MINING NUMERICAL ASSOCIATION RULES USING GENETIC ALGORITHM

Project report submitted in partial fulfillment of the requirement for the degree of

Master of Technology

in

Computer Science & Engineering

Under the Supervision of

DR. PARDEEP KUMAR

By

SHAILZA CHAUDHARY

132203

MAY 2015

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT-173234

CERTIFICATE

This is to certify that project report entitled "**Mining Numerical Association Rules Using Genetic Algorithm**", submitted by **Shailza Chaudhary** in partial fulfillment for the award of degree of Master of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Dr. Pardeeep Kumar
Assistant Professor
Senior Grade

ACKNOWLEDGEMENT

I would like to express my gratitude to people who supported me and helped me with the work on this dissertation. It was a great experience that I had during completion of this work.

First of all, I would like to thank to my supervisor **Dr. Pardeep Kumar** who has supported me for all the time in working of my thesis. He was always ready to give me a relevant feedback and especially a constructive critical view on my work, which is always very hard to gain.

I am also very grateful to **Prof. RMK Sinha**, Dean of Department of Computer Science and Engineering and **Prof. S.P. Ghrera** Head of Department of Computer Science and Engineering, Jaypee University of Information Technology for their immense motivation and relevant feedbacks, time to time.

Date: **Shailza Chaudhary**

TABLE OF CONTENTS

LIST OF ACRONYMS

PSize - Population Size

NGen - Number of generations

RMut - Mutation Rate

RCross - Crossover Rate

NAtt – Number of Attributes

NRec – Number of Records

MinSup - Minimum Support

Avg.- Average

MutRep - Mutation Repository

LIST OF TABLES

LIST OF FIGURES

Abstract

Mining Numerical Association Rule using Genetic Algorithm

Since the introduction, association rule mining technique became the most famous and widely used data mining technique because of its simplified nature of solution that it generates. Because of its robustness of deriving associations among various attributes of dataset it is used in various application areas for decision making, detection and prediction etc. Although the technique seems to be very easy in starting, especially when dealing with categorical data but becomes quite complex when it's time to deal with numeric data because of its diversity. This work is done to deal with problem of association rule mining from numeric data in an efficient way and aimed to generate more interesting rules from the dataset. For accomplishing this task we have used a well-known machine algorithm i.e. 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 two algorithm based on genetic algorithm with slight differences. These algorithms are also implemented and tested on various datasets and a comparison between proposed and existing work has also been illustrated.

Keywords: Association rule mining, decision making, Categorical data, numeric dataset, Genetic Algorithm

CHAPTER 1

INTRODUCTION

Overview

Motivation

Problem Statement

Organization

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Large amounts of data are being collected routinely in the task of day-to-day management and decision building in banking, business, administration, the delivery 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. Therefore, someone who wants relevant information from such big dataset need to have a robust, efficient, simple and computationally efficient algorithm to extract information from such datasets.

The creation, validation and improvement of such algorithms and make it understandable to user are the key activities of data mining since a long, which is completely dedicated to discovering of useful, hidden information from the ocean of data. That too retrieving that information efficiently has to be the key concern of data mining because it is the efficiency of the process that will actually makes the process implementable in real life. Various available data mining functions are classification, clustering, prediction, link analysis which benefits users of data handling for using large amount of data efficiently and also help them in deriving different type of inferences from such data. One of the important data mining tasks is mining 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 subject of computer science also helps data mining, i.e. machine learning. Machine learning basically 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 technique in our work i.e. genetic algorithm.

1.2 MOTIVATION

There is no doubt that association rule mining is an important task of data mining but generating rules from dataset is not that easy. To generate association rule one need to consider all possible combination of attributes of dataset to analysis relation between any set of attribute. With increase in number of attributes in datasets, possible combination also increases that too exponentially. Therefore association rule mining comes under category of NP-Hard problem because it has to generate and analyze a search space of 2^n where n is the number of attributes in dataset. As number of attributes increases, search space also increases. And when the value of attributes are real numbers i.e. numeric dataset and that too are to be considered this task becomes more complex and computationally intensive. Thus the motivation of numeric association rule comes from the need of analyzing data for retrieving if-then rules from a dataset with attributes having real number values to use such rules for various purposes such as decision making, prediction, etc. The main goal of generating rules from numeric dataset is to consider all possible and important correlation of various attributes in an efficient and faster way.

For example, consider a stationary shop maintaining logs of purchases of their customer such as customer 1 purchases a notebook and a pen, customer 2 purchases 2 notebook and one pen and so on. Let, at the end of month manager of stationary shop wants to analyze the behavior of customer. At that point of time let say 1000 persons visited his shop in last month so he will be having a total of 1000 records. But generating any inference from such records will be difficult for the manager therefore he need to have some mechanism that can help him in generating some results which are easily manageable and understandable. This is a situation where association rule mining can play an important role. Because information derived from numeric association rule mining is represented in the form:

IF (2-5 Notebooks) are purchased $\longrightarrow$ THEN (2-4 Pens) will be purchased. OR

IF (4-5 Books and 2-4 Notebooks) are purchased $\longrightarrow$ THEN (7-10 Pencils) will be purchased.

Which is easily understandable for the manager of stationary shop.

For generating such rules data mining will also require a help from the machine learning techniques which have solution to almost all possible problem which comes in computation results from computer. There are various algorithms and techniques available in machine learning among them we will be considering Genetic Algorithm.

The reason behind selecting genetic algorithm is that it interacts better with various attributes and performs a global search for generating result. General benefits of using genetic algorithm are:

- It has capability of solving every optimization problem. It is said that genetic algorithm always generates a result.
- It can solve the problem having many solution and it can generate all such solutions.
- It can solve a problem by taking into account multiple objectives and can optimize multiple objectives simultaneously.
- It is an easy to understand concept, without much of the domain knowledge it can generate results.

1.3 PROBLEM STATEMENT

Formally the problem of mining numeric association rule can be described as:- Given a set of records in dataset with some attributes. Find all the rules of the form:

attribute1[lb1,ub1],attribute2[lb2,ub2] $\longrightarrow$ attribute3[lb3,ub3],attribue4[lb4,ub4].

Which signifies if the value of attribute1 is between lb1-ub1, attribute2 is between lb2-ub2 then the value of attribute3 will be between lb3-ub3 and attribute4 between lb4-

ub4. For verifying the validity of such rules, the rules must satisfy some parameters which are called interestingness parameters in association rule mining context. Some of the predefined and widely used interestingness parameters are Support, Confidence, Leverage, Lift, Conviction etc there are many other interesting measure parameters are available a detailed description of 65 such parameters are discussed in [28].

Since decades association rule mining is famous among data mining experts because of its simplicity in understanding and usage with the robustness they carry to represent some information. But still there are some of the issues in association rule mining, which get solved, but reappears when we change volume of data, type of data or usage of data. Our work also tackle one of such problem that is occurred due to the type of data to be considered in the problem. We will mainly be concentrating on handling numeric data because as a shift of data from categorical to numeric is considered search space of the problem becomes very large that is why traditional algorithms cannot be used for handling such kind of data.

Some of the common problem that comes across mining numeric datasets are as follows:

- Handling the variety of numeric data, as the data is spread on the real number line it is important to carry some work out of handling the data such as by making intervals. But deciding intervals and interval length is also not easy [20][21].
- For validating rules for interestingness, various interestingness parameters have to choose from and for each interestingness parameter we have to decide a threshold value. Deciding which interestingness parameter[23] should be selected and what will be there threshold is also a big task to do.
- As discussed earlier, the search space for numeric data is quite large dealing with that is highly computationally expensive. We can’t generate all the combination of attributes with all possible intervals, so a rigorous study is needed on handling computation cost.
- Generating rule is easy, but generating and identifying interesting rules is a task of utmost importance when we talk about association rule mining. To

generate and identifying selective rules is also a key task in association rule mining.

- Most of the algorithms that are discussed in the literature either consider categorical data only or numeric data only. No such model exists in literature which combines the categorical as well as numeric data together.

1.4 ORGANIZATION

Further this dissertation is organized as follows. Chapter 2 discusses basic terminology like association rule mining, genetic algorithm and multi objective optimization. Chapter 3 explains the literature survey that we have gone through so far, which explains and compare various algorithm on association rule mining proposed by different authors. Chapter 4 enlightens the proposed work that we have implemented in this dissertation, which includes two proposed algorithms and its details. Chaper 5 explains the implementation setup that we have used for this dissertation. Chapter 6 contains the result and discussion of implemented algorithm and its comparison with earlier works. At last, Chapter 7 discusses conclusion and future scope of this work.

CHAPTER 2

TERMINOLOGY

Association Rule Mining

Genetic Algorithm

Multi-Objective Optimization

CHAPTER 2

TERMINOLOGY

2.1 ASSOCIATION RULE MINING

Association rule mining is an important data mining task that generates rules in form of IF-THEN statement from given dataset. It does this by considering all possible combination of attributes called as itemset and find the occurrences of various itemsets in dataset. More number of times itemset appears in dataset more is the importance of that itemset. This task of finding itemset which are occurring in dataset more number of times are called frequent itemsets. Once the frequent itemsets are generated next step is to generate combination of frequent itemset among themselves and again considering occurrences of these combination. The combination that generated now are called association rules.

Formal definition of association rule problem as stated in [1]:

Definition 1: Let $I = \{I_1, I_2, \dots, I_n\}$ be a set of n different attributes of a dataset, also called literals. Let D be a dataset, where each record of dataset is called R. This dataset consist of a set of attributes such that $R \subseteq I$. An association rule is represented in the form $X \Rightarrow Y$, where $X, Y \subset I$, are sets of attributes referred as itemsets, where $X \cap Y = \emptyset$. Here, X is known as antecedent, and Y as consequent.

To check for the validity of rule in terms of interestingness there are two important measures for association rules to check interestingness, support (s) and confidence (α), which can be described as below.

Definition 2: The support count measure (s) of an association rule is the ratio (generally in %) of the records that have union of antecedent and consequent ($X \cup Y$) to the total records in the dataset.

Therefore, if it is said that the support count of a rule is 40% then it means that 40% of the total records comes with combination of X and Y.

Definition 3: For a given set of records, confidence (α) is the ratio (in %) of the number of records that have union of antecedent and consequent ($X \cup Y$) to the total records that contain only antecedent part (X).

Thus, if it is said that a rule has a confidence of 90%, it means that 90% of the records having X also have Y in the same record. The confidence of a rule signify the degree of correlation among attributes of the dataset, i.e. between X and Y. Confidence is an indication of how important a rule is for the given dataset. Generally a large confidence is desirable for association rules. Mining of association rules from a dataset aimed at finding all the rules that satisfy the user specified interestingness parameter's threshold i.e. support and confidence here. The task of association rules mining can be split into two subtasks [3]. First, finding itemsets which are occurring more frequently, i.e. frequent itemsets. The second task is to find association rules by combining frequent itemsets obtained in the earlier step.
To get a clearer view of how association rule mining work here is an example.

Example: Let's take a small dataset with 4 items (attributes) I= {Notebook, Pen, Books, Pencil} and four transactions logs as shown in Table 1. All possible itemsets for set of items (I) are illustrated in Table 2. Let's take that the minimum support and minimum confidence for current example are 40% and 60%, respectively. 'When generated there will be several association rules will be generated which could be useful or could be of no use. Here, for discussion purpose I have taken the itemsets shown in Table 3.

First, we have to find out the sets of items those occurrence is more i.e. which is more frequently coming in dataset. Second task is to verify whether the rule generated have the confidence of at least 60%. If the above both conditions are verified for a rule, we can consider that this rule is of use for the system for which we are generating rules. Now, Itemsets associated with the generated rules are: {Notebook, Pen}, and {Pen, Books}. The support for each individual itemset is at least 40% as in Table 2. Therefore, all of these itemsets are considered frequent. The confidence for rule generated by frequent itemset is shown in Table 3. It can be seen that the first rule (Notebook $\Rightarrow$ Pen) is valid. Whereas, the second rule (Pen$\Rightarrow$Books) is not valid because its confidence is less than 60%.

Transaction ID	Items
T1	Notebook, Pen, Books
T2	Pen, Books, Pencil
T3	Pen
T4	Notebook, Pen

Table 1: Dataset of transaction logs for Example

Itemset	Support, s	Frequent/Infrequent
Notebook	50%	Large
Pen	100%	Large
Books	50%	Large
Pencil	25%	Small
Notebook, Pen	50%	Large
Notebook, Books	25%	Small
Notebook, Pencil	0%	Small
Pen, Books	50%	Large
Pen, Pencil	25%	Small
Books, Pencil	25%	Small
Notebook, Pen, Books	25%	Small
Notebook, Pen, Pencil	0%	Small
Notebook, Books, Pencil	0%	Small
Pen, Books, Pencil	25%	Small
Notebook, Pen Books, Pencil	0%	Small

Table 2: Support of Itemsets in Table 1, Support of 40%

Rule	**Confidence**	**Valid Rule**
Notebook ⇒ Pen	100%	Yes
Pen ⇒ Notebook	50%	No
Pen⇒Books	50%	No
Books⇒Pen	100%	Yes

Table 3: Confidence of some rules with Confidence=60%

As it can be seen from the example that, mining rule like this from dataset can be very tedious and computationally expensive, therefore it cannot be used for real life datasets. For doing such a task many algorithms can be found in literature. Some of them are discussed in the literature survey section.

2.2 GENETIC ALGORITHM

Many different algorithm can be found in the literature that is suitable for mining association rules from a dataset but most of the algorithm are either slow or need domain knowledge of the problem. To deal with this problem there is a algorithmic concept in machine learning that we can select for rule mining i.e. genetic algorithm. It has the capability to reduce the computation cost of generating rules addition to that it also improves its solution in each iteration.

Genetic Algorithm (GA) [2] is a powerful algorithmic technique that can be used for generating solution for a search and optimization problem. It basically follows one of the nature's principle i.e. survival of the fittest as stated in Darwin's theory of evolution. Which tells that the one that is stronger survives for longer on the same concept this algorithm finds better and better solution in each generation and keeps on improving its own solution in each generation and it keeps on doing this till it kinds a near optimal solution.

To have a more clear view of how genetic algorithm works and what it need for processing a problem some of the key terminology is needed to be understand. The terminology used in genetic algorithm is described next.

2.2.1 Encoding

The first step of implementation in GA is representation of solution of problem in a string format which is called encoding. It basically encodes the solution to be generated in a format that can be used in the process of algorithm further. Various encoding scheme are available to be used but the selection of the encoding to be used is dependent on the problem to be solved.

Example of one of such encoding scheme is:

Binary encoding

Solution in terms of chromosomeI: 110011001100

Solution in terms of chromosomeII: 000010101010

Let say a tuple of dataset is t be selected from the dataset that is best. In that case chromosome represented in the algorithm indicates that there are 12 attributes in the dataset which ae represented as 1 if that attribute is present in that tuple and 0 if that attribute is not present in the tuple. In chromosome I above represents 1st, 2nd, 5th, 6th, 9th and 10th attribute present in the tuple. Other encoding scheme available to be used are real encoding, permutation encoding, value encoding, tree encoding etc. Depending on the nature of the problem and the type of solution to be found any of these encoding scheme can be used. After encoding the next step in GA is selection of the chromosome among various chromosomes.

2.2.2 Selection

In selection process, the key task is to decide which chromosome to select so that a better result can be found. As stated in Darwin's theory of evolution the strongest individual survives for long and they are the ones who takes part in generation of offspring's with the same concept here we need to find best chromosome which are more relevant to the solution generation process. To select such chromosome one of the widely considered selection process is Roulette wheel selection process, which select the solution using stochastic sampling technique with replacement i.e. every chromosome in the generation has some probability to get selected based on some fitness value. Fitness value can be calculated from the relevancy of that chromosome to the problem. Some of the other techniques for selection are tournament selection, steady state selection and rank selection. After the selection of chromosome from the population the next step is to apply crossover and mutation on these chromosomes to generate more chromosome for next generation.

2.2.3 Crossover

Crossover is the process of mixing the occurrence of attribute of two chromosome of previous generation to generate two new chromosomes that have properties of both the selected chromosome used for crossover. Basic concept behind the crossover process is that the chromosome formed by mixing two parent chromosome will have good property of both the parents hence the chromosome formed by crossover will perform better than their parent chromosomes.

Example of crossover:

('|' is the crossover point):

Chromosome A=10011 | 10101110111

Chromosome B=**11001 | 01010011010**

Child A=10011 | **01010011010**

Child B=**11001** | 10101110111

Based on selected crossover point various other crossover operator discussed in literature [32] are single point crossover, multipoint crossover, Arithmetic crossover, uniform crossover etc.

After crossover the next step is to apply mutation operator for introducing variety in there solution generation process.

2.2.4 Mutation

The main reason behind using mutation operator is to implement some level of diversity in the population of next generation. So that process of GA can be stopped from entering into a local optimum solution and getting stuck into it. Mutation does this by adding some more changes in the crossover chromosomes to get more chromosome with some diversity from previous ones.

Example for mutation process:

Chromosome A=010**0**101000010011

Chromosome B=010110**1**110110111

Mutated chromosome A=010**1**101000010011

Mutated chromosome B=010110**0**110110111

There is a term involved in mutation i.e. mutation rate. It is the rate by which the diversity will be included in the system. This rate should be kept moderate as high value of this rate will introduce randomness and low value of this rate will not serve the purpose of introducing diversity to the population.

Significance of crossover and mutation:

Crossover and mutation have a rate associated with it. These rates signify the frequency at which the diversity will be introduced in the previous generation population. Crossover rate signify the number of times crossover operator will be applied on a chromosome in a generation. Similarly mutation rate signify the number of time mutation will be applied in a population. Value of crossover and mutation rate lies between 0% - 100%. Where 0 indicates no change in population and 100 indicate complete change in population. To keep the evolution property of the genetic algorithm these rate has to be kept moderate.

To get a clear view of how genetic algorithm actually works next is a flowchart of general workflow diagram of genetic algorithm, which illustrates how the above discussed terminology exactly fits in the genetic algorithm. The flow starts with the description of problem, on the basis of problem and the solution needed an encoding scheme is selected. By using that encoding scheme chromosomes are generated as initial population. For each chromosome fitness value is calculated 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 in each 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.

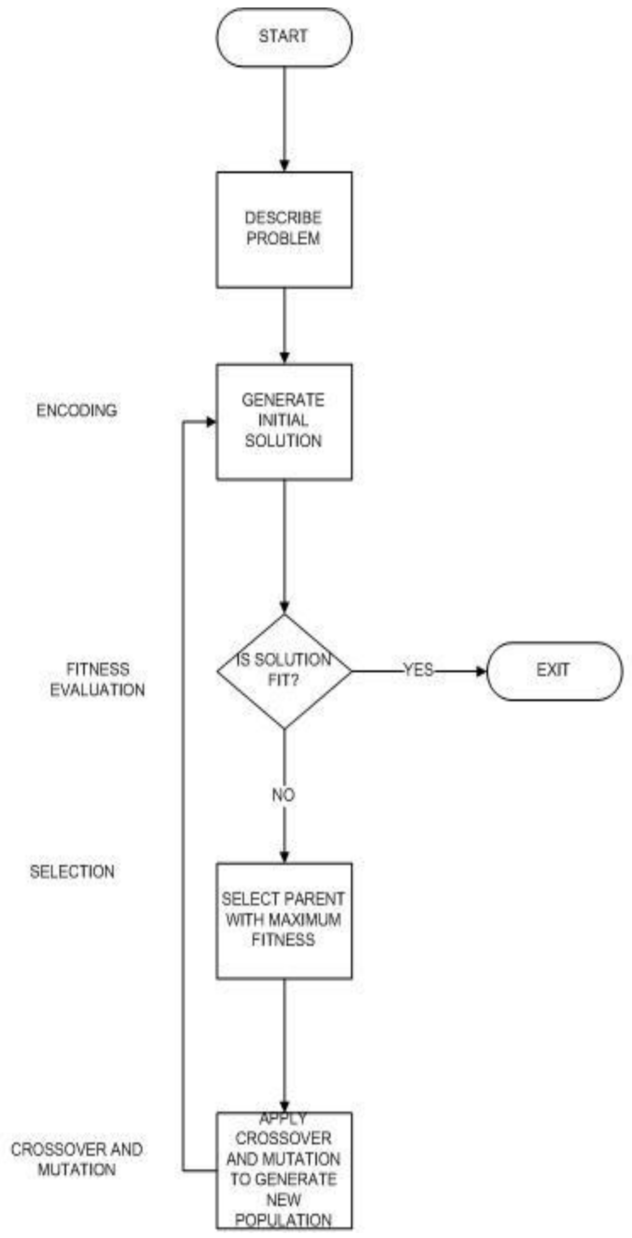
START
DESCRIBE PROBLEM
ENCODING
GENERATE INITIAL SOLUTION
FITNESS EVALUATION
IS SOLUTION FIT?
YES
EXIT
NO
SELECTION
SELECT PARENT WITH MAXIMUM FITNESS
CROSSOVER AND MUTATION
APPLY CROSSOVER AND MUTATION TO GENERATE NEW POPULATION

Figure 1: Flowchart of General Genetic Algorithm

GENERAL GENETIC ALGORITHM FLOW [26]

1. **[Start]** Generate random initial population of chromosomes
2. **[Fitness]** Calculate the fitness value of every chromosome of population
3. **[Next population]** Generate the next population by iterating following steps until the complete next population is generated
    1. **[Selection]** Select two chromosomes from a population having highest fitness.
    2. **[Crossover]** With certain crossover probability cross over the selected chromosome to form a new chromosome.
    3. **[Mutation]** With certain mutation probability mutate new chromosomes at some position of the chromosome.
    4. **[Accepting]** Place new chromosome in the next generation population
4. **[Replace]** Consider next generation population for further processing of the algorithm
5. **[Test]** If the target condition is fulfilled, **stop** the execution, and return the solution present in current population
6. **[Loop]** Go to step **2**
7. **[End]**

2.3 MULTIOBJECTIVE OPTIMIZATION

Optimization is a technique which works for finding one or more feasible solutions which are bounded by some extreme constraints of one or more than one objectives. When the optimization problem works only for one objective function then it is called *single objective optimization* and if it has more than one objective functions then it is known as *multi objective optimization*. Each optimization problem definition consists of the following basic components:

1. Objective function: It represents the quantity/expression to be optimized, i.e., the quantity to be maximized or minimized. Let *f* denotes the objective function. Which we have to maximize or minimize as per requirement.

2. A set of variables or unknowns: These are the values which affects the objective function. For example, If x represents the unknowns, then $f(x)$ defines the quality of the candidate solution x.
3. A set of constraints: It bounds the values that can be assigned to the variable/unknowns. Mostly these boundary constraints, defines the domain of values for each variable.

In most of the problems with more than one conflicting objective, there does not only exist any single optimal solution rather there exists numbers of solutions which all are optimal. Then there may exist trade-offs between different objectives. In that case a solution that is good with respect to one objective may requires some compromise in terms of some other objectives. One solution cannot be treated to be better than another without any further information. There are basically two main optimization procedure exists i.e. Ideal multi-objective optimization and preference based multi objective optimization [19].

In Ideal approach, first locate multiple optimal solutions with trade-off in a wide range of values for objectives, and then chooses one best solution using high level information.

Whereas in preference based technique a combination of multiple objective function is first constructed as a weight sum of the objectives, then it is solved as single objective optimization problem for simplicity. Generally, the weight of an objective is directly proportional to the preference factor assigned with that specific objective.

The general definition of a multi-objective optimization problem can be stated as Minimize/Maximize: Objective function $f_m(x)$, m=1, 2 ...M;

$$\text{subject to } a_j(x) \geq 0 \; j=1, 2, \ldots, j; \; b_k(x) = 0 \quad k=1,2,\ldots,k;$$

$$x^L_i \leq x_i \leq x^U_i \quad i=1, 2, \ldots n.$$

Where, a solution x is a vector of n decision variables, $x = x_1, x_2, \ldots x_n^T$. Each decision variable x_i can take a value within a lower bound x^L_i and upper bound x^U_i. These bounds constitute a decision variable space D. A solution x, that satisfies all constraints and variable bound is known as feasible solution. We are interested in this feasible solution for multi-objective genetic algorithm. Where objectives are support, confidence, lift and conviction and we have to optimize the effect of these four for generation of association rule.

CHAPTER 3

LITERATURE REVIEW

Classical Association Rule Mining Algorithms

Association Rule Mining using GA

CHAPTER 3

LITERATURE REVIEW

3.1 CLASSICAL ALGORITHM FOR ASSOCIATION RULE MINING

3.1.1 AIS

The AIS algorithm was the very first published algorithm for mining association rules by generating all large itemsets in a transaction dataset [1]. This technique only generates rules with only one item in the consequent part. That is, the association rules generated by this process are in the form of $X \Rightarrow I_j \mid \alpha$, where X is a set of attributes and I_j is a single attribute in the set I, and α is the confidence parameter of the generated rule.

3.1.2 SETM

SETM algorithm was proposed by [5] with aim of using SQL for obtaining frequent itemsets. In this algorithm each member of the large itemset keeps information of TID i.e. unique ID of the record as <TID, itemset>. Further process of this algorithm is similar to that of AIS.

But shortcoming of this algorithm is also the same as AIS i.e. it also generates large number of candidate sets and require multiple read scans of dataset and arranging of data again and again.

3.1.3 APRIORI

The apriori algorithm presented by [3] was the major milestone in association rule mining area. The work presented in this uses the axiom that any subset of a frequent itemset must be a frequent itemset. Hence it generates the candidate itemsets of next pass by merging the large itemsets of the previous pass and eliminating all small subset of previous pass as well. Therefore it can be concluded that apriori performs much better than the previous two algorithm as it works with much less candidate sets and is much faster and efficient.

3.1.4 CARMA

CARMA stands for Continuous Association Rule Mining Algorithm [6] works with the concept of computation of frequent itemsets online. Bring them online, CARMA directly shows the present association rules to the user of dataset in real time which also authorize the user to make relevant change in the parameters as per the need such as minimum support or minimum confidence, for any transaction at the first scan of the dataset. It requires at most 2 dataset scans. It generates the itemsets in the prior scan and does the counting work for all the itemsets in the next scan.

All these algorithms and many others published in the past, no doubt perform well in generating association rules which are accurate but they are computationally intensive and most of them generate much more rules than needed. These algorithms also don't consider numeric values of the attributes so, to make things easier we thought of studying mining algorithm which uses genetic algorithm as there basis. Because as per the discussion in the introduction part, Genetic algorithm is meant for doing the computationally intensive task fast and more efficiently which is the need for rule mining in large databases. Some of these algorithms earlier algorithm which uses this concept are discussed further.

3.2 ASSOCIATION RULE MINING USING GENETIC ALGORITHM

Various algorithms has been proposed so far for association rule mining from databases using genetic algorithm. Based on type of association rules generated, these algorithm can be categorized as categorical association rule mining algorithm and numeric association rule mining algorithm.

3.2.1 Categorical Association Rule Mining Algorithm

Categorical association rules are the IF-THEN rules generated from a categorical or a binary dataset.

In a binary dataset, a rule like $A,B,C \Rightarrow D,E$ signifies that:

If items *A*, *B*, and *C* are bought together, then items *D* and *E* will also be purchased. But, these rules do not indicate anything about the number of items that can be purchased; these rules simply indicates that whether the attribute is present or absent from the rule.

For applying GA for finding such rules[30], 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 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. Various algorithms discussed based on such bifurcation:

a) Chromosome Representation:

Broadly there are two chromosome representation techniques are present in literature for categorical 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 were each rule is good in itself. Therefore, for ARM case, the Michigan approach [2] is mostly used, in Michigan approach each chromosome represents exactly one rule in itself, is more suitable. Most of the ARM techniques use Michigan chromosome representation.

In an early work [7], the authors uses the Michigan approach as follows: each chromosome has length 2*k*, 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 [10], 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 (ARMGA) [9], 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 *i*th item, then the chromosome {3 | 5 4 2 1 3} represents the rule $A2, A5, A4 \Rightarrow A1, A3$. This kind of representation reduces length of the chromosome significantly.

b) Objective Functions:

Even though support count and confidence are two well-known objectives that are generally to be maximized, there are several other parameters available to measure the interestingness or strength of association rules[24]. 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. Mathematical notations for various such interestingness parameters are given in Appendix I. Each function has its own significance and combination of best of these functions gives best result in the end.

c) Evolutionary Operators:

Mostly, when binary encoding scheme is used, some of the standard crossover and mutation operators are used. For example, in [7], bit-flip mutation and multipoint crossover have been used. In [10] mutation operator, bit flip has been adopted, however, the authors did not specifically mention which crossover operator should be used.

In [11], where integer encoding for the chromosomes is used, an order-1 crossover technique is taken in 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.

d) Final Selection:

After appropriate iteration of the algorithm the set of rules generated are treated as final selected rules. On the basis of all the above parameters the interestingness and importance of rule's depends.

3.2.2 Numeric Association Rule Mining Algorithm

Numeric association rule can be represented mathematically in the form:

$(l1 \leq A1 \leq h1) \wedge (l2 \leq A2 \leq h2) \Rightarrow (l3 \leq A3 \leq h3)$.

Here *Ai* represents the *i*th attribute. *li* and *hi* represent the lower and upper bound of the attribute values, respectively. Thus, [*li, hi*] defines an interval of values for the attribute *Ai* with lower bound and upper bound.

a) **Chromosome Representation:**

The chromosomes are needed to encode the lower and upper bounds of the intervals of the attributes participating in a rule representing for numeric or quantitative association rules. In [8], where the MODENAR is proposed, the following encoding technique is adopted for the chromosomes representation. They used chromosomes where each attribute has three part. 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. A similar encoding scheme is adopted in [12]. The only difference is that in this case, the prior part of the chromosome, instead of taking the values as 0, 1, 2, adopts the values 0, 1, and −1, respectively, to denote the same meaning. In both cases, the algorithms used a Michigan encoding technique, that is, each chromosome encodes one rule.

In [14] also a three part chromosome is present where first number represents lower bound and second number represents upper bound of the attribute. With this there is a tag bit which represents position of attribute in rule i.e. if two tag bits are 00 then the attribute present in antecedent part of rule, if it is 11 then the corresponding attribute is present in the consequent part and if tag bits are 10 or 01 that represents the absence of attribute in rule. For example , A(00)(2.1)(3.5) B(11)(1.6)(4.5) C(01)(73)(73) represents if(A is between 2.1 and 3.5) then (B is between 1.6 and 4.5) and C is not present in rule.

b) Objective Functions:

MODENAR optimizes four criteria of the rules [8]: support, confidence, amplitude of the intervals and comprehensibility, which make up the subparts of the rule. Comprehensibility is the measure that is used to forces the search procedure to get shorter rules, this is done due to the assumption that relatively shorter rules generates more non redundant information. They also suggested that the amplitude of the intervals should be smaller for interesting rules, but the reason for this is not explained.

In [12], three major interestingness parameters are simultaneously optimized: lift, comprehensibility, and performance. Where Performance is calculated by the product of confidence and support measure. Lift is calculated as the ratio of support count of the rule to the product of the supports of the antecedent and the consequent of the rule. A higher value of the lift measure signifies that the rule is more interesting, since its support is high with respect to the supports of its antecedent and its confidence.

The comprehensibility can be defined simply as the reciprocal of the number of attributes in the rule.

In [14], again three objective functions are discussed that are confidence, comprehensibility, and interestingness. First two objective was similar to above and third objective that is interestingness considered to find rules that are more interesting or useful for the user, rather than all possible rules

c) Evolutionary Operators:

MODENAR [7] uses the standard crossover and mutation operators introduced by the approach of differential evolution. Moreover, this approach also introduced a rounding operator which round's off the first part of the attribute which should have an integer value (0, 1, 2) for evaluating the objective function values. In [12], multipoint crossover is used as crossover operator. Further, the two parts of the chromosome uses two different mutation operators for mutation. In the first part, where the value of chromosome are -1, 0, or 1, a random value is choosen from the set {-1, 0, 1} and it replaces the existing value. For the other part of the chromosome which consist of the lower and upper bounds of the chromosome, a random mutation is applied.

In [14], the attribute part of the rule i.e. (A, B….) remains fixed for the crossover or mutation step. K – Point crossover is used in this paper and for mutation, swapping of two tag bits is taken by using bit-flip mutation. This means that the existing bits pairs (00, 01, 10, 11) are randomly transformed into each other. Same is taken for the lower and upper value they are also changed randomly.

d) Obtaining Final Solution:

All [8], [12] and [14] use a Michigan approach of rule mining by encoding one rule in one chromosome. Thus, the final non-dominated set gives a set of numeric rules. Thus, there is no need to select any particular solution from the final non-dominate set. All the solutions will serve as the final selected rule set.

Comparison of various works proposed by different authors so far is give here in table 4 and table 5.

Table 4 shows comparison of some more algorithms based on encoding and operators. Table 5 shows specifications and shortcomings of these algorithms in brief.

	TYPE	**ENCODIG**	**CROSSOVER**	**MUTATION**	**FITNESS FUNCTION**
Ghosh and Nath(2004) [7]	Categorical	Binary (Michigan)	Multi-point	Bit flip	Confidence, Comprehensibilty, intersetingness
Alatas et. al. (2008) MODENAR [8]	Numeric	Mixed (Integer + Real) (Michigan)	Single point	DE/Rand /I	Support,Confidence, Comprehensibility, amplitude of interval
Xiaowei Yan et. al. (2009) ARMGA[9]	Categorical	Integer (Michigan)	Two point	Probability based Random Replacement	Relative Confidence
Anand et. al. (2009) [10]	Categorical	Binary (Michigan)	Not Mentioned	Bit-Flip	3 at a time among Support,Confidence, Interest, cosine, Comprehensibility.

Qodmanan et. al. (2011) ARMMGA [11]	Categorical	Integer (Michigan)	Order-I	Random Replacement	Support , Confidence
Martin et. al. (2011) NSGA-II-QAR [12]	Numeric	Real valued (Michigan)	Multi-point	Random increase decrease	Lift, Comprehensibility, Performance
Vashishtha et. al. (2011) [13]	Categorical	Binary (Michigan)	Fixed Single Point	Bit-Flip	Precision , Coverage , Generality
B. Minaei et. al. (2013) MOGAR [14]	Numeric	Real valued (Michigan)	k-point crossover	Random tag mutation	Confidence, Comprehensibility , Interestingness
Didel et. al. (2013) [15]	Categorical	Binary (Michigan)	Single Point	Bit-flip	Confidence*Support
Al-Maqaleh (2013) [16]	Categorical	Integer (Michigan)	Single Point	Random Insertion/Deletion	Support,Confidenc e,Simplicity
Diana Martin et.al. (2014)MOPNAR[17]	Numeric	Positional (Michigan)	Multi point	Random bit mutation	Comprehensibility ,Interestingness, Performance

Suryawanshi Rai et. al. (2014) MIPNAR_GA [18]	Categorical	Binary (Michigan)	Single Point	PCA mutation	$m(s) = \frac{A_i}{} W_i + \frac{}{L * (1 - Wi)} B_i$ A_i= Frequent item support B_i = Infreqent data W_i= Level of weight value of MLMS

Table 4: Comparison of Various Algorithm Based On Designing Factor

ALGORITHM	SPECIFICATION	SHORTCOMINGS	REMARKS
Ghosh and Nath(2004) [7]	Less rules generated	Require user's involvement	-
Alatas et. al. (2008)MODENAR [8]	• Simple and easy to implement • Intervals are selected on the go. • Database independent	Increase in no. of attributes and there distinct values may cause problem	-
Xiaowei Yan et. al. (2009) ARMGA [9]	• No Minimum Support required • Fast in execution time • System Automation • Database independent	No.of rul generated are more	Strict fitness function required
Anand et. al. (2009) [10]	• Better if rule length is smaller	Increase in no. of attributes could be problematic	-
Qodmanan et. al. (2011) ARMMGA [11]	• No min. support or confidence is required from user • Flexible on changing fitness • Best population is generated in each generation • Best rules are generated at the end	-	To be generalized for finding variant length rule.

D. Martin et. al. (2011) NSGA-II-QAR [12]	• Good trade-off between interpretability and accuracy • Evolutionary Learning of Intervals of attributes	Increase in no. of attributes could be problematic	Can be generalized for negative association rule
B. Minaei et. al. (2013) MOGAR [14]	• Steps are taken to remove weak association rules	Rough Intervals	-
Didel et. al. (2013) [15]	• Easy Concept	• More no. of rules generated • Increase in no. of attributes could be problematic	Better fitness function should be used
Diana Martin et. al. (2014) [17] MOPNAR	• Mine Positive and Negative QAR • Low computational cost • Good Scalability	Increase in no. of attributes could be problematic	-
Suryawanshi Rai et. al. (2014) MIPNAR_GA [18]	• Both positive and negative association rule mining. • Uses Improved Genetic Algorithm concept. Uses multilevel minimum Support.	Require Min. Support and Confidence from user. Increase in no. of attributes could be problematic	Generation of large rules needs to be managed. Efficiency can be improved

TABLE 5: Specifications and Shortcomings of Various Algorithms

CHAPTER 4

PROPOSED APPROACH

Details of implementation I

Details of implementation II

CHAPTER 4

PROPOSED APPROACH

In the literature, till date 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 two algorithm which generates better results from the previously known works of similar kind.

We have implemented the proposed two algorithms, in one implementation we have generated a random initial population and applied basic genetic algorithm work flow on the initial population and in later steps calculated the fitness of the chromosomes of the population using interestingness parameters Support, Confidence, Lift, and Conviction. The difference in this implementation from previously known algorithm lies in the fitness function used, crossover and mutation operators used. In the other implementation rest of the thing are kept similar to that of implementation I but the initial population taken is generated using careful analysis of available dataset i.e. on the basis of mean value of attributes so that we can start from the better initial population. For analysis purposes, we have implemented these algorithms on various datasets to verify results and later we have compared these results from previously known algorithms. The proposed algorithms are explained in the next section.

4.1 IMPLEMENTATION I

Implementation I is a direct implication of genetic algorithm concept for association rule mining where chromosomes are generated randomly and are improved in each iteration using crossover and mutation. With the modification in crossover and mutation operators and using power function as optimizing function.

4.1.1. Detailed Explanation of Implementation I

Detailed explanation of each Steps to perform in Implementation I is as follows:

Inputs:

PSize - Population Size

NGen - Number of generations

RMut - Mutation Rate

RCross - Crossover Rate

Output : Association Rules

Step 1: Initialization

i. Take an input dataset which contains number of attributes and records. Provide 'NAtt' and 'NRec' for the dataset.

ii. Generate an initial population having 'PSize' chromosomes on the basis of minimum and maximum value of attributes.

iii. Calculate Fitness of the initial population

Step 2: Introduce Diversity

i. Crossover: Apply crossover operator on chromosomes which satisfy fitness constraint.

ii. Mutation: Apply mutation operator on chromosomes which doesn't satisfy fitness constraint.

Step 3: Change Population

i. Calculate the fitness value for new chromosomes introduced by Step 2.

ii. Compare average fitness of last generation to the fitness of current chromosome.

iii. If fitness value of new chromosome is higher than the average fitness of last generation than consider that chromosome as fit therefore keep it in next generation.

Step 4: Iteration

i. **Repeat** Step 2-Step4 'NGen' number of times.

Step 5: Generate Rules

Decode the chromosomes found in last generation as they are the final rules of the dataset.

4.1.2. FlowChart for Implementation I

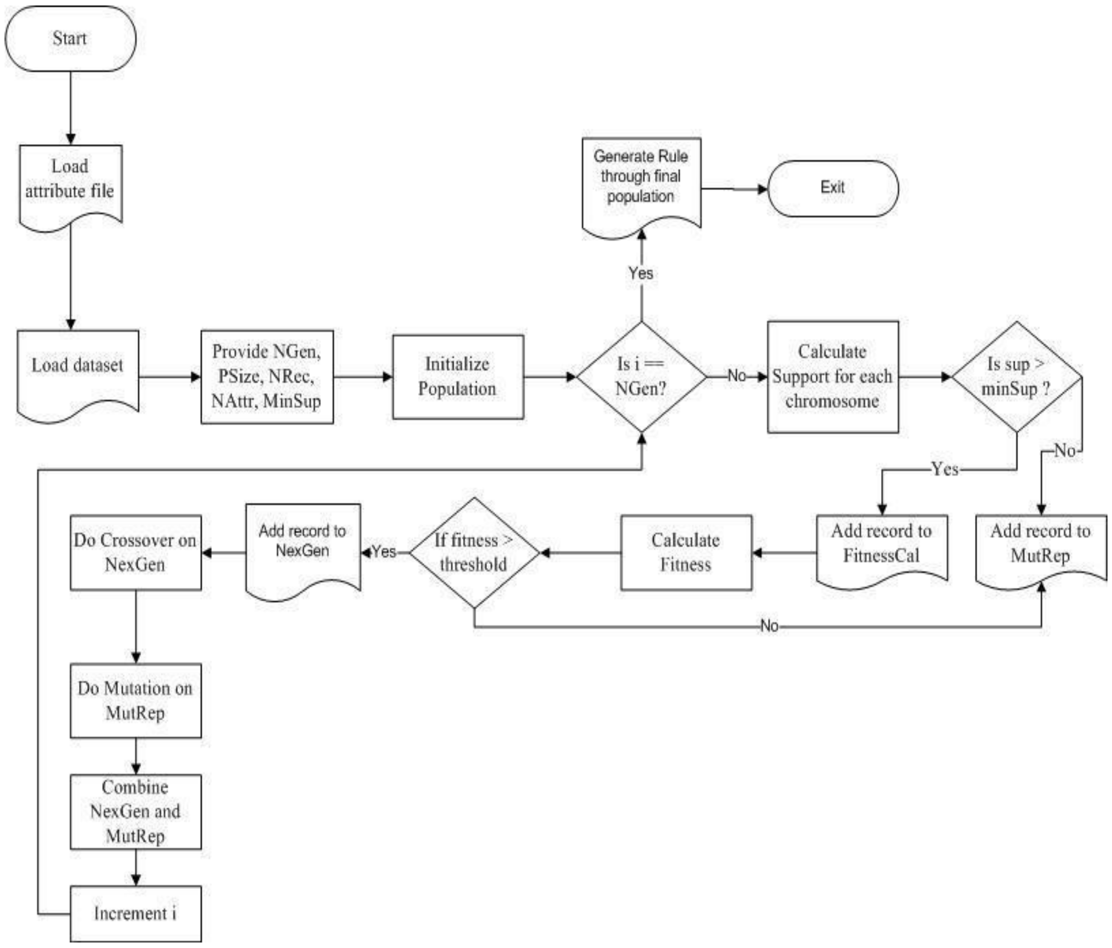

Figure 2: Flowchart for Implementation I

4.1.3. Pseudo Code for Implementation I

Algorithm: Implementation I

Input: **Dataset D**

PSize - Population Size

NGen - Number of generations

RMut - Mutation Rate

RCross - Crossover Rate

Output:

Association Rules

```
1. Begin
2.  for i=1 to PSize  // No. of Chromosome in one generartion
3.      Generate Chromosomes randomly
4.  End for
5.  for each Psize // for each chromosome
6.      Calculate Support
7.      If (Support > MinSupport)
8.      Calculate Fitness
9.  End for
10. for each number of generated rules of Step 5-9
11.     If (Fitness > Threshold) // Threshold : AvgFitness of last generation
12.     Add rule to nexGen
13.     Else
14.     Add rule to MutRep   //MutRep:Rules for mutation
15. End for
16. for  i=1 to NGen // No. of iteration
17.     do Crossover
18.     do Mutation
```

```
19.     Repeat Step 5 – Step 15
20.  End for
21.  Generate rule by final population
22. End
```

4.1.4 Technical details of Implementation I

Chromosome Representation

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

Figure 3: Chromosome Representation in implementation I

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 figure 3.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 that indicates this attribute is in antecedent part, 1 indicates it to be in consequent part, whereas 2 indicates that the attribute is not present in the rule. Chromosome in figure 3 shows a rule of form:

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

Population Generation

Population is the collection of various chromosomes i.e. a set of possible solution. In the starting of the implementation we have to specify possible number of chromosome that we want to take in one generation this number is represented as PSize in our implementation. Initially the population is generated randomly. By providing minimum and maximum value of the attribute a lower bound and an upper bound is generated in that range. Iterations are implemented on this initial population where the range selected is reduced for better refinement of the rules generated.

Fitness Calculation

For every chromosome in the population fitness is calculated. If chromosome have 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. Initially Support, Confidence, Lift, 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.

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

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

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

3. Lift [31]– indicates how frequently A and B occurred when it is known that B was already present in the record.

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

4. Conviction [32] - 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.

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

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 multiobjective maximizing function of form

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

Where constant 4,3,2,2 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 send 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 offsprings. For this implementation we have taken a hybrid of two crossover operators i.e. Single point and multipoint in one generation single point crossover is implemented and in next generation multipoint crossover is applied. 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 two selected at and one point and on certain other iterations we choose multiple point at which swapping of gene overs in crossover operator .Where first operator preserves the quality of fit chromosome whereas multi point crossover will introduce more chromosome 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 be have implemented a selective bit mutation where we select a bit to change in the chromosome and then generates 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. In our implementation we apply mutation operator over chromosomes which are not fit. There we select any of the gene (attribute) of the chromosome and increase or reduces the lower bound and upper bound. 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

Selecting two chromosome for crossover we have used the probability based procedure were the support count is used as a probability parameter. Among the chromosomes of previous population a parent chromosome is selected having value higher than the average support count of entire population.

ALGORTITHM	**TYPE**	**ENCODING**	**CROSSOVER**	**MUTATION**	**FITNESS FUNCTION**
Implementation I	Numeric	Real	Single Point And multi point	Selective bit	Support, Confidence, Lift, Conviction

Table 6: Designing Specification of Implementation I

4.2 IMPLEMENTATION II:

This section explains the second proposed algorithm i.e. implementation 2.The main benefit of this algorithm is that it generates nearly fit chromosome from the initial population itself.. Which improves its final set of solution.

4.2.1. Detailed Explanation of Implementation II

Inputs:

PSize - Population Size

NGen - Number of generations

RMut - Mutation Rate

RCross - Crossover Rate

Step 1: Initialization

i. Take an input dataset which contains number of attributes and records. Provide 'NAtt' and 'NRec' for the dataset.

ii. Generate an initial population having ‘PSize’ number of chromosomes on the basis of minimum and maximum value of attributes with the frequency of a value in the dataset.

iii. Calculate Fitness of the initial population. Using power equation as stated in the implementation I.

Step 2: Introduce Diversity

i. Crossover: Apply multi crossover operator on given population.

ii. Mutation: Apply Selective bit mutation operator on given population.

Step 3: Change Population

i. Calculate the fitness value for new chromosomes introduced by Step 2.

ii. Compare average fitness of last generation to the fitness of current chromosome.

iii. If fitness value of new chromosome is higher than the average fitness of last generation than consider that chromosome as fit therefore keep it in next generation.

Step 4: Iteration

i. **Repeat** Step 2-Step4 ‘NGen’ number of times.

Step 5: Generate Rules

Decode the chromosomes found in last generation as they are the final rules of the dataset.

Figure 4. Illustrates the flow of algorithm proposed in 4.2.1.It can be seen that this algorithm is similar to that of implementation I but the difference comes in the initialization of the population.

4.2.2 FlowChart for Implementation II

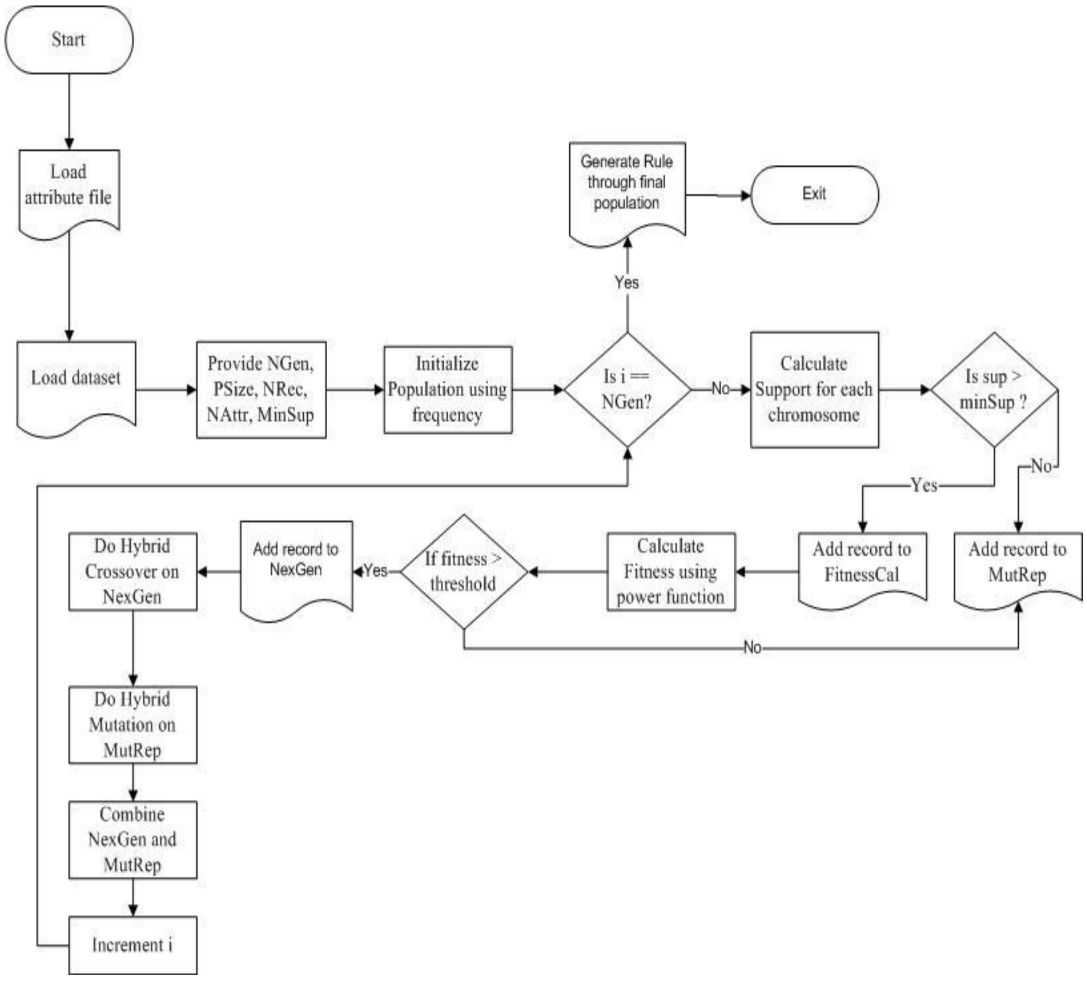
Start
Load attribute file
Load dataset
Provide NGen, PSize, NRec, NAttr, MinSup
Initialize Population using frequency
Is i == NGen?
Yes
Generate Rule through final population
Exit
No
Calculate Support for each chromosome
Is sup > minSup ?
Yes
No
Add record to FitnessCal
Add record to MutRep
Calculate 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

Figure 4: Flowchart for Implementation II

4.2.3 Pseudo Code for Implementation II

Algorithm: Implementation II

Input: Dataset D

PSize - Population Size

NGen - Number of generations

RMut - Mutation Rate

RCross - Crossover Rate

Output: Association Rules

```
1. Begin
2. for i=1 to PSize  // No. of Chromosome in one generartion
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
```

4.2.4. Technical details of Implementation II

Similar chromosome representation, Crossover operator, mutation operator, fitness function and selection procedure is used as in implementation I. Implementation II varies in terms of initial population generation from implementation I .

Population Generation

Initial population generation in implementation II 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 are improved further in other generation. If we keep a check on initial population there is chance of getting better results as the output of the genetic algorithm.

CHAPTER 5

IMPLEMENTATION DETAILS

Details of dataset used

Experimental Setup

CHAPTER 5
IMPLEMENTATION DETAILS

To analyze the proposed algorithms we have implemented both the algorithms in JAVA and tested results on various datasets of keel repository. 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 8GB 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 in this dissertation work we have implemented and tested our algorithm on 4 datasets of keel repository[33]. These datasets are:

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 7: Datasets taken for analysis

5.2 Experimental Setup

To implement the proposed algorithm we have created a java program, which is a generalized program that can take any dataset and can apply implementation I and implementation II on it. The flow of the program written for implementation is shown in Figure 5

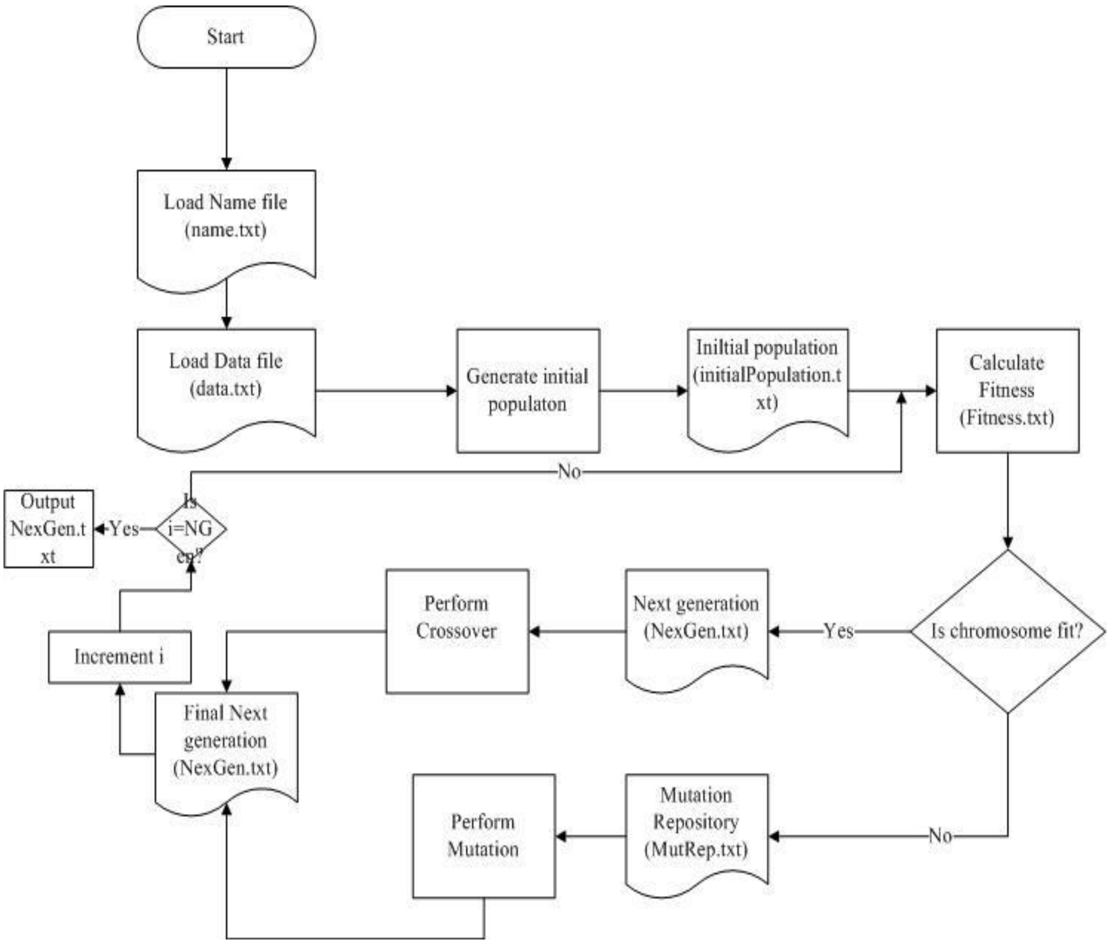
Start
Load Name file (name.txt)
Load Data file (data.txt)
Generate initial populaton
Iniltial population (initialPopulation.txt)
Calculate Fitness (Fitness.txt)
Is chromosome fit?
Yes
Next generation (NexGen.txt)
Perform Crossover
No
Mutation Repository (MutRep.txt)
Perform Mutation
Final Next generation (NexGen.txt)
Increment i
Is i=NGen?
Yes
Output NexGen.txt
No

Figure 5. Flow of implementation's program

Figure 5 illustrates the flow in which program of implementation will run. Initially we have to give dataset and names of its attributes in two separate files i.e. name.txt and data.txt. After that a function is called to generate the initial population once the initial population is generated it will be kept in the initialPopulation.txt file. Now the fitness is calculated on this file. The chromosomes that passes the test of fitness will be stored in NexGen.txt file whereas the chromosomes that fails the test will be kept in Mutation repository i.e. MutRep. After this step crossover is applied to NexGen.txt and Mutation is applied to MutRep.txt. The output of both crossover and mutation are stored as NexGen.txt .Now the same procedure will be followed till iteration reaches to NGen i.e. number of generation specified in the program. The final rules are obtained from the last NexGen.txt. Which can be decode in if then rules by RuleGeneration.java

The result generated are in form of rules and for each rule support, confidence, lift and conviction and fitness is calculated. Analyses of implemented algorithm is done in terms of running time of program, number of rules generated, average fitness at different generation. For comparison purposes these results are plotted in graphs and are illustrated in the next chapter. We have considered four datasets as stated in table 7 for testing purpose. Initially the implementation I and II are tested on quake and stocks dataset. Where we have noted down the number of rules generated, execution time and average fitness at different minimum support count. After this we have compared the results of these two algorithm with themselves and found the result as desired that is implementation II responds better. Later we have compared our results with the existing algorithm. The data of the previous algorithm was taken from [14][16][17].With the similar parameters i.e. population size ,number of generation, min. support etc. on which the results of existing algorithms were given exactly with those same parameters we have executed our algorithm. And whose results are illustrated in form of bar plot in next chapter.

CHAPTER 6
RESULTS AND DISCUSSION

Results of Proposed algorithms

Comparison between two proposed algorithms

Comparison of proposed algorithms with existing

CHAPTER 6
RESULT AND DISCUSSION

The implementation which is 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. All these results are illustrated in graphs in this chapter. To start with, we have first illustrated different results that we get on implementation of algorithms I on Quake and Stocks dataset. Later we have illustrated graph showing results of comparison between implementation I and implementation II for same datasets and at the end we will illustrate the comparison between algorithms proposed by different authors and our proposed algorithm.

6.1 Results of different datasets for implentation I and II

Results of implementation I on Quake dataset

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

Table 8 : Parameter values for implementation on Quake dataset

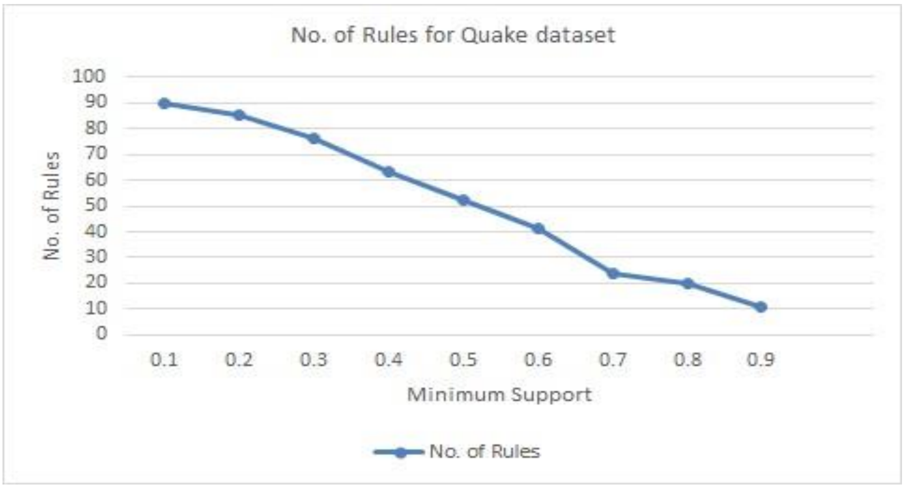

Figure 6. No. of rules generated by Implementation I for Quake dataset

Figure 6 illustrate the number rules that get generated by implementation on different support count. As discussed earlier support count is the first filter of selecting chromosomes for calculating fitness. Minimum support has to be given manual at the starting on the basis of need of application. It can be shown that with increase in minimum support count the number of rules that are generating are decreasing. Which signify that selecting min. support for filter purpose is a correct decision.

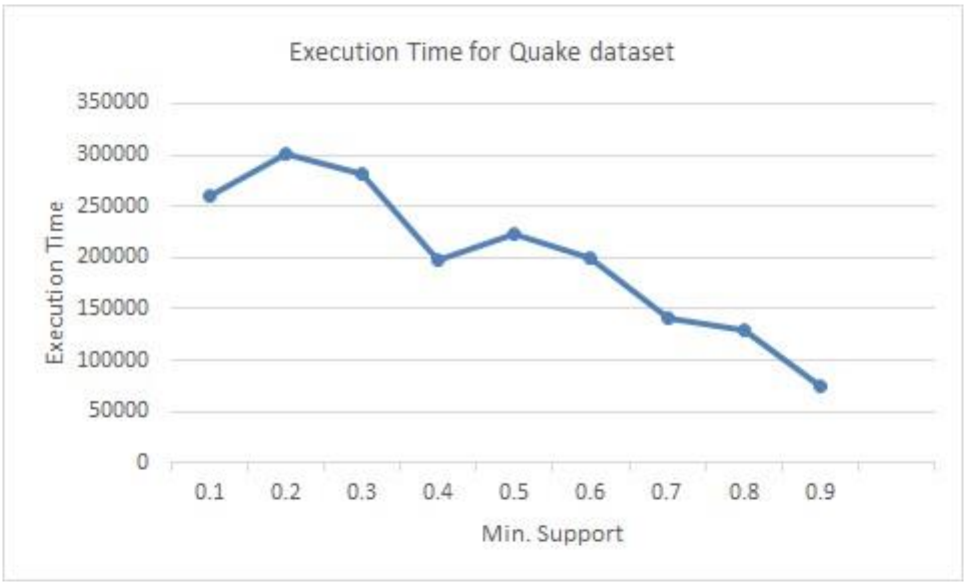

Figure 7. Execution Time taken by Implementation I for Quake dataset

Figure 7 illustrates the execution time taken by implementation I for quake dataset. 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 at such a low min. support value number of rules generated are much higher, therefore for processing them execution time of simulation get 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.

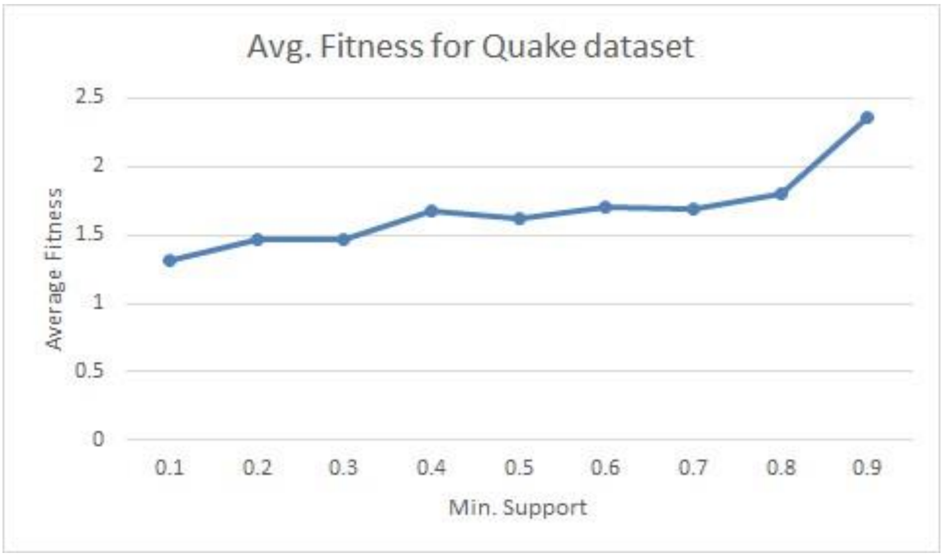

Figure 8. Average Fitness of Rules generated by Implementation I for Quake dataset

Figure 8 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 that is more interesting rules get generated.

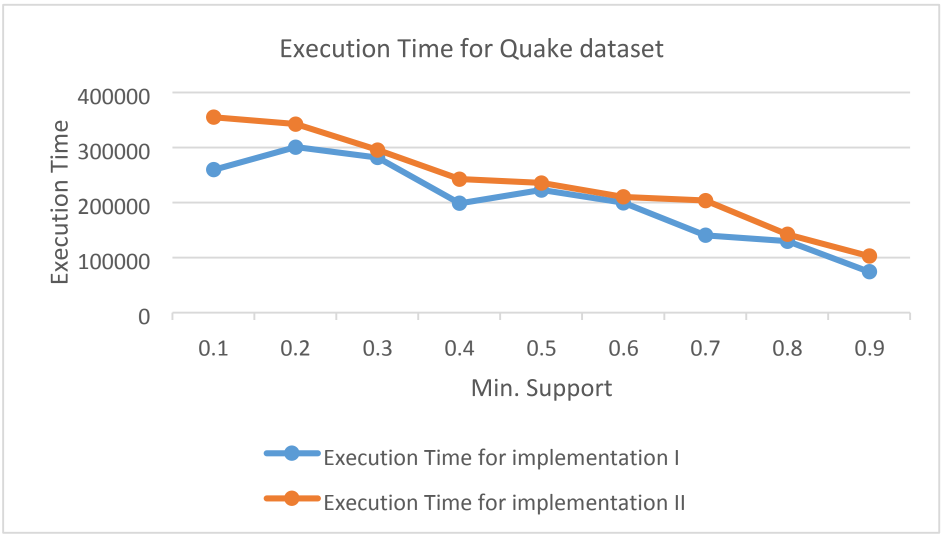

Figure 9. Comparison Between execution time for Implementation I and II for Quake dataset

Figure 9 illustrates the comparison of implementation I and implementation II in terms of execution time. As the implementation II has more processing to do while selecting initial population which increases the overall execution time of the algorithm. By this result it can be a thought that the implementation I is better but it is not so as increase in execution time is also contributing in increase of fitness value of rules that get generated , which is the main motive of the association rule mining.

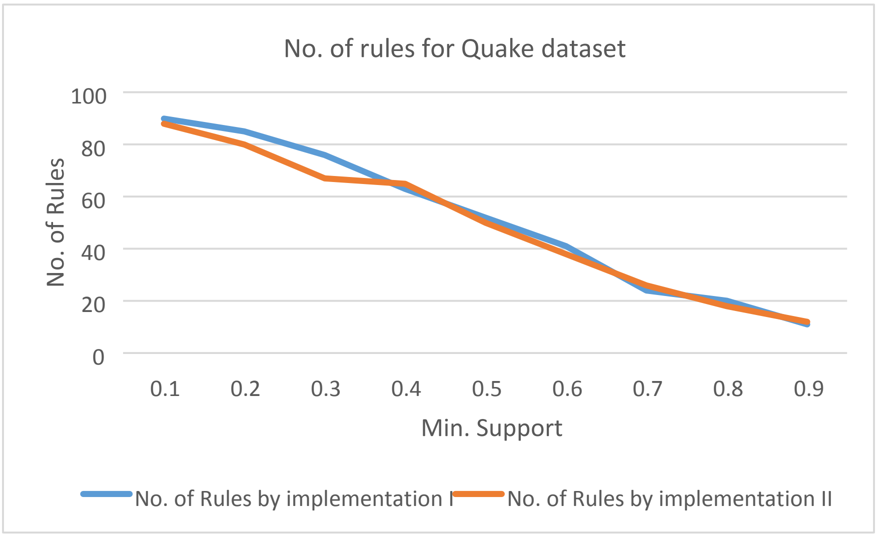

Figure 10. Comparison Between no. of rules generated for Implementation I and II for Quake dataset

Figure 10 illustrates the comparison between implementation I and implementation II in terms of numbers of rules generated at different minimum support count. It can be shown that by increasing the minimum support count the number of rules generated by implementation I and II are nearby same. By this result we can conclude that both implementation I and II are giving nearly same results but again to conclude which implementation is better we have to see the fitness of these rule. Which we will see in the next figure.

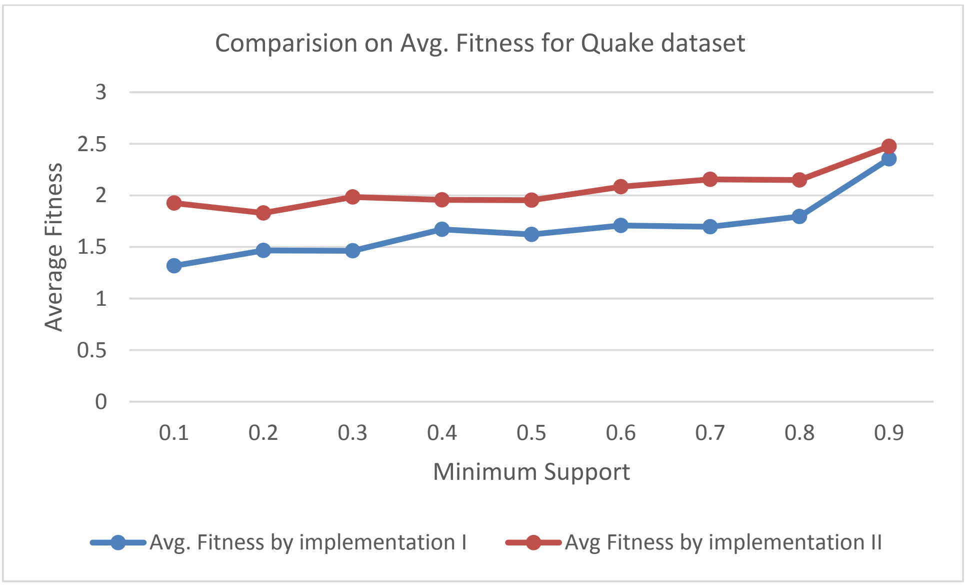

Figure 11. Comparison Between no. on Average fitness generated for Implementation I and II for Quake dataset

Figure 11 illustrate the comparison of implementation I and implementation II on basis of average fitness of rules on quake dataset. It can be concluded that the average fitness increase for implementation II.

For verifying the results of the implementations that we have done, we again generated results of similar manner for other datasets also. A set of results for both implementation I and implementation II for Stocks dataset is illustrated here.

Parameters	Values
Dataset : Stocks dataset	(10/950) (attributes/Records)
Population Size	100
Number of generation	10

Table 9: Parameter specification for implementation on stocks dataset.

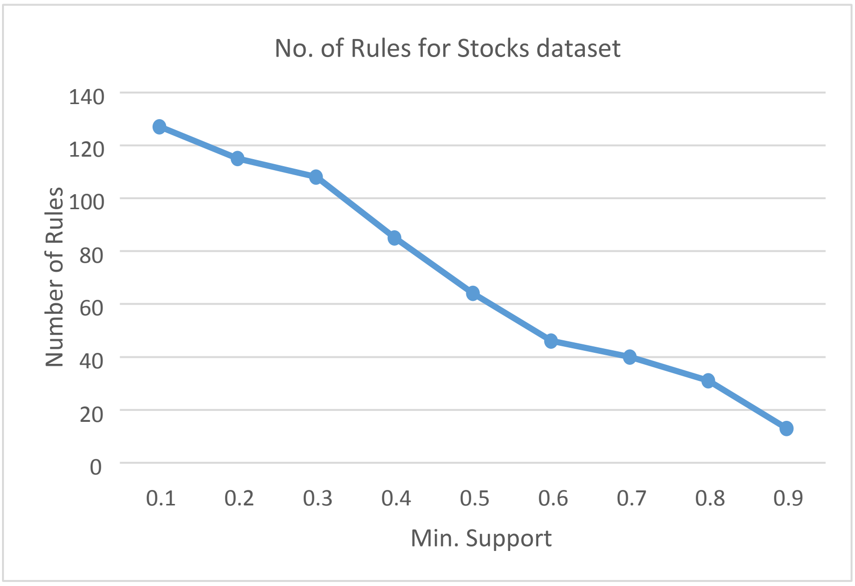

Figure 12. No. of rules generated by Implementation I for Stocks dataset

Figure 12 illustrates the no. of rules generated by implementation I for stocks dataset. This is a similar kind of plot as shown in Figure 6 but with different dataset. This figure results similar to that of Figure 6, i.e. with increase in minimum support count number of rules that get generated is decreasing.

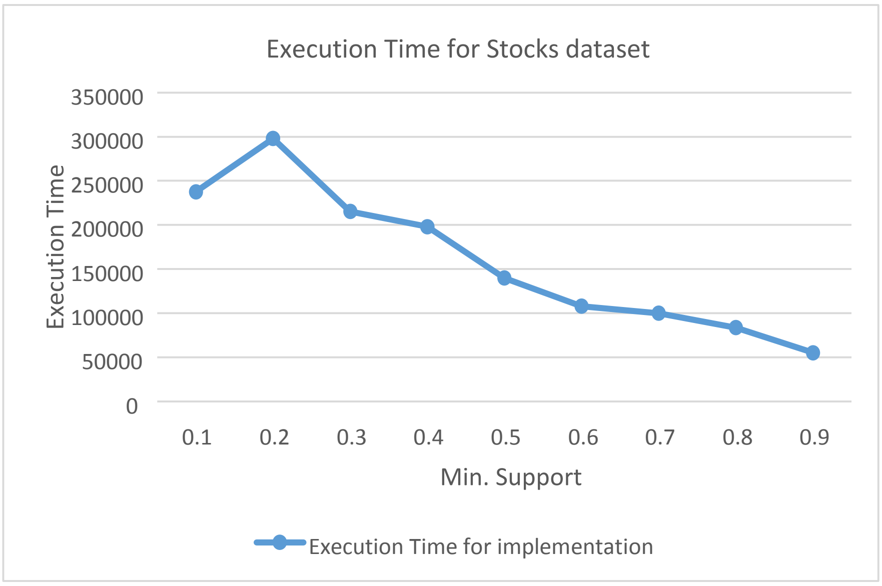

Figure 13. Execution Time taken by Implementation I for Stocks dataset

Figure 13 is illustrating execution time taken by implementation I for Stocks dataset. Similar to that of Figure 7, it also illustrates that after a certain minimum support execution time gets a decreasing order.

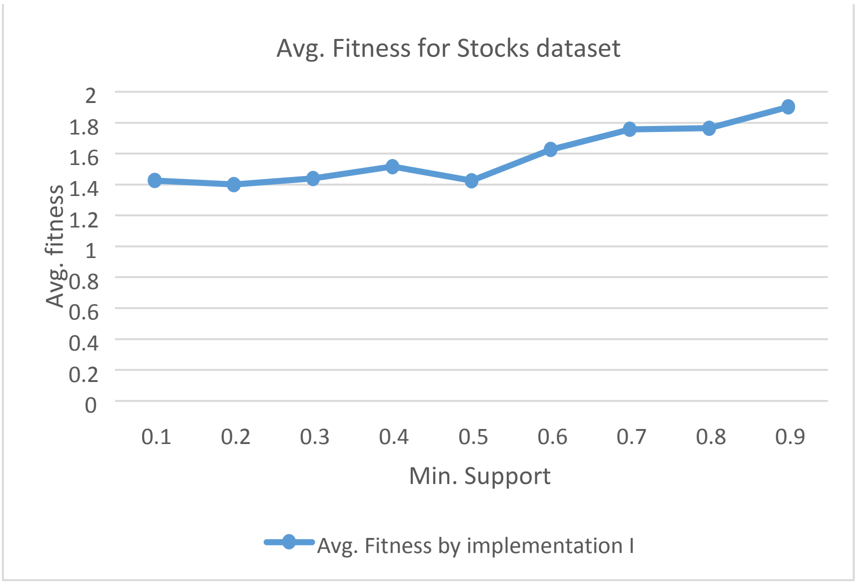

Figure 14. Average Fitness of Rules generated by Implementation I for Stocks dataset

Figure 14 illustrates the average fitness of rules that get generated by implementation I on Stocks dataset. It is showing similar kind of results as of Figure 8 i.e. with increase in minimum support average fitness of rules that are generating is also increasing which means we are getting more relevant rules by increasing minimum support.

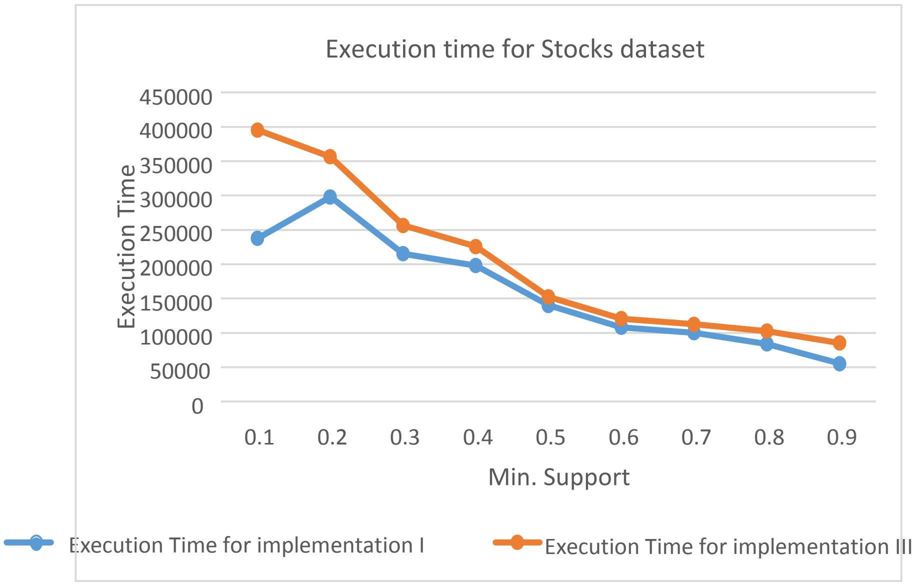

Figure 15. Comparison Between execution time for Implementation I and II for Stocks dataset

Figure 15 is illustrating comparison between execution time for implementation I and II when execution is done on Stocks dataset. This figure is showing same result as of Figure 9. Which means that irrespective of datasets results for implementation I and II will give similar kind of result for Stocks dataset also i.e. implementation II will take more time in execution than implementation I.

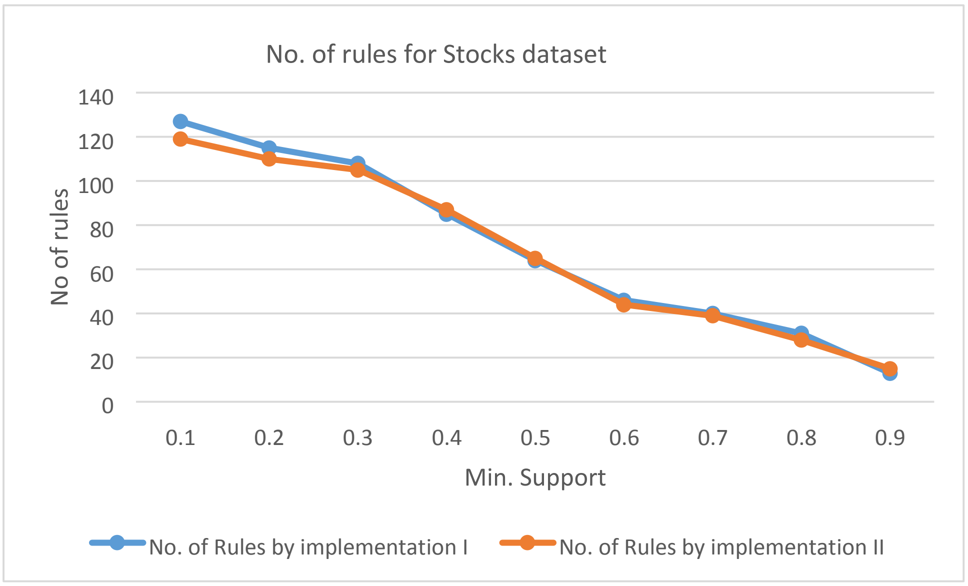

Figure 16. Comparison between number of rules generated by implementation I and II for Stocks dataset

Figure 16. illustrates the similar kind of result as of Figure 10 i.e. with increase in minimum support count number of rules generated by implementation I and II are almost similar for stocks dataset also.

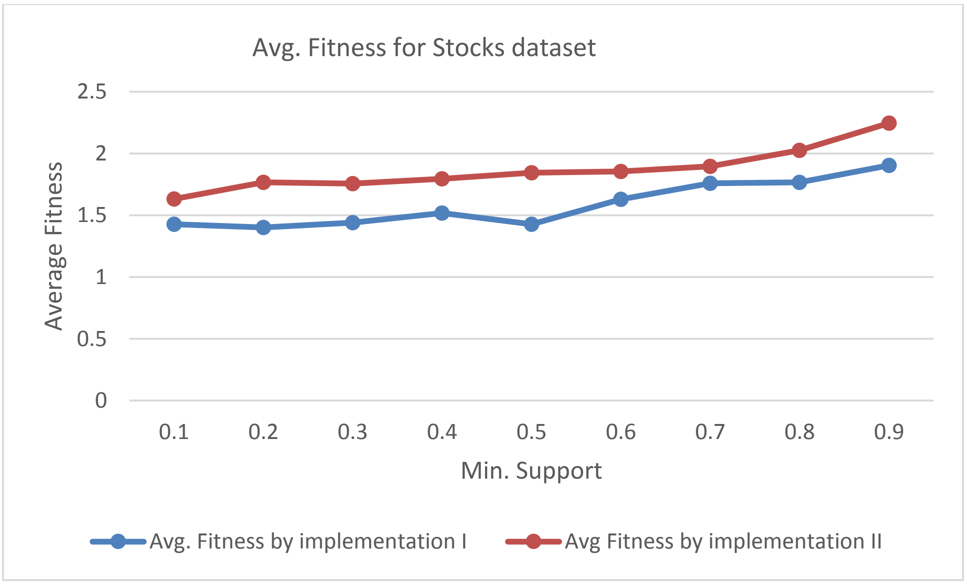

Figure 17. Comparison between average fitness of rules generated by implementations and II for Stocks dataset

Figure 17 illustrates the similar kind of result as of Figure 11. It shows the comparison between average fitness value of rules generated by implementation I and II for Stocks dataset. It can be concluded from both Figure 11 and 16 that the average fitness of rules generated by implementation II is better, i.e. more interesting rules are generated by implementation II.

All the above results are shown for comparing the execution time, number of rules generated by implementation I and II. From which we have concluded that the implementation II takes more time in execution but is better in terms of generating more fit rules i.e. more interesting rule as the average of fitness for rules generated by implementation II is higher than that of implementation I.

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 on each iteration of execution the solution set gets better. To see the effect of this property in our proposed implementation we have plotted the results of different generation which are illustrated here.

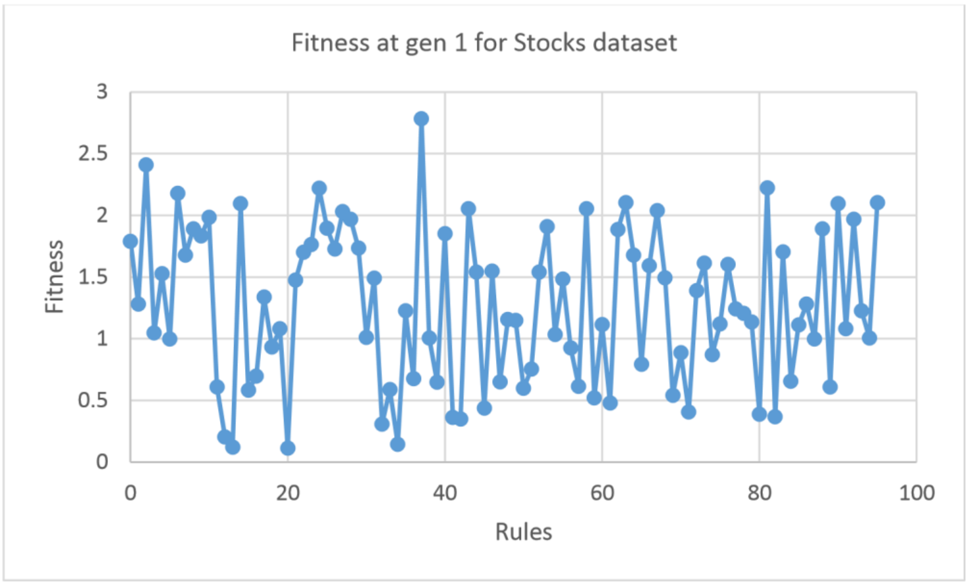

Figure 18. Fitness of rules generated for Stocks dataset by implementation II

Figure 18. Illustrates the fitness of rules generated for Stocks dataset by implementation I at generation 1. That is at the first iteration of execution, whatever chromosomes are generated the fitness of those chromosomes is shown as dots in the above figure. From this plot we can conclude that the chromosomes that are generated by implementation I have fitness more in range of o to 1.5 which is to be improved in next iteration.

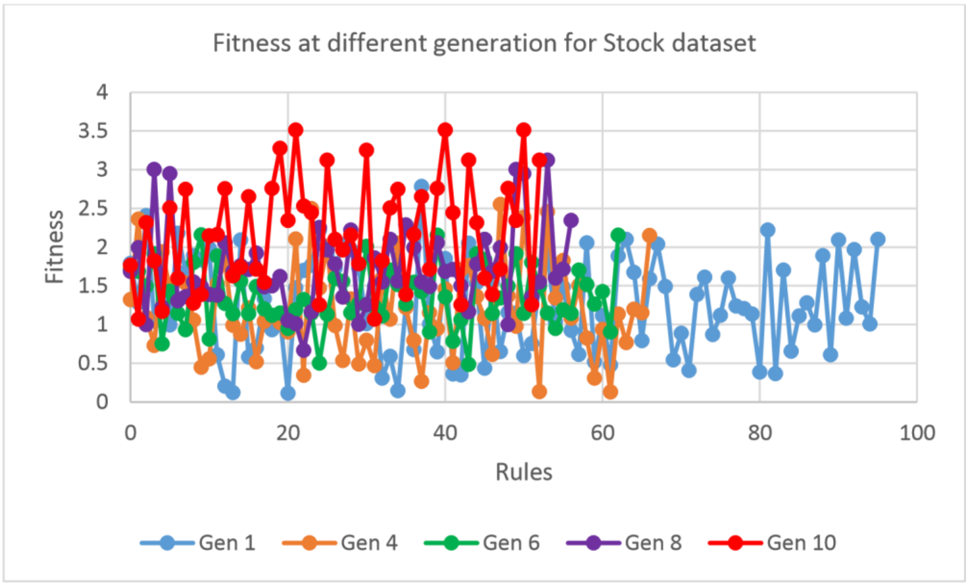

Figure 19. Fitness of rules generated for Stocks dataset at different generation

Figure 19. 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 get better and better with increasing generation, i.e. at each iteration rules are getting more interesting in terms of fitness. Also the number of rules are decreasing at each generation because of filtering of unfit or not- interesting rules at each iteration.

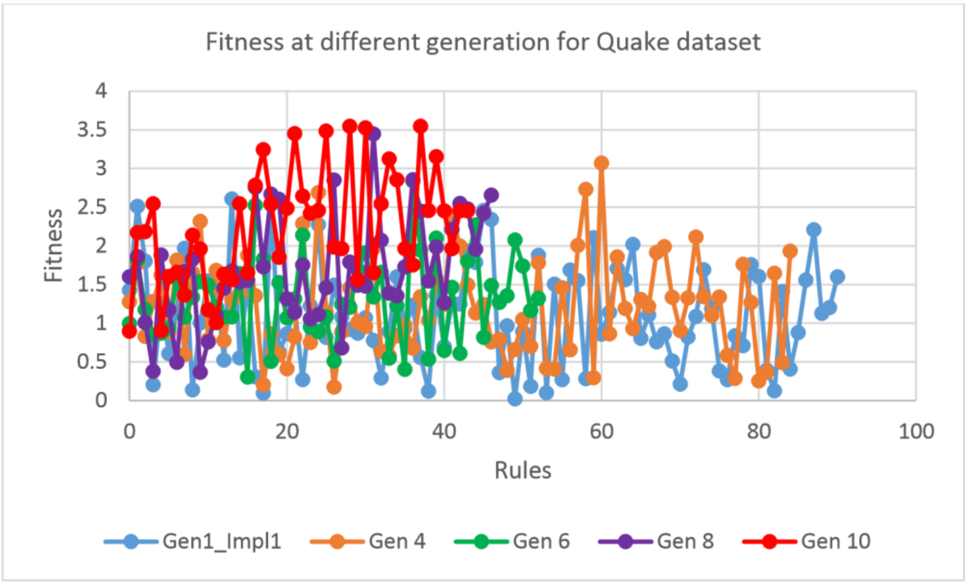

Figure 20 Fitness of rules generated for Quake dataset at different generation

Figure 20. Illustrates the fitness of rules at different generation for quake dataset. It shows the similar kind of results as of Figure 19 i.e. with increase in generation the rules are getting more refined due to increase in the fitness value of rules(Shown by peaks in the graph) and decreasing the number of rules(Shown by tail of the plot for each generation).

With the results of both the above graph we can see that our proposed algorithm is following the basic property of genetic algorithm i.e. the evolving nature of the solution at each iteration.

As it was discussed in the last chapter that the implementation I and II varies in terms of initial population generation i.e. implementation I generates the chromosome for initial population randomly by keeping the track of minimum and maximum value of each attribute. Whereas implementation II generated the chromosomes for initial population by keeping track of minimum and maximum value of each attribute and the frequency of values occurring in the dataset. To see the effect of this change for initial population generation we have plotted a graph for fitness of initial chromosomes i.e. the rules of generation 1.

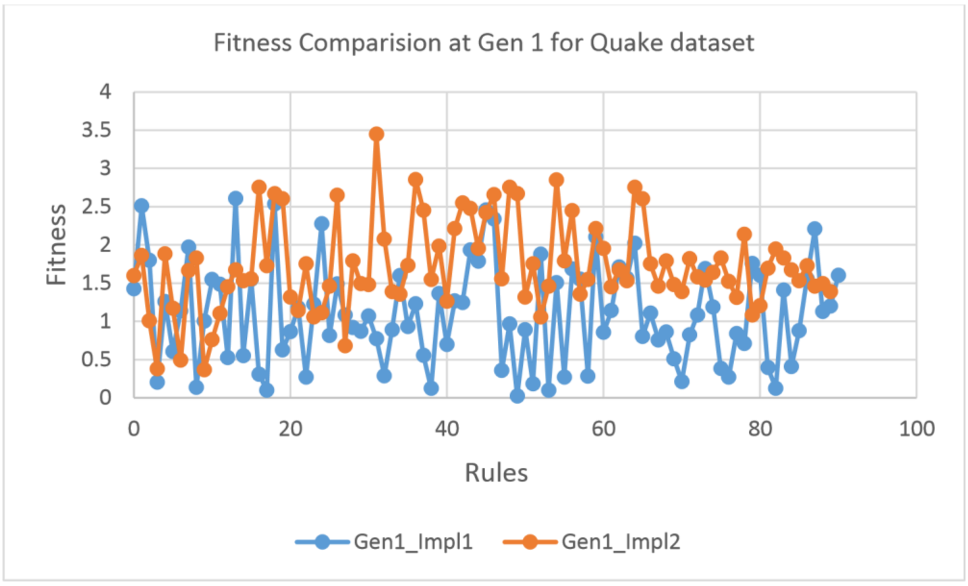

Figure 21. Fitness of rules of initial population for implementation I and II on quake dataset

Figure 21 illustrates the fitness of initial population generated by implementation I and implementation II on quake dataset. From this figure we can conclude that the rules generated as initial population from implementation II has higher fitness value than compared to that of implementation I .Which was the main goal of the changes that were introduced in implementation II. So from this conclusion we can say that implementation II takes more time in execution but generates more interesting rules.

6.2 Comparison of proposed approach with existeing approach

Next, to compare our proposed algorithm implementation I and implementation II with previously known algorithms we have plotted the comparison bar graphs. Where we have compared our results with four previously known algorithms that are EARMGA, GENAR, MODENAR and MOPNAR. Theory regarding these algorithms have been discussed in the literature survey chapter. 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 that are Quake, Pollution, Stocks and Stulong. The values of different parameters i.e. Number of rules, average support and average confidence for previously known algorithm have been taken from the literature [17] and that is compared with the results generated by implementing proposed algorithms.

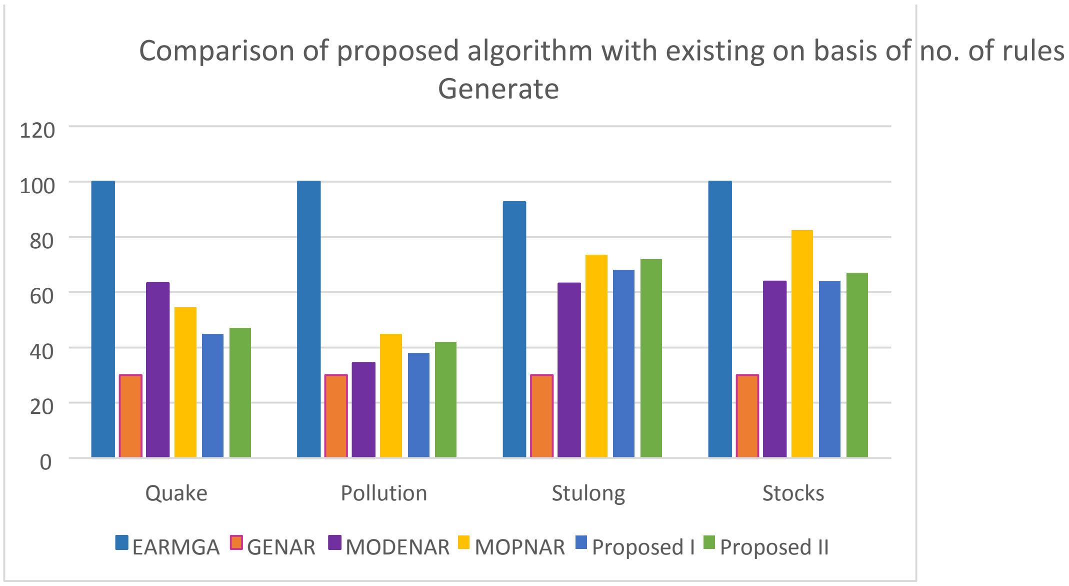

Figure 22. Comparison of proposed algorithm with existing on basis of no. of rules generated

	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 I	45	38	68	64
Proposed II	47	42	72	67

Table 10 : Number of rules generated by different algorithms

Figure 22 is the graphical plot of data presented in table 10 collected from [14][16][17]. 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 these algorithms generated with different datasets. From this plot we can conclude that implementation I and implementation II have generated less number of rules as compared to EARMGA, MODENAR and MOPNAR.

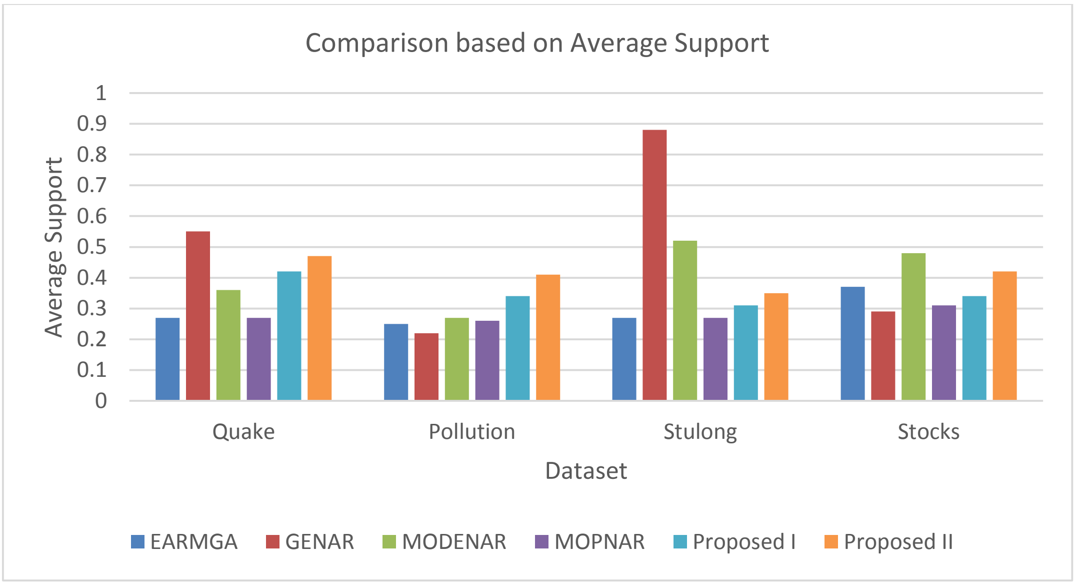

Figure 23. Comparison of proposed algorithm with existing on basis of Average Support of rules generated

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 I	0.42	0.34	0.31	0.34
Proposed II	0.47	0.41	0.35	0.42

Table 11: Average Support of different algorithms

Figure 23 is the graphical plot of data presented in table 11 collected from [14][16][17]. This bar plot illustrate 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 shown it can be concluded that implementation I and implementation II results in better average support count than EARMGA, MODENAR, MOPNAR. Which signify that rules generated by implementation I and implementation II are more interesting than the rules generated by previous algorithms.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

Conclusion of dissertation

Future research direction of the work

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In the following dissertation work we have studied and analyzed the problem of association rule mining for numeric datasets. As processing numeric dataset for getting information is a tedious and computationally complex task because of the nature of the data we have to deal with. Main challenges that we come across while starting with the process of association rule mining for numeric dataset was the huge number of expected rule that will be generated by numeric dataset. Therefore, we needed a technique which run 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 so that more interesting rules can be generated. The pre-existing technique that follow these criteria is genetic algorithm. Which has the property of running for solution in parallel and improving their own solution in each 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 dissertation work we have proposed two approaches for mining numeric association rule 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 rule, as it analyze multiple rules together at a time from a population which decreases the probability of getting stuck in a local optimal solution which might occur while using any other traditional technique which works on analyzing single rule at a time. Further, in each iteration 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 each iterations rules get improved. Rather than using traditional methods for optimization i.e. derivative function, fitness functions are used for optimization which makes the optimization task more efficient.

7.2 FUTURE SCOPE

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 problem 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 [34][35].We would like to test our proposed work on similar kind of real world problem.
3. While working on the implementations we realized that if this algorithm is implemented in a parallel environment it can give much better results that with less execution time. Hence in future we would like to implement this algorithm in a parallel environment.
4. Moreover in the current work we have only concentered on IF-THEN rules, In future we would like to implement genetic algorithm for mining IF –THEN, ELSE rules.
5. Also one of the issue 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 algorithm can also be the future research direction of this work.

All these are future research directions that we want to follow for further work on same subject.

REFERENCES

[1] R. Agrawal, T. Imielinski and A. N. Swami, "Mining Association Rules Between Set of Items in Large Databases," in *ACM SIGMOD International Conference on Management of Data*, Washington D.C.,pp. 207-216,May 1993.

[2] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison- Wesley Publishing Co., Inc., 1989.

[3] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *Proc. of the Twentieth International Conference on Very Large Databases*, Santiago de Chile, Chile, pp. 487-499, January 1994.

[4] C. Borgelt, "Efficient implementations of Apriori and Eclat," in *Proc. Workshop Freq. Itemset Mining Implement*, Melbourne, FL, vol. 90, pp. 280-296, November 2003.

[5] M. Houtsma and A. Swami, "Set-Oriented Mining for Association Rules in Relational Databases," in *Proc. of the 11th IEEE International Conference on Data Engineering*, Taipei, Taiwan, pp. 25-33, March 1995.

[6] C. Hidber, "Online Association Rule Mining," in *Proc. ACM SIGMOD International Conference on Management of Data*, Philadelphia, PA, USA, pp. 145-156, June 1999.

[7] A. Ghosh and B. Nath, "Multi-objective rule mining using genetic algorithms,"in *Journal of Information Science,Elsevier*, vol. 163, pp. 123-133, June 2004.

[8] B. Alatas, E. Akin and A. Karci, "MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules," *Journal of Applied Soft Computing, Elsevier,* vol. 8, no. 1,pp. 646–656, January 2008.

[9] X. Yan, C. Zhang and S. Zhang, "Genetic algorithm-based stratergy for identifying association rules without specifying actual minimum support," *Journal of Expert System Application, Elsevier,* vol. 36, no. 2, pp. 3066–3076, January 2009.

[10] R. Anand, A. Vaid and P. K. Singh, "Association rule mining using multiobjective evolutionary algorithms: Strengths and challenges," in *Proc.of Nature &*

Biologically Inspired Computing, Coimbatore, India , pp. 385-390, December 2009.

[11] H. R. Qodmanan, M. Nasiri and B. Minaei-Bidgoli, "Multiobjective association rule mining with genetic algorithm without specifying minimum support and minimum confidence," *Journal of Expert System Application, Elsevier,* vol. 38, no. 1, pp. 288–298, January 2011.

[12] D. Martin, A. Rosete, J. Alcala-Fdez and F. Herrera, "A multiobjective evolutionary algorithm for mining quantitative association rules," in *Proc. of 11th International Conference on ntelligent Systems Design and Applications*, Cordoba, pp. 1397–1402, November 2011.

[13] J.Vashishtha, D. Kumar, S. Ratnoo and K. Kundu, "Mining comprehensible and interesting rules : A genetic algorithm approach," *International Journal of Computer Applications,* vol. 31, no. 1, pp. 39-47, Oct 2011.

[14] B.Minaei-Bidgoli, R.Barmaki and M. Nasir, "Mining numerical association rules via multiobjective genetic algorithms," *Journal of Informantion Science, Elsevier,* vol. 233, pp. 15-24, June 2013.

[15] A. Didel and Poonam, "Association rule mining using genetic algorithm," *International Journal of Science Technology & Management* , vol. 1, no. 5, pp. 61-67, May 2013.

[16] B. Al-Maqaleh, "Discovering Interesting association rules: A multi-objective Genetic algorithm approach," *International Journal of Application in Information Science,* vol. 5, no. 3, pp. 47-52, February 2013.

[17] D. Martin, A. Rosete and J. Alcala, "A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules," *IEEE Transaction on evolutionary computation,* vol. 18, no. 1, pp. 54-69, February 2014.

[18] N. S. Rai, S. Jain and A. Jain, "Mining interesting positive and negative association rule based on improved genetic algorithm," *International Journal of Advanced Computer Science Applications,* vol. 5, no. 1, pp. 160-165, May 2014.

[19] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay and C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: part II," *IEEE Transaction on evolutionary computation,* vol. 18, no.1,pp. 20-35, February 2014.

[20] M. Vannucci and V. Colla, "Meaningful Discretization of Continuous Features for Association Rules Mining by Means of a Som," in *Proc. of the European Symposium on Artificial Neural Networks*,Belgium, pp. 489-494, April 2004.

[21] C. Tsai, C. Lee and W. Yang, "A discretization algorithm based on class-attribute contingency coefficient," *Journal of Information Science,Elsevier* , vol. 178, no. 3, pp. 714-731,February 2008.

[22] D. Janssens, T. Brijs, K. Vanhoof and G. Wets, "Evaluating the performance of cost-based discretization versus entropy and error-based discretization," *Journal of Computers and Operations Research*, vol. 33, no. 1, pp. 3107-3123, November 2006.

[23] P. Tan, V. Kumar and J. Srivastava, "Selecting the right interestingness measure for association patterns," in *Proc. 8th International Conference of KDD*, Edmonton, AB, Canada,pp. 32-41, July 2002.

[24] F. Berzal, I. Blanco, D. Sanchez and M. Vila, "Measuring the accuracy and interest of association rules: A new framework," *Journal of Intelligent Data Analysis*, vol. 6, no. 3, pp. 221-235, 2002.

[25] J. Alcala-Fdez, N. Flugy-Pape, A. Bonarini and F. Herrera, "Analysis of the effectiveness of the genetic algorithms based on extraction of association rules," *Journal of Fundamenta Informaticae - Intelligent Data Analysis in Granular Computing*, vol. 98, no. 1, pp. 1-14, January 2010.

[26] M. Obitko, " Genetic Algorithm Tutorial," [Online]. Available: https://courses.cs.washington.edu/courses/cse473/ 06sp/GeneticAlgDemo/index.html.

[27] jaime, "The genetic algorithm notebook," [Online]. Available: http://www.geneticprogramming.com/.

[28] C. Tew, C. Giraud-Carrier and K. Tanner, "Behaviour-based clustering and analysis of interestingness measures for association rule mining," *Springer Data mining knowledge Journal*, vol. 28, no. 4, pp. 1004-1045, June 2013.

[29] K. Venugopal, K. Srivivasa and L. Patnaik, "Dynamic Association Rule mining Using Genetic Algorithms," in *Soft Computing for Data Mining Applications*, Springer Berlin Heidelberg, vol. 190, pp. 63-80, 2009.

[30] A. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," in *Advances in evolutionary computing*, New York, NY, USA, Springer-Verlag, pp. 819-845,2003 .

[31] S. Brin, R. Motwani, J. D. Ullman and S. Tsu, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," in *Proc. of ACM SIGMOD international conference on Management of data*, New York ,NY ,USA, pp. 255-264, June 1997.

[32] J. Magalhaes and Cidem "A Comparative Study of Crossover Operators for Genetic Algorithms to," *wseas transactions on computers*, vol. 12, no. 4, pp. 164-173, April 2013.

[33] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Value Logic and Soft Computing, vol. 17, no. 2-3, pp. 255-287, 2011.

[34] N. H. Moin, O. C. Sin and a. M. Omar, "Hybrid Genetic Algorithm with Multiparents Crossover for Job Shop Scheduling Problems," *Mathematical Problems in Engineering, Hindawi*, vol. 2015, pp. 1-12,2015.

[35] S. Barak and M. Modarres, "Developing an approach to evaluate stocks by forecasting effective," *Expert Systems with Applications, Elsevier,* vol. 42, pp. 1325-1339, 2015.

Appendix A

Various interestingness parameters that can be used for association rule mining are:

- Support : Count(A ∪ B)/N
- Confidence : Support(A ∪ B)/Support(A)
- Comprehensibility: Log(1+|B|)/log(1+|A ∪ B|)
- Interestingness: confidence(A) * Confidence(B) * (1-Support(A ∪ B)/No. of tuples in database)
- Amplitude of interval: (Max(A) – Min(A))/tradeoff constant
- Relative Confidence : Support(A ∪ B)- Support(A) * Support(B) /Support(A)(1-Support(Y))
- Performance : Support * Confidence
- Lift : Confidence(A-> B) /SUP(B)
- Precision: Count(A∩ B)/|A|
- Coverage: Count(A∩ B)/|B|
- Generality: Count(A ∩B)/|No. of tuples|
- Leverage : P(B|A) – P(A)*P(B)
- OddsRatio: $\frac{P(AB)*P(\backsim AB)}{P(A(\backsim B))*P(\backsim AB)}$
- Conviction = P(A)P(~B)/P(A~B)