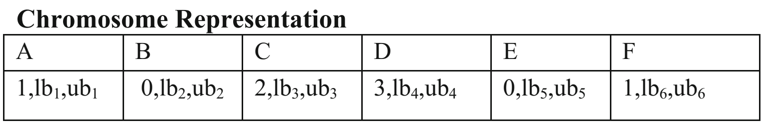
In Qodmanan et al. [22], where integer encoding for the chromosomes is used, an order-1 crossover technique is taken into consideration. In this technique, first a segment is selected from any two parent chromosomes and these are copied to the two child chromosomes. Next, starting from the right side of the segment, the values of the genes that didn't appear in the selected segment of the first parent, are copied to the first child. The same procedure is repeated for the second child as well. The mutation operator introduces diversity by replacing a selected attribute value from the chromosome with a random attribute value not present in the chromosome currently. Alatas et al. [15] used differential evolution based crossover and mutation operators.

4 Proposed Approach

In the literature, till date a lot of work has been done in the field of association rule mining but most of the work is done for extracting rules from categorical data only. A very few algorithm can be seen for dealing with numeric data to get the relevant and interesting association rules. We have kept our focus mainly on mining association rule from numeric data. To handle numeric data efficiently, it is required that the data related to each attribute should be checked individually which increases the search space exponentially. Therefore, to deal with such problem, we have used genetic algorithm which finds solution with the least possible domain knowledge. In this work we have analyzed the basic structure of previously available work on same domain and proposed an algorithm which generates better results from the previously known works of similar kind.

In the proposed work we will be generating the rules of the form:

$$\text{attribute1[lb1, ub1], attribute2[lb2, ub2]} \rightarrow \text{attribute3[lb3, ub3]}$$



Chromosome Representation

A	B	C	D	E	F
1,lb_1,ub_1	0,lb_2,ub_2	2,lb_3,ub_3	3,lb_4,ub_4	0,lb_5,ub_5	1,lb_6,ub_6

Fig. 2 Chromosome representation for proposed approach

which signifies if the value of attribute1 is between lb1 and ub1, attribute2 is between lb2 and ub2 then the value of attribute3 will be between lb3 and ub3. In this approach there is no limitation on length of either of antecedent or consequent part.

4.1 Technical Details of Implementation

As discussed earlier, for using genetic algorithm for any problem, the basic components of genetic algorithm (i.e. encoding scheme, initial population, crossover and mutation operator and fitness function) have to be decided. Therefore, in our case for generating rules using genetic algorithm, the part of algorithm that we have used is as follows:

Chromosome Representation
We have used Michigan encoding scheme in this implementation where one chromosome represent one rule in the population. The chromosome is represented as shown in Fig. 2 where column under A represents a gene corresponding to attribute A which consists of 3 parts. First part represents the location of attribute whether it will be in antecedent part or it will be in consequent part. If first bit of gene is 0 indicates this attribute is in antecedent part, 1 indicates it to be in consequent part, whereas 2 indicate that the attribute is not present in the rule. Other two parts of the gene are lower bound and upper bound value of the attribute. Chromosome in Fig. 2 shows a rule of form:

$$B[lb_2, ub_2],\ E[lb_5, ub_5] => A[lb_1, ub_1],\ F[lb_6, ub_6]$$

which represents rule as IF B has a value between lb_2 and ub_2 and value of E is between lb_5 and ub_5 THEN the value of A will be in lb_1 and ub_1 and the value of F will be in lb_6 and ub_6.

Population Generation
Initial population generation in implementation is done by keeping in mind the frequency of value of attribute that occurs in the dataset i.e. a value that is more frequently coming into dataset are selected for generating chromosome. This is because the initial population is an important factor for better output of genetic algorithm as the whole flow of genetic algorithm is broadly based on the first generation. It is the solution of the first generation which is improved further in other generation. If we

keep a check on initial population then there is chance of getting better results as the output of the genetic algorithm.

Fitness Calculation

For every chromosome in the population, fitness is calculated. This fitness is the indicator of interestingness and importance of the rule. Various important interestingness parameters have been compared in Tan et al. [23]. It is to be decided prior that which interestingness parameters to be selected for better results. We have selected support, confidence, lift and conviction for our proposed work. If chromosome has fitness greater than the threshold fitness then it is considered for next generation else it is restricted to enter in the next generation. The threshold is calculated as the average of the fitness of rules of last generation. In this implementation we have taken a power fitness function. For calculating fitness function, initially support, confidence, lift and conviction of the chromosome is calculated.

For a sample rule like 'if A then B':

Interestingness Measures used in proposed algorithm are:

1. Support [1]—indicates how frequently A and B occur in the dataset.

$$\text{Support} = \frac{\text{Count}(\text{A} \cup \text{B})}{\#\,\text{of Records}}$$

2. Confidence [1]—indicates how frequently A and B occurred when it is known that A was already present in the record.

$$\text{Confidence} = \frac{\text{Support}(\text{A} \cup \text{B})}{\text{Support}(\text{A})}$$

3. Lift [24]—indicates how frequently A and B occurred when it is known that B was already present in the record.

$$\text{Lift} = \frac{\text{Confidence}(A \rightarrow B)}{\text{Support}(B)}$$

4. Conviction—indicates the comparison of the probability that A appears without B if they are dependent with the actual frequency of the appearance of A without B.

$$\text{conviction} = \frac{\text{P(A)P}(\sim\text{B})}{\text{P}(A(\sim\text{B}))}$$

First fitness filter is created on the basis of support. A minimum Support value is taken and only the chromosomes having fitness value greater than support will be considered for rest of the fitness value calculation process.

Rest of the fitness calculation is done by a multi-objective maximizing function of form

$$Fitness = \left(Support^4\right) + \left(Confidence^3\right) + \left(Lift^2\right) + \left(Conviction^2\right)$$

where constant 4, 3, 2, 2 in the power are chosen on the basis of importance of these interestingness parameter. The main motive of this function is to maximize the effect of interestingness parameter. The chromosome with the fitness value less than the average fitness value is not considered for next generation. This chromosome is sent to the mutation repository for generating new chromosome out of it with few improvements.

Crossover Operator
Crossover is done among two fit parent chromosome to generate two new off springs. Various crossover operators and their performance have been discussed in Magalhães-Mendes and Cidem [25]. For this implementation, we have taken a hybrid of two crossover operator i.e. single point and every point crossover operator. In one generation single point crossover is implemented and in next generation every point crossover is applied. This is repeated in the same way for all iteration. This is done to introduce better structure to generate new population. The idea behind combining the two crossover operator is that, at certain iteration we crossover at one point and on certain other iterations we do the crossover at every point in which swapping of gene of each attribute takes place. First operator preserves the quality of fit chromosome whereas every point crossover will introduce more chromosomes with better quality of previous chromosome. By keeping the balance among both of them will generate better chromosomes at different generation.

Mutation Operator
Mutation is applied to bring diversity in population. In mutation some of the bits of previous chromosome are changed to some other random value. For our implementation, we have implemented a selective bit mutation where a bit is selected to change in the chromosome and then generate a new chromosome. And while applying the mutation operator we also have kept a provision of generating some new random chromosome similar to the way it was generated in initial population phase. We apply mutation operator over chromosomes which are not fit i.e. the chromosome that have not come in the threshold fitness range in previous generation. There we select any of the gene (attribute) of the chromosome and increase or reduces the lower bound and upper bound or change the first bit which is representing location of attribute in the rule. By changing lower bound and upper bound there is chance of improvement in chromosome. Once the chromosome become fit by changing range of some attribute next time it will enter the population and may be selected for crossover in next few iterations.

Selection of Chromosome
We have used the probability based procedure for selecting two chromosomes for crossover in which the support count is used as a probability parameter. A parent chromosome is selected having value higher than the average support count of entire population.

4.2 Flow Chart for Implementation

Flow chart of our proposed approach is depicted in Fig. 3.

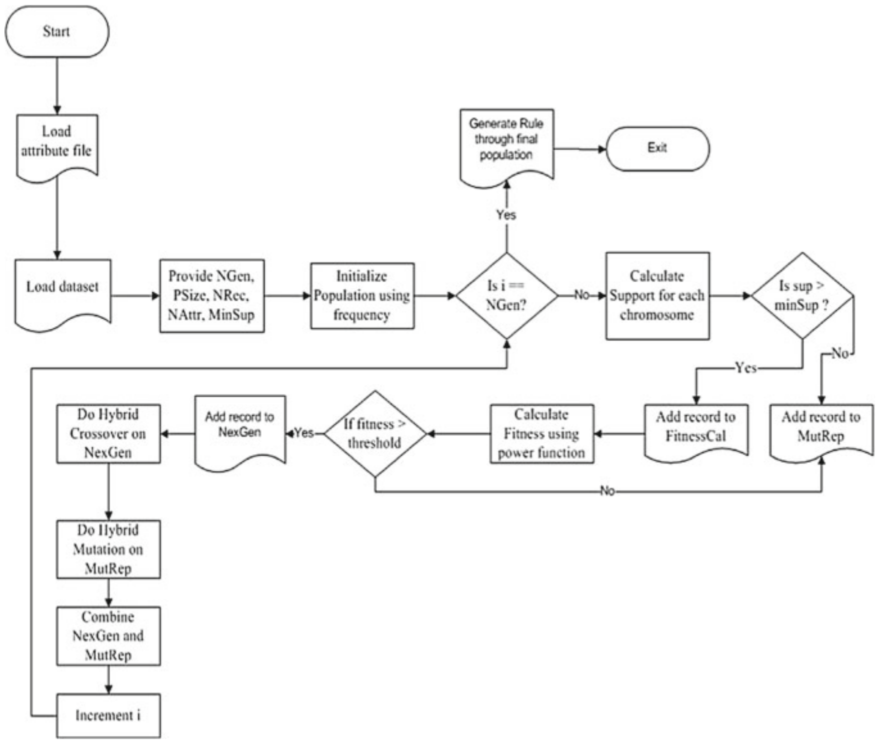

Start
Load attribute file
Load dataset
Provide NGen, PSize, NRec, NAttr, MinSup
Initialize Population using frequency
Is i == NGen?
Yes
Generate Rule through final population
Exit
No
Calculate Support for each chromosome
Is sup > minSup ?
Yes
No
Add record to FitnessCal
Add record to MutRep
Calculate Fitness using power function
If fitness > threshold
Yes
No
Add record to NexGen
Do Hybrid Crossover on NexGen
Do Hybrid Mutation on MutRep
Combine NexGen and MutRep
Increment i

Fig. 3 Flowchart for proposed approach

4.3 Pseudo Code for Implementation

Algorithm: **Implementation**

```
Input: Dataset D

PSize - Population Size
NGen - Number of generations
RMut- Mutation Rate
RCross- Crossover Rate

Output: Association Rules

1.Begin
2. fori=1 to PSize  //No. of Chromosome in one generation
3.   for each attribute
4.          Check upper bound and lower bound.
5.          Calculate frequency of values in dataset.
6.          Generate chromosome from step 4 – 5 information
7.   End for
8. End for    //Initial population generated
9. for each Psize  // for each chromosome
10.          Calculate Support
11.          If Support >MinSupport
12.          Calculate Fitness
13. End for
14. for each number of generated rules of Step 5-9
15.    If (Fitness >AvgFitness) //AvgFitness of last generation
16.         Add rule to nexGen
17.    Else
18.         Add rule to MutRep   //MutRep:Rules for mutation
19. End for
20.   for  i=1 to NGen // No. of iteration
21.      do Crossover on nextGen
22.      do Mutation on MutRep
23.      Repeat Step 9 – Step 19
24.  End for
25. End
```

5 Experimental Setup

We have implemented the algorithm in JAVA and tested results on various datasets of keel repository to analyze the proposed algorithm. The program is implemented on NetBeans 8.0.2 and executed on a DELL Inspiron Laptop with Intel® Core™ i5-4210U CPU @ 1.70 GHZ 2.50 GHz and 8 GB RAM.

5.1 Datasets

The above proposed algorithms are implemented in such a way that it can be executed for any dataset of real numbers. For presenting the result, we have implemented and tested our algorithm on 4 datasets of keel repository as mentioned in Alcala-Fdez et al. [26]. These datasets are shown in Table 1.

The implementation done in this work is a generalized implementation which can be applied to any dataset. As the dataset and the name of the attributes have to be given in form of a .txt file in the beginning of the execution of the process. Here we have discussed the performance of the proposed algorithm on the measures of number of rules generated, execution time of algorithm and trends in the average fitness value of the rules at different generation of the execution.

6 Results of Proposed Algorithm

We have first illustrated different results that we got on implementation of proposed algorithms on Quake and Stocks dataset. Later we have illustrated the comparison between algorithms proposed by different authors and our proposed algorithm. We have taken crossover rate as 0.2 and mutation rate as 0.02 carry out results. Results of proposed algorithm on Quake dataset are shown in Table 2.

Figure 4 illustrate the number of rules that are generated by implementation on different support counts. As discussed earlier support count is the first filter of selecting chromosomes for calculating fitness. Minimum support has to be given manually at the starting on the basis of need of application. It can be shown that with increase

Table 1 Datasets taken for analysis

Dataset name	No. of attributes	No. of records	Type of dataset
Quake	4	2178	Real valued
Stocks	10	950	Real valued
Pollution	16	60	Real and integer
Stulong	5	1419	Integer

Table 2 Parameter values for implementation on Quake dataset

Parameters	Values
Dataset: Quake dataset	(4/2178) (attributes/records)
Population size	100
Number of generation	10

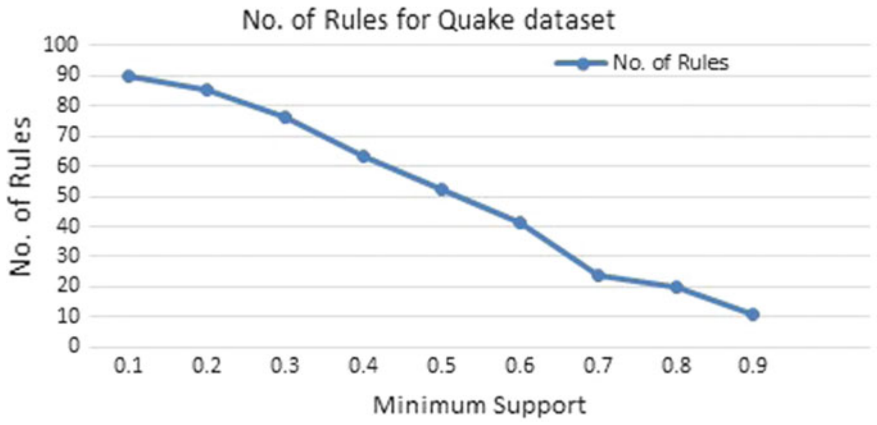


Fig. 4 No. of rules generated for Quake dataset

in minimum support count, the numbers of rules that are generating are decreasing which signify that selecting min. support for filter purpose is a correct decision.

Figure 5 illustrates the execution time taken for quake dataset for single generation. It can be seen that with increasing the minimum support count, execution time is decreasing and it can also be concluded from this graph that for minimum support of 0.1, 0.2, 0.3, the execution time is much higher because number of rules generated are much higher at such a low min. support value. Therefore, execution time of simulation gets increased whereas from minimum count 0.4 onwards, the number of rules that get filtered out for further processing is low. Hence, execution time also decreases to a relevant extent.

Figure 6 illustrates the average fitness of rules with minimum support count. As the min. support count is increasing, the average of fitness value of the generated rules is also increases. It signifies that with increase in minimum support, the relevancy of the rules increases i.e. more interesting rules get generated.

Further we have illustrated some more plots for different generations to analyze the improving nature of rules at each generation. It is a property of genetic algorithm that all iterations of execution lead to improve the solution set. To see the effect of this property in our proposed implementation, we have plotted the results of different generation which are illustrated here.

Figure 7 illustrates the fitness of rules generated for Stocks dataset for generation 1. The fitness of those chromosomes is shown as dots in the figure.

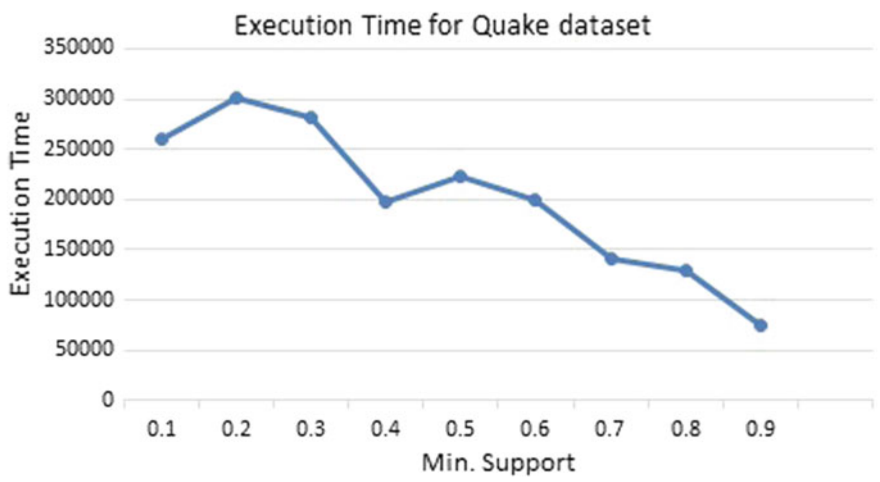


Fig. 5 Execution time taken for Quake dataset

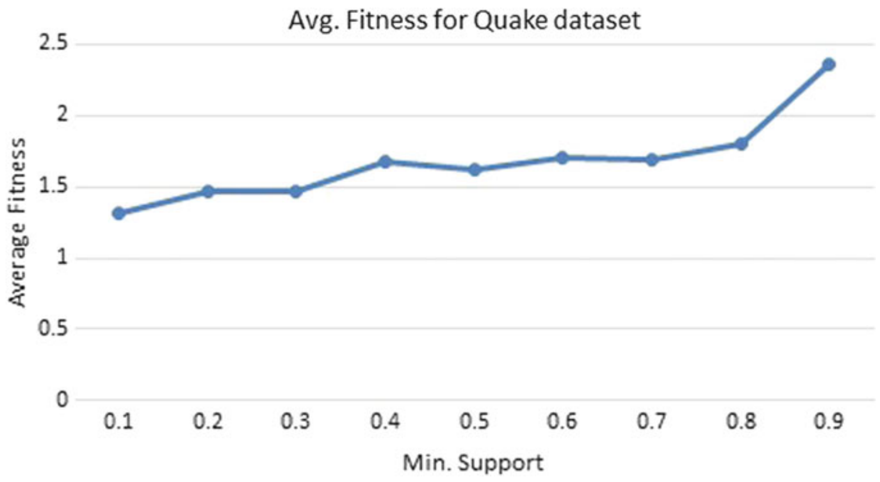


Fig. 6 Average fitness of rules generated for Quake dataset

Figure 8 illustrates the fitness of rules that are being generated at different generation i.e. for generation 1, generation 4, generation 6, generation 8 and generation 10 of the implementation for Stocks dataset. It can be shown in the figure that fitness of rules gets better and better with increasing generation, i.e. at all iteration rules are getting more interesting in terms of fitness. Also the numbers of rules are decreasing at each generation because of filtering of unfit or not-interesting rules at all iteration.

Figure 9 illustrates the fitness of rules at different generation for quake dataset. It shows the similar kind of results as of Fig. 8 i.e. with increase in generation, the rules are getting more refined due to increase in the fitness value of rules (shown by

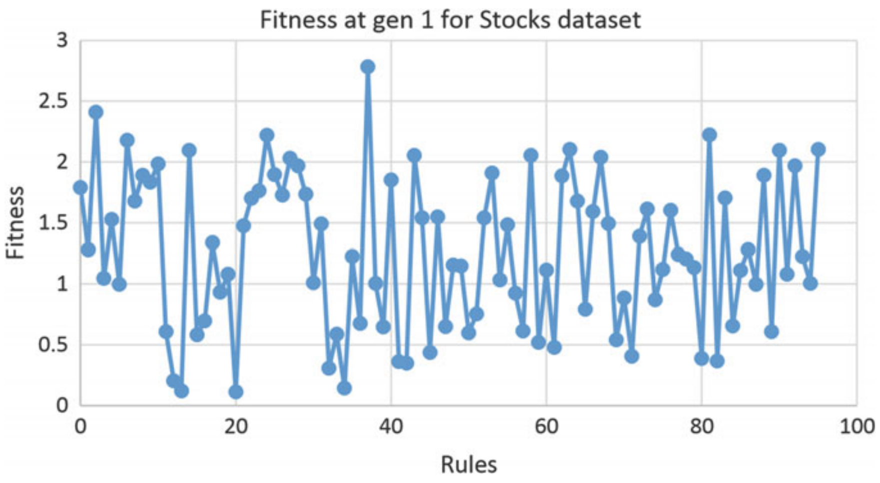


Fig. 7 Fitness of rules generated for Stocks dataset

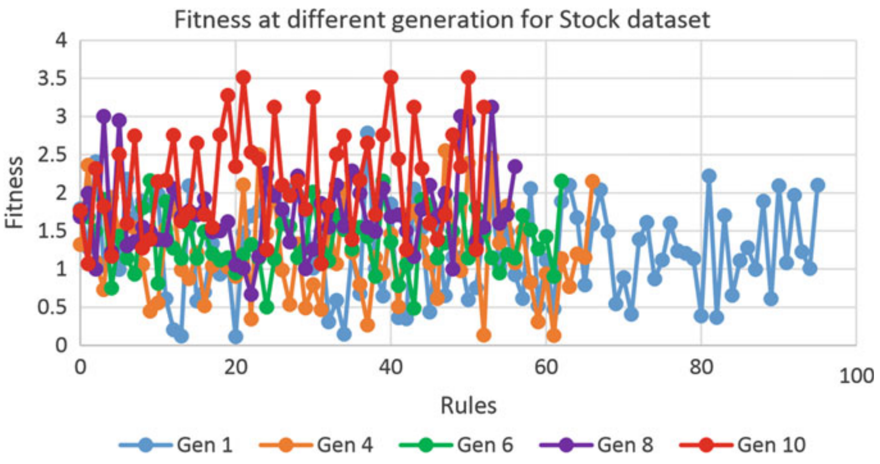


Fig. 8 Fitness of rules generated for Stocks dataset at different generation

peaks in the graph) and decreasing the number of rules (shown by tail of the plot for each generation).

We can see that our proposed algorithm is following the basic property of genetic algorithm i.e. the evolving nature of the solution at all iterations with the results of both the above graphs.

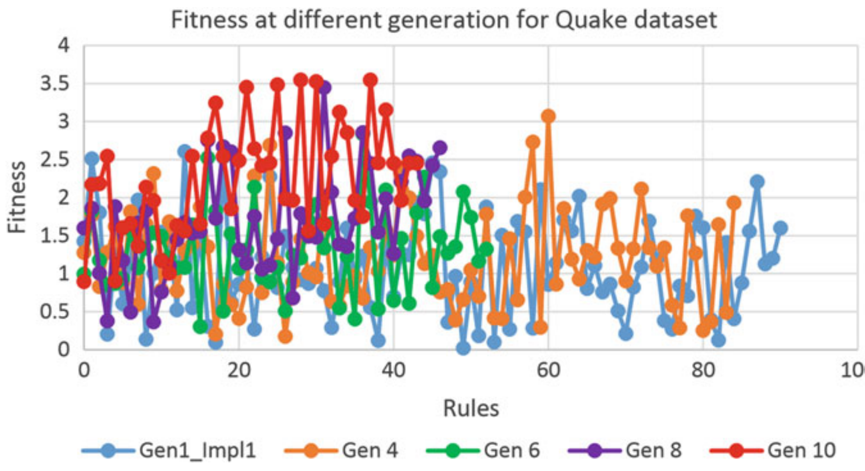


Fig. 9 Fitness of rules generated for Quake dataset at different generation

6.1 Comparison of Proposed Approach with Existing Approach

Next, to compare our proposed algorithm with previously known algorithms, we have plotted the comparison bar graphs. We have compared our results with four previously known algorithms that are EARMGA [14], GENAR [27], MODENAR [15] and MOPNAR [16]. Details of these algorithms have been discussed in the literature survey section. We have compared our algorithm on the basis of number of rules these algorithms have generated at min. support count of 0.5, average of support count of the rules that are generated by these algorithms. For comparison we have taken four datasets from the Keel dataset repository i.e. Quake, Pollution, Stocks and Stulong. The comparison values of previous approaches i.e. number of rules and average support for previously known algorithm have been taken from the literature Martin et al. [16] and that is compared with the results generated by implementing proposed algorithms.

Figure 10 is the graphical bar plot of data presented in Table 3 taken from Minaei-Bidgoli et al. [28], Al-Maqaleh [29], Martin et al. [16] and the results found on implementation of our proposed approach. This bar plot illustrate the comparison between various previously known algorithms with the proposed algorithm of our work on the basis of number of rules generated by these algorithms with different datasets. From this plot, we can conclude that our proposed algorithm have generated less number of rules as compared to EARMGA, MODENAR and MOPNAR (see Fig. 11 and Table 4).

This bar plot illustrates the average support of rules generated by different algorithms on different datasets. Higher average support of rules is a desirable parameter, as higher value of support count indicates the more interesting rules. As per the plot

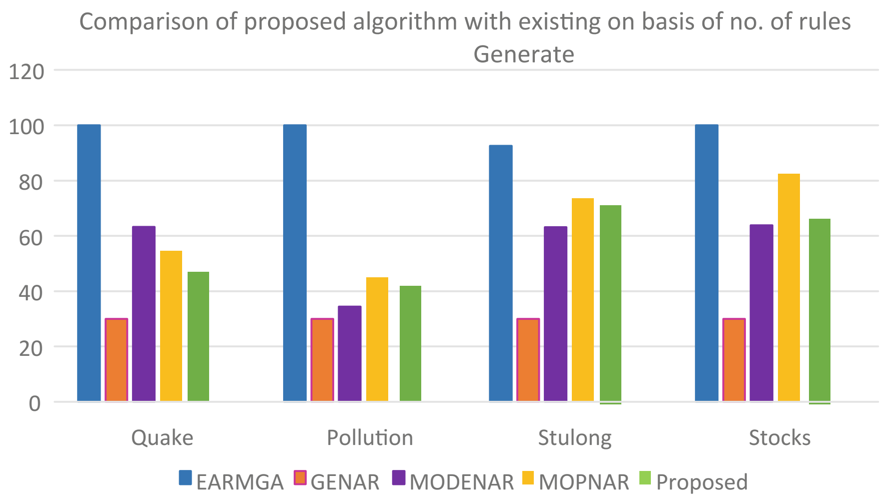


Fig. 10 Comparison of proposed algorithm with existing on basis of no. of rules generated

Table 3 Number of rules generated by different algorithms

	Quake	Pollution	Stulong	Stocks
EARMGA	100	100	92.6	100
GENAR	30	30	30	30
MODENAR	63.3	34.4	63.2	63.8
MOPNAR	54.6	45	73.6	82.4
Proposed approach	45	38	68	64

Table 4 Average support of different algorithms

Avg. support	Quake	Pollution	Stulong	Stocks
EARMGA	0.27	0.25	0.27	0.37
GENAR	0.55	0.22	0.88	0.29
MODENAR	0.36	0.27	0.52	0.48
MOPNAR	0.27	0.26	0.27	0.31
Proposed approach	0.47	0.41	0.35	0.42

shown above it can be concluded that our proposed algorithm results in better average support count than EARMGA, MODENAR, MOPNAR which signify that rules generated by our approach are more interesting than the rules generated by previous algorithms.

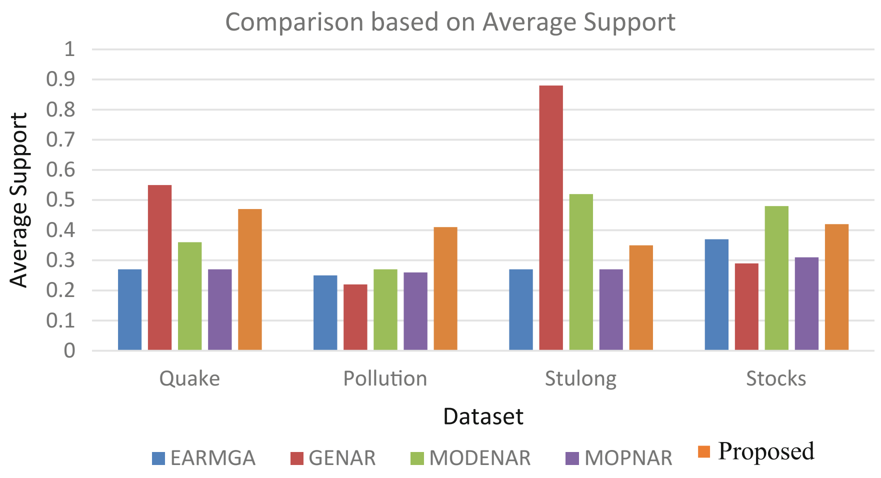


Fig. 11 Comparison of proposed algorithm with existing on basis of average support of rules generated

7 Conclusion and Future Scope

We have studied and analyzed the problem of association rule mining for numeric datasets. Processing of numeric dataset for getting information is a tedious and computationally complex task because of the nature of the data. Association rule mining for numeric dataset was a challenging task as the huge number of expected rules will be generated. Therefore, we needed a technique which runs for many solutions in parallel so that we can process huge number of rules and also have the tendency of improving its solution by itself for generation of more interesting rules. The pre-existing technique that follows these criteria is genetic algorithm which has the property of running for solution in parallel and improving their own solution in all iteration i.e. generation in terms of genetic algorithm. Therefore we have decided to solve the problem of association rule mining with genetic algorithm.

In this work, we have proposed an approach for mining numeric association rules using genetic algorithm and both the algorithms are implemented and analyzed using keel repository datasets. After implementation and analysis, we can conclude that genetic algorithm is a good approach to be used for mining numeric association rules as it analyzes multiple rules together at a time which decreases the probability of getting stuck in a local optimal solution. Further, in all iterations, best rules are more refined and new random rules joined the population which makes the scenario continuous improving i.e. even if we have started from a random initial population of rules but with all iterations rules get improved. Rather than using traditional methods for optimization i.e. derivative function etc., fitness functions are used for optimization which makes it more efficient.

In the current work we have focused only on generating association rules from testing datasets. It is still a long way to go for using this solution in real life scenario. To improve the current solution and embedding that solution to real world problems, various research opportunities could be:

1. Genetic algorithm is a robust technique for handling NP Hard problems and as association rule mining is also an NP Hard problem therefore we can say that we are going in correct direction. But it is still a long way to go as the size of the dataset increases more refine operators and fitness function have to be used. The future research direction for this work would be to develop more refined operators that could be used for better results.
2. Further Genetic algorithm can be applied to many real world problem some of which are discussed in Moin et al. [30], Barak and Modarres [31].We would like to test our proposed work on similar kind of real world problem.
3. Also one of the issues with the proposed algorithm is that with the increase in the size of dataset, the efficiency of the algorithm degrades i.e. proposed algorithms are not scalable enough. Therefore study regarding scalability of such algorithms can also be the future research direction of this work.

The proposed algorithm is efficient one and may be one of the possible candidates for the research community to get real time information from databases in smart city applications ranging from healthcare applications to intelligent transportation system [32–34].

References

1. Agrawal R, Imielinski T, Swami AN (1993) Mining association rules between set of items in large databases. In: ACM SIGMOD international conference on management of data, Washington D.C.
2. Tew C, Giraud-Carrier C, Tanner K (2013) Behaviour-based clustering and analysis of interestingness measures for association rule mining. Springer Data Min Knowl Discov J 28(4):1004–1045
3. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the twentieth international conference on very large databases, Santiago de Chile, Chile
4. Houtsma M, Swami A (1995) Set-oriented mining for association rules in relational databases. In: Proceedings of the 11th IEEE international conference on data engineering, Taipei, Taiwan
5. Hidber C (1999) Online association rule mining. In: Proceedings of the ACM SIGMOD international conference on management of data, Philadelphia, PA, USA
6. Borgelt C (2003) Efficient implementations of Apriori and Eclat. In: Proceedings of the workshop frequent itemset mining implementations, Melbourne, FL
7. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Publishing Co., Inc., Boston
8. Elhoseny M, Yuan X, Yu Z, Mao C, El-Minir H, Riad A (2015) Balancing energy consumption in heterogeneous wireless sensor networks using genetic algorithm. IEEE Commun Lett 19(12):2194–2197. https://doi.org/10.1109/LCOMM.2014.2381226

9. Yuan X, Elhoseny M, El-Minir H, Riad A (2017) A genetic algorithm-based, dynamic clustering method towards improved WSN longevity. J Netw Syst Manage Springer, US 25(1):21–46. https://doi.org/10.1007/s10922-016-9379-7
10. Obitko M (n.d.) Genetic algorithm tutorial. Retrieved from https://courses.cs.washington.edu/courses/cse473/06sp/GeneticAlgDemo/index.html
11. Freitas AA (2003) A survey of evolutionary algorithms for data mining and knowledge discovery. Advances in evolutionary computing. Springer, New York, NY, USA, pp 819–845
12. Ghosh A, Nath B (2004) Multi-objective rule mining using genetic algorithms. J Inf Sci Elsevier 163:123–133
13. Anand R, Vaid A, Singh PK (2009) Association rule mining using multiobjective evolutionary algorithms: strengths and challenges. In: Proceedings of the nature & biologically inspired computing, Coimbatore, India
14. Yan X, Zhang C, Zhang S (2009) Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. J Expert Syst Appl Elsevier 36(2):3066–3076
15. Alatas B, Akin E, Karci A (2008) MODENAR: multi-objective differential evolution algorithm for mining numeric association rules. J Appl Soft Comput Elsevier 8(1):646–656
16. Martin D, Rosete A, Alcala J (2014) A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules. IEEE Trans Evol Comput 18(1):54–69
17. Metawa N, Hassan MK, Elhoseny M (2017) Genetic algorithm based model for optimizing bank lending decisions. Expert Syst Appl 80:75–82
18. Berzal F, Blanco I, Sanchez D, Vila M (2002) Measuring the accuracy and interest of association rules: a new framework. J Intell Data Anal 6(3):221–235
19. Anand HS, Vinodchandra SS (2016) Association rule mining using treap. Int J Mach Learn Cybern. https://doi.org/10.1007/s13042-016-0546-7
20. Umit C, Bilal A (2017) Automatic mining of quantitative association rules with gravitational search algorithm. Int J Softw Eng Knowl Eng 27(3):343–372
21. Uroš M, Milan Z, Iztok F Jr, Iztok F (2017) Modified binary cuckoo search for association rule mining. J Intell Fuzzy Syst 32(6):4319–4330
22. Qodmanan HR, Nasiri M, Minaei-Bidgoli B (2011) Multiobjective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. J Expert Syst Appl Elsevier 38(1):288–298
23. Tan P, Kumar V, Srivastava J (2002) Selecting the right interestingness measure for association patterns. In: Proceedings of the 8th international conference of KDD, Edmonton, AB, Canada
24. Brin S, Motwani R, Ullman JD, Tsu S (1997) Dynamic itemset counting and implication rules for market basket data. In: Proceedings of the ACM SIGMOD international conference on management of data, New York, NY, USA
25. Magalhães-Mendes J, Cidem (2013) A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. WSEAS Trans Comput 12(4):164–173
26. Alcala-Fdez J, Fernandez A, Luengo J, Derrac J, Garcia S, Sanchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multiple Value Logic Soft Comput 17(2–3):255–287
27. Mata J, Alvaez J, Riquelme J (2001) Mining numeric association rules with genetic algorithms. In: Proceedings of the 5th international conference artificial neural network genetic algorithms, pp 264–267
28. Minaei-Bidgoli B, Barmaki R, Nasir M (2013) Mining numerical association rules via multi-objective genetic algorithms. J Inf Sci Elsevier 233:15–24
29. Al-Maqaleh B (2013) Discovering Interesting association rules: a multi-objective Genetic algorithm approach. Int J Appl Inf Sci 5(3):47–52
30. Moin NH, Sin OC, Omar M (2015) Hybrid genetic algorithm with multiparents crossover for job shop scheduling problems. In: Mathematical problems in engineering, Hindawi, 2015
31. Barak S, Modarres M (2015) Developing an approach to evaluate stocks by forecasting effective. Expert Syst Appl Elsevier 42:1325–1339

32. Darwish A, Hassanien A, Elhoseny M, Sangaiah A, Muhammad K (2017) The impact of the hybrid platform of internet of things and cloud computing on healthcare systems: opportunities, challenges, and open problems. J Ambient Intell Humanized Comput. First Online: 29 Dec 2017. https://doi.org/10.1007/s12652-017-0659-1
33. Elhoseny H, Elhoseny M, Riad AM, Hassanien A (2018) A framework for big data analysis in smart cities. In: Hassanien A, Tolba M, Elhoseny M, Mostafa M (eds) The international conference on advanced machine learning technologies and applications (AMLTA2018). AMLTA 2018. Advances in Intelligent Systems and Computing, vol 723. Springer, Cham. https://doi.org/10.1007/978-3-319-74690-6_40
34. Shehab A, Ismail A, Osman L, Elhoseny M, El-Henawy IM (2018) Quantified self using IoT wearable devices. In: Hassanien A, Shaalan K, Gaber T, Tolba M (eds) Proceedings of the international conference on advanced intelligent systems and informatics 2017. AISI 2017. Advances in intelligent systems and computing, vol 639. Springer, Cham. https://doi.org/10.1007/978-3-319-64861-3_77