CHAPTER 4

GENETIC ALGORITHM FOR MULTI-RELATIONAL DATA

4.1 INTRODUCTION

This chapter describes the adaptation of GA to optimize the association rule mining. The association rule is defined as relation between two data points in the dataset. Since, the optimization algorithm helps to mining algorithm to mine the most significant rules. Here, the optimization technique such as Genetic algorithm is discussed to mine the most significant rules. The GA is one of the best optimization techniques, which is suitable for the above problem; the reason behind this is the GA has genetic operators such as crossover and mutation. Initially, every rule is evaluated based on the fitness function, which is related to the multi-relational datamining algorithm. Then GA has sorted the evaluated rules based on the fitness values, subsequently it selects the rules, which are greater than the fitness rate. From the set of selected rules, GA selects the first two rules, which are subjected to crossover genetic operator.

The result of the crossover produces two offspring’s subsequently this pair of offspring’s are subjected into mutation process. The result of the mutation process also produces pair of mutated offspring. The resultant rules from the crossover and mutation process such as pair of offspring’s and pair of mutated offspring’s are evaluated based on the fitness function to select the rules for crossover and mutation process. This process is terminated when the
number of iteration which is given by the user get satisfied. Finally, the rules are selected which has the fitness value greater than the minimum fitness rate. These selected rules are called as optimized rules. At last, the experimentation is carried out to evaluate the proposed algorithm in terms of number of optimized rules mined by the proposed algorithm and required running time and memory usage also evaluated by varying the values of fitness value, number of iterations, support values.

4.2 GENETIC ALGORITHM

A tough non-linear search technique that is predominantly applicable to troubles involving large numbers of variables is provided by GAs. The technique of producing a recombination operator like crossover and mutation is the power of genetic algorithms to the population of individuals. GAs is not an easy random search, in spite of their randomized nature. To produce new solutions with enhanced satisfaction, GAs takes advantage of the old knowledge held in a parent population, thus, at each generation the population undergoes simulated evolution. Relatively good solutions recreated; and relatively bad ones are die out and that are changed by fitter offspring. It is an important characteristic of genetic algorithms is the fact that they are very effective when searching or optimizing spaces those are not smooth or continuous. Using calculus based methods, these are very difficult or impossible to search e.g. hill climbing.

a. Initialization

Several independent solutions are randomly generated, and to form a primary population. On the nature of the problem, the population size depends, but in general, it contains several hundreds or thousands of possible results. Allowing the complete range of possible results, usually the
population is generated accidentally. In areas where best results are likely to be found, occasionally, the results may be "seeded".

b. Selection

A proportion of the existing population is chosen, during each consecutive generation to rise a new generation. Fitter solutions are typically more probable to be chosen, where independent solutions are selected during a fitness-based process. By evaluating the fitness of each solution, certain selection methods preferentially choose the best solutions. As the latter process may be very time-consuming, other methods rate just a random sample of the population.

c. Reproduction

Through genetic operators: crossover (also called recombination), and/or mutation, the next step is to compose a second-generation population of solutions from those chosen. For breeding from the group selected previously for each new solution to be produced, a pair of "parent" solutions is selected. A new solution is prepared, by means of the above methods of crossover and mutation, which typically shares many of the characteristics of its "parents". Even though Crossover and Mutation are known as the chief genetic operators, it is likely to use other operators such as regrouping, colonization-extinction, or migration in genetic algorithms.

d. Termination

This generational process is repeated, until a concluding condition has been reached. Common concluding circumstances are

1) A solution is found that satisfies minimum criteria

2) Fixed number of generations achieved
3) Allocated budget achieved

4) The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better conclusion

5) Manual inspection and

6) Combinations of the above.

4.3 THE PROPOSED GA OPTIMIZED ASSOCIATION RULE MINING ALGORITHM FOR MULTI-RELATIONAL DATA

The association rule mining is one of widely used area in the field of data mining. The association rule-mining algorithm is applied on the large database, the result of the ARM algorithm produces lot of association rules based on the items in the database. Each of the association rule represents the relationship between the items in the rule. The effectiveness of the relationship between the item sets is defined based on two measures. The first one is support value of a rule and the second one is confidence value of a rule. If the database becomes very large then the mining algorithm produces several number of rules. In this chapter, the association rule-mining algorithm is applied on the multi-relational database. The result of the above process produces the several number of multi-relational association rules. This several number of rules are need to optimize to a minimum count. In this chapter, the genetic algorithm adapted to optimize the multi-relational association rules.

4.3.1 Mining the Multi-Relational Data

The initial process of the proposed approach is the mining the multi-relational data. The proposed approach adapts the apriori algorithm for association rule mining. Generally, the multi-relational association rules
introduce the notion of a frequent query and query subsumption. Instead of
the frequent item sets of the single relation case, in the multi-relational case
there are frequent queries. The support of a query \( l \) is simply the number of
tuples it returns: \( |\{x|l\}| \). The support of the query needs to be compared to
something. For this purpose, a focus \( T \) query is chosen. The relative support of
a query is then the number of tuples returned by the query divided by the
number of tuples returned by the focus query. A frequent query is one that has
higher support than a user supplied constant \( \alpha \). For two queries \( l \) and \( l' \), \( l \)
subsumes \( l' \)

\[
\text{If } \left| \{x|l'\} \right| \subseteq \left| \{x|l\} \right|
\]

Subsumption is a semantic notion, it holds for all possible database
instances, so the tuples returned by \( l' \) are always a subset of the tuples
returned by \( l \). The query \( l' \) is also said to be a specialization or refinement
of \( l \).

**Definition1.** A multi-relational association rule \( l \Rightarrow l' \) exists, where \( l \) and \( l' \)
are two queries and \( l \) subsumes \( l' \). The support of the rule is given by \( \left| \{x|l'\} \right| \)
and confidence is given by \( \frac{\left| \{x|l'\} \right|}{\left| \{x|l\} \right|} \). As in the single relation case, a significant
rule is one with higher support and confidence than the user supplied
constants \( \alpha \) and \( \beta \).

### 4.3.2 Optimizing Rules using Genetic Algorithm

Searching and optimizing the trouble in the problem space, is the
general behavior of a genetic algorithm. A prominent site in the area of
optimization algorithms is achieved by GA. The best features of the genetic
algorithm in proposed approach, for the optimization of the rules are
constructed from the multi–relational association mining. To optimize the rules constructed by the algorithm there has to be a resolution, since the rules constructed from the multi-relational mining algorithm are numerous. The GA, will suit for solving the current problem, with its distinct features. As discussed in the above section, the GA performs the following operations, 1) Initialization, 2) Selection, 3) Crossover, 4) Mutation and 5) Termination. Here, the set of rules are considered as the population, in the proposed approach. To decide the best rules from the problem space using the Genetic algorithm is the aim of proposed approach.

4.3.2.1 Initialization

The genetic algorithm begins its processing by starting the initialization process that helps to prepare the input data for the further processing. The result of the initialization process is the population, which consist set of association rules since the initialization process first finds the association rules from the multi-relational database. These set of association rules are derived by the multi-relational association rule-mining algorithm, which is applied on the multi-relational database. The multi relational database contains a set of tables $MRDB=\{T_i\}$ where $1 \leq i \leq n$ each of which interrelated with some characteristics such as primary key, foreign key and so on.

Consider the multi-relational database $MRDB$, which consists multiple, tables such as $T_1, T_2$ and $T_3$ these tables are connected through primary key or foreign key. The tables and its corresponding data are represented in the Equation (4.1).
From the above Equation (4.1), every data is taken from the each table such as $T_1$, $T_2$, and $T_3$ which belongs to multi-relational database $d_1^1, d_2^2, d_3^3$. It can be said as $d_1^1, d_2^2, d_3^3 \in MRDB$. The multi-relational association rule-mining algorithm is applied on the above tables to mine the multi-relational association rules. The following Equation (4.2) represents the set of multi-relational association rules mined from $MRDB$.

$$AR_{MR} = \{ (d_1^1, d_3^3 \rightarrow d_2^2), (d_1^1 \rightarrow d_2^2, d_1^1) \} \ldots$$

(4.2)

From the above Equation (4.2), the symbol $AR_{MR}$ is the representation of set of multi-relational association rules. Each of the rules contains many items, which may be derived from the single table or multiple tables. Once the $AR_{MR}$ mined from $MRDB$, the genetic algorithm utilizes the multi-relational association rules as input data to construct the initial population. The GA slightly modifies the original format of the multi-relational association rule to construct the initial population as appropriate format. The initial population of the genetic algorithm consist of set of patterns, which is derived from the multi-relational association rules.

$$P_{MR} = \{ (d_1^1, d_3^3, d_2^2), (d_1^1, d_2^2, d_1^1) \} \ldots$$

(4.3)

The above Equation (4.3) represents set of patterns used for the initialization process where the value of $P_{MR}$ represents the set of multi-relational patterns from which each of the pattern is represented as the initialization population of the GA $P_{GA} = \{ p_i \}$ where $1 \leq i \leq m$ which can be
written as \( P_{GA} = \{ p_1, p_2, \ldots, p_n \} \) where \( p_1 = \{ d_1^1, d_3^3 \rightarrow d_5^2 \} \), \( p_2 = \{ d_3^1 \rightarrow d_5^2, d_1^1 \} \) and so on.

Consider the following Table 4.1 as multi-relational database \( MRDB \) as example. With the intention of finding the multi-relational association rules given to GA as input for the initialization process, the multi-relational association rule-mining algorithm is applied on \( MRDB \). The result of the mining process produces the set of multi-relational association rules \( AR_{MR} \), which is represented in the Equation (4.4) subsequently, which is converted into set of multi-relational patterns \( P_{MR} \) which is represented in the Equation (4.5) then it is converted into population of the initialization process \( P_{GA} \) which is represented in the Equation (4.6).

**Table 4.1 Representation of Multi-Relational Database \( MRDB \)**

<table>
<thead>
<tr>
<th>Multi-relational database</th>
<th>Tables</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table T1</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>Table T2</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>Table T3</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

\[
AR_{MR} = \{ (a \rightarrow c), (d \rightarrow c,a), (b \rightarrow e), (a,b \rightarrow d), (b,c \rightarrow d) \}
\]  
(4.4)

\[
P_{MR} = \{ (a,c), (d,c,a), (b,e), (a,b,d), (b,c,d) \}.
\]  
(4.5)

\[
P_{GA} = \{ p_1, p_2, p_3, p_4, p_5 \}
\]  
(4.6)

**4.3.2.2 Selection**
The ultimate aim is to select the important rule from the result of the multi-relational association rule mining. Once the population process of the genetic algorithm is completed, the next step is the selection of best patterns from the set of patterns $P_{GA}$ of the initialization process. The best patterns are selected based on durability, which is calculated based on the fitness function. The fitness function of each pattern is the principal feature that defines feasibility of each pattern. Based on these criteria, proposed algorithm defines the fitness function of the pattern based on the multi-relational association rules and genetic algorithm. The following Equation (4.7) defines the fitness function of the pattern.

$$f(p_i) = \left[ \text{conf}(r_i) + \log\left(\text{sup}(r_i) \times k_i\left(\frac{k_i(r_i)}{k_i(l_i)}\right) \times \text{freq}(r_i)\right) \right].$$  \hspace{1cm} (4.7)

From the above equation where $f(p_i)$ represent the fitness of $p_i$, $\text{conf}(r_i)$ confidence of the rule $r_i$, $k_r$ represents number of items present in the rule $r_i$ which can be written as $k_r = k_r(r) + k_r(l)$ where the symbols $k_r(r)$ and $k_r(l)$ is representation of number of items presents in the right side of the rule separator and left side of the rule separator respectively. Fitness is calculated for each rule from $P_{GA}$ based on the above fitness function (Equation (4.7)).

The rules that have sufficient fitness get selected for further processing and the rest of the rules get rejected. The rules, which are fit or not fit is determined by the fitness rate of the rule. In this chapter, consider the fitness $f_r$ rate of the rule as 75%. Once the fitness value of the rule is calculated based on the above Equation (4.7), the rules are arranged into the list based on fitness rate of each rule $f_r$. Once the rule is arranged based on the fitness rate, the next step is to discard the rules, which has the fitness rate value below (75%). The remaining selected rules are used for making the
combinations which helps to optimize the multi-relational association rules. The combinations are made by the reproduction function of the genetic algorithm.

Consider the multi-relational database, which is represented, in the above Table 4.1, from which the resultant value of the initialization process is $P_{GA} = \{p_1, p_2, p_3, p_4, p_5\}$.

$$
\begin{align*}
\frac{f(p_1)}{f(p_2)} &= 0.95 + \log \left( 0.75 \times 2 \left( \frac{1}{1} \right) \times 1 \right) \\
\frac{f(p_3)}{f(p_4)} &= 0.95 + 0.1760 \\
\frac{f(p_5)}{f(p_6)} &= 1.126
\end{align*}
$$

The above Equation (4.8) is representation of the example of calculation of fitness function for $p_i (a \rightarrow c)$. Likewise, fitness rate for every rule from $P_{GA}$ is calculated based on the fitness function subsequently, the rules are sorted based on the fitness rate of the individuals, which is represented in the following Table 4.2.

Table 4.2 Fitness Sorted List

<table>
<thead>
<tr>
<th>Rules (P$_{GA}$)</th>
<th>Fitness</th>
<th>$f_r$ (75%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>1.126</td>
<td>Selected</td>
</tr>
<tr>
<td>$P_2$</td>
<td>1.098</td>
<td>Selected</td>
</tr>
<tr>
<td>$P_3$</td>
<td>1.006</td>
<td>Selected</td>
</tr>
<tr>
<td>$P_4$</td>
<td>0.752</td>
<td>Discarded</td>
</tr>
<tr>
<td>$P_5$</td>
<td>0.741</td>
<td>Discarded</td>
</tr>
</tbody>
</table>
From the above Table 4.2, the rules $P_4$ and $P_5$ are rejected since their fitness is less than the fitness rate and the rules $P_1, P_2$ and $P_3$ are selected for the third section, reproduction.

4.3.2.3 Reproduction

Once the selection section removed the non-feasible rules based on the fitness rate, the next thing is to generate the rules through combination of items present in the set of selected rules. The combination process is done through two operators of the genetic algorithm, the first one is Crossover and the second one is Mutation. Once the new combinations are generated based on the above two operators, the new chromosome is evaluated based on fitness. If the fitness of the newly generated chromosome (new rule) satisfied the fitness rate, the rule is taken into account else the combinations made up to the fitness rate becomes unsatisfied.

4.3.2.4 Crossover

The crossover is one of the operator in genetic algorithm which helps to generate the new chromosomes (new rules) through creation of offspring from the selected two parents by switching their features. The generation of new chromosome need to be strong since the crossover operator selects the rules only from the previous section (the rules which are satisfied the fitness rate). In this chapter, the crossover operator has two techniques such as one point crossover and two-point crossover.

Every association rule has two parts, first one is left side of the rule separator“$\rightarrow$” and the second one is right side of the rule separator“$\rightarrow$”. With the intension of generating the new chromosome, the items present in the left side or right side of the rule separator is changed since in this chapter
the single point crossover technique is adapted to generate the new chromosomes. To make the newly generated rules, the crossover technique selects two rules from $P_{GA}$ as parents from which the one point cross over selects the single crossover point initially then the selected crossover points are interchanged between the parents accordingly. The resultant chromosomes from the single crossover point has the characters from both the parents. The following Table 4.3 and 4.4 is the representation of the crossover technique.

**Table 4.3 Selected Parents for Cross Over Technique**

<table>
<thead>
<tr>
<th>Parent 1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 4.4 Parents after the one point crossover processing**

<table>
<thead>
<tr>
<th>Offspring 1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offspring 2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In the proposed approach, the rules are represents by the parents since the chromosomes (parents) have symbols in it. For this purpose, here the rules are represented in terms of data structure. In data structure, the tree encoding is the mostly used technique to represents the data that has the symbols. In the proposed approach the selected parents or chromosomes from $P_{GA}$ are encoded through tree encoding method. The following Figures 4.1,4.2 and 4.3 are the representation of the process of the one point crossover where, the Figure 4.1 represents the selection of parents from the sorted list based on their fitness rate. The Figure 4.2 signifies the selection of crossover point from each parent finally the Figure 4.3 denotes that the newly generated pair of offspring after the one point crossover operation.
Figure 4.1 Representations of the Selected Pair of Parents from the Sorted List

Figure 4.2 Selected Portions in each Parent for One Point Crossover

Figure 4.3 Pair of Offspring (New Chromosomes) after One Point Crossover
The following Figures 4.4, 4.5 and 4.6 is the representation of one point crossover operations for the above multi-relational database, which is represented in the Table 4.1, the Figure 4.4 represents the selection of parents from the sorted list based on their fitness rate is shown in the Table 4.1.

![Figure 4.4 Representations of the Selected Pair of Parents of the Table 4.1](image1)

![Figure 4.5 Representations of the Selected Pair of Parents of the Table 4.1](image2)

![Figure 4.6 Representations of Selected Node in the Pair of Parents of Table 4.1](image3)
The Figure 4.5 signifies the selection of crossover point from each parent finally the Figure 4.6 denotes the newly generated pair of offspring after the one point crossover operation.

4.3.2.5 Mutation

The mutation is another operator in the genetic algorithm, which helps to generate the new rules through creation of offspring after the crossover operation. The mutation process has many techniques, which are listed below

- Bit string mutation
- Flip bit
- Boundary
- Non-uniform
- Uniform
- Gaussian

In this chapter, the “flip bit mutation” is adapted for the mutation process. This technique generates the offspring by changing the single bit of the original offspring. The flip bit mutation is quite simple and fast method. The following Table 4.5 and 4.6 is the representation of the mutation process in which Table 4.5 represents the selection of bit need to change and the Table 4.6 is the representation of pair mutated offspring after the mutation process.

<table>
<thead>
<tr>
<th>Offspring 1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offspring 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 4.6 Offspring after the Mutation Process

<table>
<thead>
<tr>
<th>Mutated offspring 1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutated offspring 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The resultant pair of offspring from the one point crossover is used for the mutation process. The following Figures 4.7, 4.8 and 4.9 are the representation of the process of flip bit mutation, whereas the Figure 4.7 represents the resultant pair of offspring from the one point crossover operation. The Figure 4.8 signifies the selection of bit from each offspring for the mutation process finally the Figure 4.9 denotes that the newly generated pair of mutated offspring after the mutation process. Similarly Figures 4.10, 4.11 and 4.12 represents the pair of offspring from one point cross over and the process for and after the mutation process.

---

**Figure 4.7** Represents the Resultant Pair of Offspring from the One Point Crossover

**Figure 4.8** Represents the Selected Item of the each Offspring for the Mutation Process
Figure 4.9 Representations of the Offspring after the Mutation Process

Figure 4.10 Represents the Pair of Offspring from the one Point Crossover of the Table 4.1

Figure 4.11 Represents the Selected Bit of the Offspring for Mutation Process
4.3.2.6 Termination

Once the new chromosomes are generated based on the crossover and mutation operators, the next step is to evaluate the newly generated chromosomes based on fitness function subsequently the newly generated chromosomes and the existing chromosomes are sorted based on their fitness rate. From this sorted list, first two parents are selected for generating the next generation of chromosomes. This process is continuously repeated until the number of iterations get finished. The user can give the number of iterations as termination criteria since, there are no other possible termination criteria, which will fulfill the requirement of the proposed approach. Finally optimized rules are selected from the sorted list and result, which are greater than the minimum fitness rate.

The pseudo code of the optimized algorithm for mining multi-relation patterns is described as follows:
Figure 4.12.1 Pseudo code of the optimized algorithm

Input: Multi-relational database, \( D \in d1, d2, d3 \)
Output: Optimized rules, \( R_{op} \)
Parameters:
\( n \rightarrow \) Number of iteration
\( n(Rs) \rightarrow \) Number of best rules

Pseudocode:

\[
\text{Begin}
\]
\[
R = \text{Mine}(D, \text{min}_\text{sup})
\]
\[
\text{Ini\_po} = \{R1, R2, \ldots , Rn\} \in \mathcal{R}
\]
\[
\text{for } i = 1 \text{ to } n \text{ do}
\]
\[
\text{Fitness, } f(R(i)) = \text{conf}(r) + \log(\text{support}(r).k(r)/k(l)) \times \text{freq}
\]
\[
\text{endfor}
\]
\[
\text{for } i = 1 \text{ to } n \text{ do}
\]
\[
\text{Select Best set of Rule, } R_S
\]
\[
\text{for } j = 1 \text{ to } n(Rs) \text{ do}
\]
\[
(C1.C2) = \text{Crossover}(R_i,R_j)
\]
\[
M = \text{Mutation}(C1)
\]
\[
\text{Fitness, } f(Rs(i)) = \text{conf}(r) + \log(\text{support}(r).k(r)/k(l)) \times \text{freq}
\]
\[
\text{endfor}
\]
\[
\text{endfor}
\]
\[
\text{Select Optimal set of Rule, } R_{op}
\]
\[
\text{End}
\]
4.4 RESULT AND ANALYSIS

In this section, the work is analyzed with the intension to evaluate the performance of it. The experiment result of the proposed algorithm is evaluated based on the number of optimized rules of the proposed algorithm, running time, memory usage for the various values of fitness and number of iterations for the dataset1, dataset2, dataset3, dataset4.

4.4.1 Experimental Results

The following Table 4.7 is the representation of the example of optimized multi-relational association rule of the dataset 1 and its description.

Table 4.7 Represents the Significant Rules and its Description

<table>
<thead>
<tr>
<th>Rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[v-3131] → [ model-29, type-2, color-4, city-26, time-10]</td>
<td>The person brought the vehicle id v-3131 then he selects the model 29 and he choose type 2 also he selects the color 4 in the city 26 at time 10.</td>
</tr>
<tr>
<td>[v-3273] → [ model-31, color-13, time-10, city-8]</td>
<td>The person brought the vehicle id v-3273 he selects the model 31 and he selects the color 13 in the city 8 at the time 10.</td>
</tr>
<tr>
<td>[v-3803] → [ color-10, type-2, city-26]</td>
<td>The person brought the vehicle id v-3273 he selects the type 2 and he selects the color 10 in the city 26.</td>
</tr>
</tbody>
</table>

4.4.2 Number of Optimized Rules based on Iteration and Fitness Values

The following Figure 4.13 denotes that number of optimized rules achieved by the proposed algorithm on dataset1. The number of rules is
evaluated based on fitness value and number of iterations. Based on the
previous algorithm, there are 60848 number of rules are generated for the
dataset 1 for the minimum support 3. The experimentation of the proposed
algorithm carried out by varying the values of number of iterations and the
minimum fitness value. The proposed algorithm optimize the rules into 216
(maximum) which is achieved for the minimum fitness as 3 at 50 iterations.
From the following Figure 4.13, when the value of number of iteration
increases, the number of number of optimized rules also is increased. The
reason behind this is most number of combinations are generated since more
number of rules are has the fitness rate greater than minimum fitness.

Figure 4.13 Number of Optimized Rule for the Dataset 1

The following Figure 4.14 denotes that number of optimized rules
achieved by the proposed algorithm on dataset 2. The number of rules is
evaluated based on fitness value and number of iterations. Based on the
previous algorithm, there are 18797 number of rules are generated for the
dataset 2 for the minimum support 4. The experimentation of the proposed algorithm carried out by varying the values of number of iterations and the minimum fitness value. The proposed algorithm optimize the rules into 233 (maximum) which is achieved for the minimum fitness as 3 at 10 iterations. From the following Figure 4.14, the thing is when the value of fitness rate increased, the number of rules, which has fitness rate greater than the minimum fitness rate, get reduced.

![Number of Optimized Rule for the Dataset 2](image)

**Figure 4.14 Number of Optimized Rule for the Dataset 2**

The following Figure 4.15 denotes that number of optimized rules achieved by the proposed algorithm on dataset 3. The number of rules is evaluated based on fitness value and number of iterations. Based on the previous algorithm, there are 2047 number of rules are generated for the dataset 3 for the minimum support 2.
The experimentation of the proposed algorithm carried out by varying the values of number of iterations and the minimum fitness value. The proposed algorithm optimize the rules into 386 (maximum) which is achieved for the minimum fitness as 3 at 10 iterations. From the following Figure 4.15, the thing is when the value of fitness rate increased, the number of rules, which has fitness rate greater than the minimum fitness rate is reduced.

The following Figure 4.16 denotes that number of optimized rules achieved by the proposed algorithm on dataset 4. The number of rules is evaluated based on fitness value and number of iterations. Based on the previous algorithm, there are 1029 number of rules are generated for the dataset 4 for the minimum support 5. The experimentation of the proposed algorithm carried out by varying the values of number of iterations and the minimum fitness value. The proposed algorithm optimize the rules into 35 (maximum) which is achieved for the minimum fitness as 3 at 50 iterations. From the following Figure 4.16, the thing is when the value of fitness rate
increased, the number of rules, which has fitness rate greater than the minimum fitness rate get reduced.

![Figure 4.16 Number of Optimized Rule for the Dataset 4](image)

**4.4.3 Evaluation of Running Time**

The following Figure 4.17 is representation of evaluation of running time taken for the proposed approach to optimize multi-relational association rules of the dataset 1. In this section, the running time of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous algorithm, there are 60848 number of rules are generated for the dataset 1 for the minimum support 3. To optimize 60848 number of rules at 10 number of iteration, the proposed algorithm needs 14 seconds and it takes 47 seconds, 77 seconds for 30 iterations and 50 iterations respectively.
Figure 4.17  Evaluation of Running Time Based on Number of Iterations for the Dataset 1

The following Figure 4.18 is representation of evaluation of running time taken for the proposed approach to optimize multi-relational association rules of the dataset 2. In this section, the running time of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous algorithm, there are 18797 number of rules are generated for the dataset 2 for the minimum support 4. To optimize 18797 number of rules at 10 number of iteration, the proposed algorithm needs 17 seconds and it takes 35 seconds, 45 seconds for 30 iterations and 50 iterations respectively.
The following Figure 4.19 is representation of evaluation of running time taken for the proposed approach to optimize multi-relational association rules of the dataset 3. In this section, the running time of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous algorithm, there are 2047 number of rules generated for the dataset 3 for the minimum support 2. To optimize 2047 number of rules at 10 number of iteration, the proposed algorithm needs 9 seconds and it takes 28 seconds, 48 seconds for 30 iterations and 50 iterations respectively.
The following Figure 4.20 is representation of evaluation of running time taken for the proposed approach to optimize multi-relational association rules of the dataset 4.

In this section, the running time of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous
algorithm, there are 1029 number of rules are generated for the dataset 4 for the minimum support 5. To optimize 1029 number of rules at 10 number of iteration, the proposed algorithm needs 1 second and it takes 8 seconds 48 seconds for 30 iterations and 50 iterations.

4.4.4 Evaluation of Memory Usage

The following Figure 4.21 is representation of evaluation of memory usage for the proposed approach to optimize multi-relational association rules of the dataset 1. In this section, the memory usage of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous algorithm, there are 60848 number of rules are generated for the dataset 1 for the minimum support 3. To optimize 60848 number of rules at 10 number of iteration, the proposed algorithm needs 9268832 bytes and it takes 9362864 bytes 9186496 bytes for 30 iterations and 50 iterations.

![Figure 4.21](image)

Figure 4.21 Evaluation of Memory Usage based on Number of Iterations for the Dataset 1
The following Figure 4.22 is representation of evaluation of memory usage for the proposed approach to optimize multi-relational association rules of the dataset 2. In this section, the memory usage of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous algorithm, there are 18797 number of rules are generated for the dataset 2 for the minimum support 4. To optimize 18797 number of rules at 10 number of iteration, the proposed algorithm needs 4758346 bytes and it takes it takes 4843534 bytes 4959965 bytes for 30 iterations and 50 iterations.

![Figure 4.22 Evaluation of Memory Usage based on Number of Iterations for the Dataset 2](image)

The following Figure 4.23 is representation of evaluation of memory usage for the proposed approach to optimize multi-relational association rules of the dataset 3. In this section, the memory usage of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous algorithm, there are 2047 number of rules are generated for the dataset 3 for the minimum support 2. To optimize 2047 number of rules at 10 number of iteration, the proposed algorithm needs 3211552 bytes and it takes it takes 3148072 bytes 3463432 bytes for 30 iterations and 50 iterations.
The following Figure 4.24 is representation of evaluation of memory usage for the proposed approach to optimize multi-relational association rules of the dataset 1. In this section, the memory usage of the proposed algorithm is evaluated based on the values of number of iteration. Based on the previous algorithm, there are 1029 number of rules are generated for the dataset 4 for the minimum support 5. To optimize 1029 number of rules at 10 number of iteration, proposed algorithm needs 423504 second and it takes 430408 bytes for 30 and 50 iterations.
4.4.5 Evaluation of Number of Rules Reduced by Proposed Algorithm

The following Figure 4.25 is representation of the number of reduced multi-relational association rules by the proposed algorithm based on the various minimum support value, fitness values and number of iterations of the dataset 1. For the dataset 1, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, previous algorithm produces 310350, 60848, 19406, 9048 numbers of multi-relational associations rules. These rules are optimized by the proposed algorithm, which is represented in the following Figure 4.25.

![Figure 4.25 Number of Rules Reduced for the Dataset 1](image)

The following Figure 4.26 is representation of the number of reduced multi-relational association rules by the proposed algorithm based on the various minimum support values, fitness values and number of iterations of the dataset 2. For the dataset 2, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational
association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 131824, 130892, 18797, 5906 numbers of multi-relational associations rules. These rules are optimized by the proposed algorithm, which is represented in the following Figure 4.26.

![Dataset 2](image)

**Figure 4.26 Number of Rules Reduced for the Dataset 2**

The following Figure 4.27 is representation of the number of reduced multi-relational association rules by the proposed algorithm based on the various minimum support value, fitness values and number of iterations of the dataset 3. For the dataset 3, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 2047, 2047, 1023, 1023 numbers of multi-relational associations rules. These rules are optimized by the proposed algorithm, which is represented in the following Figure 4.27.
Figure 4.27 Number of Rules Reduced for the Dataset 3

The following Figure 4.28 is representation of the number of reduced multi-relational association rules by the proposed algorithm based on the various minimum support value, fitness values and number of iterations of the dataset 4. For the dataset 4, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 7630, 2467, 1494, 1023 numbers of multi-relational associations rules. These rules are optimized by proposed algorithm, which is represented in the following Figure 4.28.

Figure 4.28 Number of Rules Reduced for the Dataset 4
4.4.6 Evaluation of Memory Usage

In this chapter, the memory usage is evaluated by calculation of how much memory space need for the proposed algorithm to optimize the various number of outcome multi-relational association rules. The various number of rules are obtained by varying the minimum support values of the previous algorithm.

The following Figure 4.29 is representation of the memory usage of the proposed algorithm to optimize the number of rules of the previous algorithm. The memory usage of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of iterations of the dataset 1. For the dataset 1, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically.

Figure 4.29 Evaluation of Memory Usage by varying the Min-support of Dataset 1
For the minimum support values 2, 3, 4, 5, the previous algorithm produces 310350, 60848, 19406, 9048 numbers of multi-relational associations rules. The memory needed for the proposed algorithm to optimize the above set of rules for various values of fitness and number of iterations is represented in the Figure 4.29.

The following Figure 4.30 is representation of the memory usage of the proposed algorithm to optimize the number of rules of the previous algorithm. The memory usage of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of iterations of the dataset 2.

![Figure 4.30](image)

**Figure 4.30** Evaluation of Memory Usage by varying the Min-support of Dataset 2

For the dataset 2, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the minimum support values
2, 3, 4, 5, the previous algorithm produces 131824, 130892, 18797, 5906 numbers of multi-relational associations rules. The memory needed for the proposed algorithm to optimize the above set of rules for various values of fitness and number of iterations is represented in the Figure 4.30.

The following Figure 4.31 is representation of the memory usage of the proposed algorithm to optimize the number of rules of the previous algorithm. The memory usage of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of iterations of the dataset 3.

![Figure 4.31](image-url)  
**Figure 4.31  Evaluation of Memory Usage by varying the Min-support of Dataset 3**

For the dataset 3, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 2047, 2047, 1023, 1023
numbers of multi-relational associations rules. The memory needed for the proposed algorithm to optimize the above set of rules for various values of fitness and number of iterations is represented in the Figure 4.31.

The following Figure 4.32 is representation of the memory usage of the proposed algorithm to optimize the number of rules of the previous algorithm. The memory usage of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of iterations of the dataset 4. For the dataset 4, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 7630, 2467, 1494, 1023 numbers of multi-relational associations rules. The memory needed for the proposed algorithm to optimize the above set of rules for various values of fitness and number of iterations is represented in the following Figure 4.32.

![Figure 4.32](image)

**Figure 4.32** Evaluation of Memory Usage by varying the Min-support of Dataset 4
4.4.7 Evaluation of Running Time

In this chapter, the memory usage is evaluated by calculation of how much time required this algorithm to optimize the various number of outcome multi-relational association rules. The various number of rules are obtained by varying the minimum support values of the previous algorithm.

The following Figure 4.33 is representation of the required running time for the proposed algorithm to optimize the number of rules of the previous algorithm. The running time of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of iterations of the dataset 1.

![Figure 4.33 Evaluation of running Time by varying the Min-support of Dataset 1](image)

For the dataset 1, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 310350, 60848, 19406, 9048 numbers of multi-relational associations rules. The required running
time for the proposed algorithm to optimize the above set of rules for the various values of fitness and number of iterations is represented in the Figure 4.33.

The following Figure 4.34 is representation of the required running time for the proposed algorithm to optimize the number of rules of the previous algorithm. The running time of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of iterations of the dataset 2.

![Dataset 2](image)

**Figure 4.34 Evaluation of running Time by varying the Min-support of Dataset 2**

For the dataset 2, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 131824, 130892, 18797, 5906 numbers of multi-relational associations rules. The required running time for the proposed algorithm to optimize the above set of rules for various values of fitness and number of iterations is represented in the Figure 4.34.
The following Figure 4.35 is representation of the required running time for the proposed algorithm to optimize the number of rules of the previous algorithm. The running time of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of iterations of the dataset 3. For the dataset 3, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 2047, 2047, 1023, 1023 numbers of multi-relational associations rules.

**Figure 4.35 Evaluation of running Time by varying the Min-support of Dataset 3**

The required running time for the proposed algorithm to optimize the above set of rules for various values of fitness and number of iterations is represented in the above Figure 4.35.

The following Figure 4.36 is representation of the required running time for the proposed algorithm to optimize the number of rules of the previous algorithm. The running time of the proposed algorithm is evaluated based on the various minimum support value, fitness values and number of
iterations of the dataset 4. For the dataset 4, in the previous algorithm, when the value of minimum support increased then the number of generated multi-relational association rules get reduced automatically. For the for the minimum support values 2, 3, 4, 5, the previous algorithm produces 131824, 130892, 18797, 5906 numbers of multi-relational associations rules. The required running time for the proposed algorithm to optimize the above set of rules for various values of fitness and number of iterations is represented in the following Figure 4.36.

Figure 4.36 Evaluation of running Time by varying the Min-support of Dataset 4

4.5 SUMMARY

The technique of optimize the multi-relational association rules through the genetic algorithm is discussed here. The GA successfully optimized the rules which provide the most significant set of rules based on the fitness rate of the rule. The output from the proposed algorithm represents that optimization process of the proposed algorithm is processed well under threshold values such as minimum support, fitness value of the rules, minimum fitness rate etc. The outcome result of the proposed algorithm
denotes that the proposed algorithm has improved well in terms of mining the multi-relational association rules. The result part of the proposed algorithm defines that more significant amount of rules which was obtained by the minimum fitness rate and here the proposed algorithm is evaluated in terms of running time and memory usage and number of rules mined.