Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)

Data Mining based Query Processing using Rough Sets and Genetic Algorithms

Srinivasa K G and Jagadish M
Dept. of Computer Science
M S Ramaiah Institute of Technology
Bangalore 560 054
kgsrinivas@msrit.edu, jagadish88@gmail.com

Venugopal K R
Dept. of Computer Science
UVCE, Bangalore University
Bangalore 560 001
venugopalkr@gmail.com

L M Patnaik
Microprocessor Applications Laboratory
Indian Institute of Science
Bangalore, India
lalit@micro.iisc.ernet.in

***Abstract*—The optimization of queries is critical in database management systems and the complexity involved in finding optimal solutions has led to the development of heuristic approaches. Answering data mining query involves a random search over large databases. Due to the enormity of the data set involved, model simplification is necessary for quick answering of data mining queries. In this paper, we propose a hybrid model using rough sets and genetic algorithms for fast and efficient query answering. Rough sets are used to classify and summarize the datasets, whereas genetic algorithms are used for answering association related queries and feedback for adaptive classification. Here, we consider three types of queries, i.e., select, aggregate and classification based data mining queries. Summary tables that are built using rough sets and analytical model of attributes are used to speed up select queries. Mining associations, building concept hierarchies and reinforcement of reducts are achieved through genetic algorithms. The experiments are conducted on three real-life data sets, which include KDD 99 Cup data, Forest Cover-type data and Iris data. The performance of the proposed algorithm is analyzed for both execution time and classification accuracy and the results obtained are good.**

***Index Terms*—Rough Sets, Genetic Algorithms, Query Answering, Optimization.**

I. Introduction

Major issues in data mining are the mining methodology and user interaction issues. The mining methodology is concerned with coverage of wide spectrum of data analysis and user interaction deals with interactive mining of knowledge at multiple levels of abstraction with reference to the domain knowledge. Just as relational query languages allow users to pose ad-hoc queries, data mining query languages(DMQL) have been developed to describe ad-hoc mining tasks by facilitating the specification of the relevant knowledge, kinds of knowledge to be mined and the condition and constraints to be enforced on discovered patterns. Some of the commands used in knowledge discovery to specify the kinds of knowledge to be mined include (i) Generalized relations (ii) Characteristic rules (iii) Discriminant rules (iv) Classification rules (v) Association rules.

The syntax of DMQL is close to that of SQL and is generally of the form,

```
use database <database_name>
{use hierarchy <hierarchy_name>
  for <attribute>}
<rule_spec>
related to <attr_or_agg_list>
from <relation(s)>
[where <conditions> ]
[order by <order list>]
{with [<kinds of>] threshold=<value>
[for <attribute(s)>]}
```

The optimization of queries is critical in aspect of database management and the exponential complexity involved in finding optimal solutions has led to the development of heuristic approaches. Here, we experiment with the concepts of rough sets and genetic algorithms in order to reduce the query execution time with approximate reasoning of data, without going into exact statistical measures. Related work can be found in [6]–[12].

II. Problem Definition

Assume that the entire database is represented as a single relational table *R* with attributes $A_1, A_2, A_3, \ldots A_n$. Let the number of tuples in *R* be *N* with each tuple identifiable by a $tuple - id$, where t_i represents the i^{th} tuple. The objective of this paper is to efficiently answer the queries belonging to the following three categories.

1) An information retrieval query whose purpose is to search the tuples satisfying one or more conditions usually specified by the WHERE clause
2) An aggregate query which involves the extraction of statistical information that does not exist as it is, but have to be derived from the existing attributes.
3) The third type of queries are those that exist for the discovery of knowledge and patterns in an information system. These involve mining characteristics, identifying associations, dynamic classification defining hierarchies and visualization among others.

Assumptions:

1) The proposed framework works in conjunction with the existing information systems.
2) Only the queries belonging to the categories as mentioned above are executed by the proposed framework, whereas, other types of queries are executed by the existing information systems.

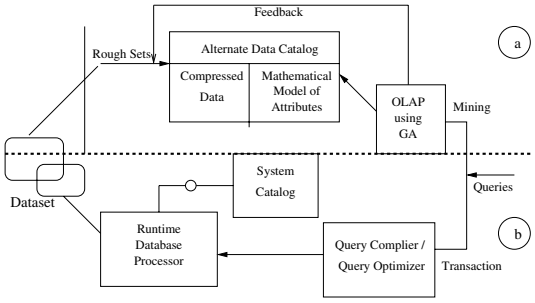


Fig. 1. Architecture of the Proposed Model

3) The entire dataset is represented as a single relational table.

III. ARCHITECTURE

A data mining query language that allows ad-hoc mining is used for this purpose. The DMQL adopts SQL like syntax so that it can be seamlessly integrated with relational query language SQL.

Figure 1 shows the block diagram of the proposed frame work {Upper Section(a)} in conjunction with the existing information system{Lower Section(b)}. Here, a classification based information rich alternate data catalog is built using roughsets on the primary data source. Only the essential features that best describe the database are stored in the catalog in the form of summary tables. Genetic algorithm-based approach is used for association and concept hierarchy queries, in which the summarized models are used in order to reduce the execution time. Genetic algorithms are also used as feedback to improve the accuracy of classification. Only the queries belonging to the three categories as mentioned in the problem definition are answered by the proposed framework (Figure 1, section (a)) , whereas, the remaining queries are answered by the existing system (Figure 1, section (b)).

The proposed framework as shown in Figure 1, seeks to speed up the processing time required to produce results for all the three types of queries, where the results obtained from the proposed system slightly differ from those obtained by querying the actual database.

For the first type of query, the performance is measured by the commonality between the results obtained. If T is the set of $tuple - id$s that are retrieved for an actual query and T' is the set obtained from the proposed framework; the quality can be measured as, $\alpha = |T \cap T'|/|T|$ and $\beta = |T - T'|/|T|$ where α represents the relevance factor and β signifies the margin of error. For other types of queries, the deviation of the obtained result from the actual one is quantifiable and can be flexibly changed by the user. Mining related queries such as finding association, characteristics, concepts hierarchies are evaluated based on the percentage of data sets that are correctly classified. The approach given here focuses on efficient results with lower query-execution times.

A. Rough Sets

The rough set theory, despite being relatively simple, has the capability to deal with imperfections, such as noise and unknown values. Some of the concepts that are relevant to this article are briefed here. Details can be referred from [1]–[3] and [4].

1) Information System: An Information System is a set of objects with attributes related to it.

An example of decision system is shown in Table 1, with income being the decision attribute.

Table 1: An Example of Decision System

	studies	education	works	income
1	no	good	yes	high
2	no	good	yes	high
3	yes	good	yes	none
4	no	poor	no	low
5	no	poor	no	medium

2) Indiscernibility: With every subset of attributes $B \subseteq A$ in the IS $\mathcal{A} = (U, A)$, an equivalence relation IND(B) called an Indiscernibility Relation is associated: which is defined as follows:

$$IND(B) = \{(x, y) \in U^2 | a(x) = a(y) \forall a \in B \ \}$$

By definition, $U/IND(B)$ is the set of all equivalence classes in relation $IND(B)$. From Table 1, it can be observed that,

$$U/IND(\{studies, education, works\}) = \{\{1, 2\}, \{3\}, \{4, 5\}\}$$

The objects that are grouped together cannot be discerned between one another when using the selected set of attributes. The equivalence class is formed with such a group. In Table 2, it can be observed that, class E_1 comes from objects 1 and 2, class E_2 from object 3, while class E_3 comes from objects 4 and 5.

Table 2 : Equivalence classes

	studies	education	works
E_1	no	good	yes
E_2	yes	good	yes
E_3	no	poor	no

The equivalence classes induced by Indiscernibility relation are known as granules. The partition induced by equivalence relation can be used to build new subsets of the universe.

3) Reducts: Indiscernibility relation reduces the data by identifying equivalence classes, using the available attributes. Only one element of the equivalent class is needed to represent the entire class. The minimal set of attributes min_A are taken from initial relation *A*, such that min_A induces same partition on the domain of DS as done by *A*. The above set of attributes are called reducts. Reducts have been appropriately characterized in [4] by *discernibility matrices* and *discernibility functions*. For a set of attributes $B \subseteq A$ in $A = (\mathcal{U}, A)$, the Discernibility Matrix $M_D(B) = m_D(i, j)_{nxn}$ $1 \leq i, j \leq$

$n = |U/IND(B)|$, where

$$m_D(i,j) = \{a \in B | a(E_i) \neq a(E_j)\} for\ i,j = 1,2,\ldots,n.$$

The entry $m_D(i,j)$ in the discernibility matrix is the set of attributes from B that discern object classes $E_i, E_j \in U/IND(B)$.

Table 3 : Discernibility Matrix

	E_1	E_2	E_3
E_1	-	studies	education works
E_2	studies	-	studies education works
E_3	education works	education works studies	-

The Discernibility Function $f(B)$ of a set of attributes $B \subseteq A$ is

$$f(B) = \bigwedge_{i,j \in \{1\ldots n\}} \bigvee \overline{m}_D(E_i, E_j)$$

where $n = |U/IND(B)|$, and $m_D(E_i, E_j)$ is the disjunction taken over the set of boolean variables $m_D(i,j)$ corresponding to the discernibility matrix element $m_D(i,j)$. The relative discernibility function $f'(B)$ computes the minimal sets of attributes required to discern any equivalence class from all the others. Similarly, the relative discernibility function $f'(E,B)$ computes the minimal sets of attributes required to discern a given class E from the others.

The following relative discernibility functions can be calculated:

f'(E_1, C) = studies ∧ (education ∨ works)
f'(E_2, C) = studies ∧ (studies ∨ education ∨ works)
f'(E_3, C) = (education ∨ works) ∧
(studies ∨ education ∨ works)

$Dispensibility$: An attribute a is said to be dispensable or superfluous in $B \subseteq A$ if $IND(B) = IND(B - \{a\})$, otherwise the attribute is indispensable in B.

From Table 3, it can be noted that, $IND(C) = IND(C - \{works\}) = IND(C - \{education\})$. The only dispensable attribute is $studies$.

It is clear that all the attributes are not needed to determine the classification of the dataset. *Dispensable* attributes should be mapped to the attributes from which its value can be derived. The attribute mapping table (Table 4) is constructed so that the query containing any of these attributes can be translated if needed, before execution. For example, the query

```
select ... from <table> where works="yes";
```

can be translated to

```
select ... from <table>
where education="good";
```

Table 4: A mapping of attributes to reducts

$\mathcal{A}$	$\mathcal{R}$
A_1	R_1
A_2	R_1
A_3	R_1
A_6	R_2
A_7	R_2

The above methodology is described for discrete attributes, whereas slight modifications are required to make the rough set to work for continuous attributes. The difficulty of continuous-type attributes arises due to the fact that they cannot be used to partition the data set into classes based on their values since the number of values they assume are nearly the same as the number of tuples present in the dataset.
The usual approach taken during the cleaning process is to quantize the values into classes and then treat the attributes like a discrete valued function [13]. However in this case the complexity is increased by the fact that the classifier needs to produce efficient results for user-invoked SQL-like queries. The approach taken here is predominantly analytical in nature described in section V.

B. Information Streaks

1) TupleID: Assume that each of the tuples in the data table is identified by a unique identifier called *TupleID*. The primary attribute can also be used as unique identifier. However, for the explanation we assume *TupleID* to be just an integer identifying the position of the tuple. The objective of finding information streaks(consecutive tuples) is to identify $TupleID$-ranges such that tuples within the range belong predominantly to an information class. Each range value contains the summarized information about the attributes it is built upon.

The pseudocode used to find the information streaks is given in Algorithm 1. Its purpose is to find tuple-ranges of size $\geq l$ in the entire dataset such that tuples in the range can be considered to belong to an information class(E). In order to avoid fragmentation, w is used as a weighing factor which proportionally increases with the length of the current streak. The sensitivity of classification can be considered by constants β and β'; p denotes the number of samples taken at each iteration and thus determines the resolution of the classification.

It can be observed that the above algorithm resembles clustering with proximity consideration. Clustering takes into account all the classes and hence tuples are bound to be well distributed in the entire data space. Accessing tuples belonging to one cluster would mean many blocks of data being read, causing time overhead. Information Streaks would overcome this problem with some sacrifice being made to the accuracy of classification. Figure 2 depicts a typical scenario. Left side of Figure 2 shows the data block accesses by the clustering algorithm. Consider the highlighted tuples belonging to a particular information class. The clustering algorithm would

Algorithm 1 Obtaining tuple-ranges

$p:$ *number of samples to be taken at each iteration*
l : *the minimum length for a tentative information range to be accepted*
α :*the tolerance level for a sample belonging to a different class to be included in the current streak*
β, β':*constants to determine the nature of exponential averaging*
w: *information to store the weight associated with each class*
E_k:*information class to be included in the current streak*
$tl:$ *store the tentative length of the streak*
$E_c:$ *current information class*
$p\mathcal{A}:$ *a pseudo tuple fragment*

$s \leftarrow 1$
$tl \leftarrow 0$
while $s \leq lastTupleID$ **do**
  sample tuples with TupleID's from s *to* $s+p$
  produce $p\mathcal{A}$ *whose attribute values are obtained from averaging over the sample set*
  Classify $p\mathcal{A}$ *based on rules deducted earlier(say* E_k*)*
  if tl = 0 **then**
    $w_k \leftarrow \beta + (1-\beta)w_k$
    $E_c \leftarrow E_k$
  else if $E_c \neq E_k$ **then**
    $w \leftarrow (1-\beta)w_k - \beta'$
    if $w_k > \alpha$ **then**
      Add the current sample range to the streak
      $tl \leftarrow tl + p$
    else
      if $tl > l$ **then**
        current streak to the range table with summarized information
      else
        $s \leftarrow s + tl$
        $tl \leftarrow 0$
      end if
    end if
  end if
end while

retrieve all the tuples resulting in four block accesses, whereas only one block is accessed using the information streaks.

IV. Modeling of Continuous-type Data

Continuous-type attributes which were not included in the reduct table are individually represented as mathematical functions. This might seem like an unreasonable assumption to make since real-world data scarcely lend themselves to be fit into analytical deductions. However, it is a feasible concept to use when attributes change gradually over a period of time such as daily temperature, traffic flow, stock index, etc. If A is an attribute that has been represented by a mathematical function $f(t_{id})$, then any query involving condition checking

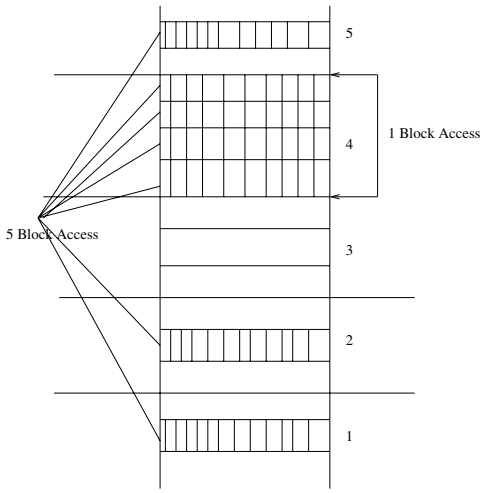


Fig. 2. Comparison of Clustering against Information Streaks

on values of A can be answered by solving $f(t_{id})$, to find the corresponding tuple-id ranges.

All the functions shown in Figures 3 through 6 are obtained by approximating the exchange rates to function of sine and cosine components of three frequencies. The Figure 3 depicts single valued selection for $tupleID$ versus exchange values. The points of intersection give the $tupleID$ satisfying the condition for exchange-rate = 1.2. In Figure 4, the query involving the range function is given. The corresponding tuple ranges are $t_1 - t'_1$, $t_2 - t'_2$ and $t_3 - t'_3$(shaded regions in Figure 4) for satisfying the range between 1.2 to 1.4.

A combination of interval-based sampling and analytical treatment can be obtained if patterns are found in *classes* [13]. The smoothness of data can be achieved by taking the mean for each of τ values, where τ is an user defined value for smoothning the data values.

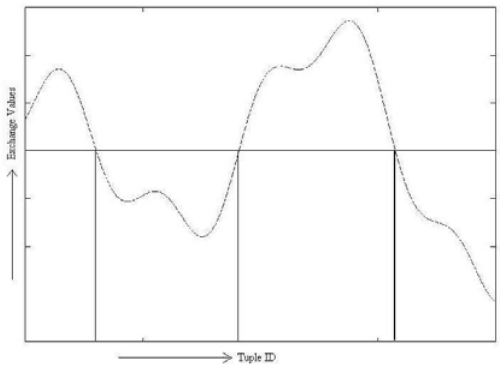


Fig. 3. A single-valued selection

Some of the parameters involved in projection of tuples are given below:

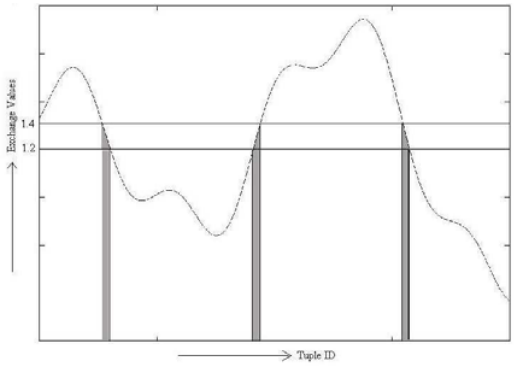


Fig. 4. Range query involving " $SELECT \ldots WHERE 1.2 <= EXCHANGE < 1.4$

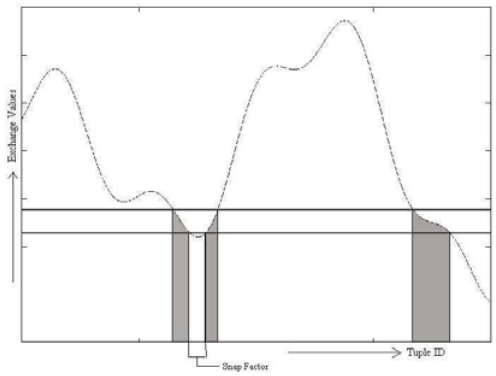


Fig. 6. Snap factor

(i) Marginal error width (m_e) : Since the projections are not accurate enough to pick single-values, the number of adjacent tuples to be included is determined by m. Selection in Figure 4 cannot be accurate. Therefore, adjacent tuples around the point of selection are also considered. The width of the selection defines the marginal error. This scenario is shown in Figure 5.

(ii) Snap factor (s_n) : A width lower than which two adjacent projection ranges can be merged, s_n appears to achieve the same function as that of τ in attempting to coalesce closely split tuples. However, a closer look would reveal that τ is a parameter used when the attribute model is $built$ and hence a change in τ would require that data blocks are re-read, whereas s_n is a tunable factor that can be set at the time of execution of a query. Figure 6 shows the effect of snap factor, where the two tuple ranges with the distance lesser than s_n are merged into a single range.

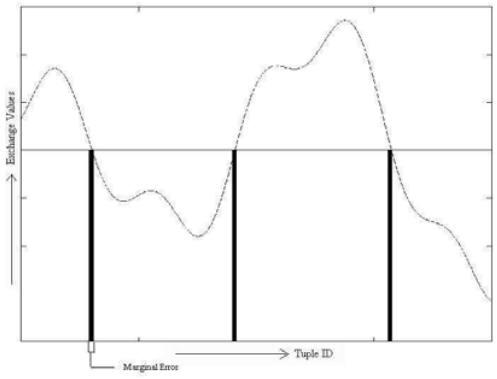


Fig. 5. marginal error selection

V. Genetic Algorithm Model

A genetic algorithm uses mechanics of natural selection to guide search. A search algorithm balances the need for exploration to avoid local optima. Genetic algorithms dynamically achieve this balance through the recombination and selection operators [5], [15]. The steps taken by a genetic algorithm in the context of classification or association rule discovery are:

- The input to the algorithm is a collection of training data.
- The set is encoded into a structure capable of being used for genetic algorithms.
- An initial population is randomly generated.
- The evolution process on populations brings out a set of class models.

A. GA and query languages

The search bias during genetic search depends on the kind of problem solved, the structure of search space and the genetic operators. There are two possible ways to build meaningful blocks

1) Search through possible encodings for a good one while searching for a solution.
2) Expand the number of good encodings by increasing the types of building blocks considered meaningful.

Here, task-based specific queries are posed in form of DMQL queries. Some of the popular mining characteristics sought are comparison, classification, associations and concept hierarchies.
The specification of each query may be given as:

```
<Mine_Knowledge_Specification> ::=
            <Mine_Char>  |
            <Mine_Discr> |
            <Mine_Assoc> |
            <Mine_Class>
```

In general initial population is created consisting of randomly generated rules. Each rule represented by a string of bits. For example, if the training set consist of two attributes

A_1 and A_2 and a class C, then the rule "IF A_1 AND NOT A_2 THEN C" can be encoded as 101. Attributes with more values can be encoded using more bits, while continuous attributes can be encoded after interval-based classification. The fitness of a rule is assessed by its classification accuracy on the dataset. The general structure of GA is as follows :

```
t=0;
P(t) = Initialize Random Population
       (no_attributes, attribute_domains);
while ( t < max_generations )
      Evaluate fitness (P(t), dataset)
      t = t+1
      P(t) = select(P(t-1))
      Crossover(P(t));
      Mutate(P(t))
end while
```

the modified version is,

```
t=0;
P(t) = Generate biased Population
       based on query attributes
while (t<max_generations)
   Evaluate fitness(P(t), summarized data)
   t = t+1
   P(t) = select(P(t-1))
   Crossover(P(t));
   Mutate(P(t))
end while
```

1) Associations: Each rule can be represented as a string. In generation of rules along with the support of a pattern, the number of set positions are also important. A pattern full of don't cares will gain a support of 100% but has no meaning in terms of knowledge discovery. Each discovered rule in the rule set is usually represented in the form

$$IF < condition_1 > \& < condition_2 > \ldots \& < condition_n > THEN < action >$$

There are various representation methods for conditions and actions in terms of rule properties (fuzzy or non-fuzzy) and the attribute properties(continuous or discrete). A rule set supposed to be the solution for a classification problem. Processing a query which is of the form,

```
<Mine Assoc> ::= mine associations
                 [as <pattern name>]
                 [match <meta-pattern>]
```

involves the extraction of rules consisting of two parts: searching for hidden patterns and generation of rules based on those patterns. After the validation of candidates, the rules are selected based on expected lends of support and confidence. Let the attributes selected to form a classification rule R_i are $< A_1, A_2, \ldots >$, out of which b attributes are of boolean type, and k of continuous and p_i denotes the number of intervals of A_i^{th} attribute and d the number of discrete attributes with each taking at the most d_i values. The length of the binary chromosome is given by :

$$l_b = b + log\left(\sum_{i=0}^{k} p_i\right) + log\left(\sum_{j=0}^{d} d_j\right) + m$$

where m is the length of the consequent.

The fitness of a chromosome reflects the success rate achieved and the corresponding rule set is used for classification. The GA operators use this information to evolve better chromosome over the generations. The fitness function actually measures the collective behavior of the rule set. The evaluation of the fitness can be speeded up using the summarized table built from the reduct rules and information streaks. In case of rules exactly matching the reduct rules, the support and confidence measures of earlier classification can be directly used with a new threshold value. The decomposition of a query in order to find rules already computed is a part of query relaxation. Query relaxation involves the rewriting of query to form a new query. Some of the main reasons for relaxation are

- the query is too general or too specific
- some of the query terms may not be used in the database
- one or more query terms have multiple meanings subject to context.

One approach used to deal with the above scenarios is generalization which involves rewriting of a query to a more generalized term based on information found in association mappings and attribute transformation. It is also possible to reduce associated clauses in the query to a single generalized attribute based on reduct table.

2) Concept Hierarchies: A frequently used and an important type of query is the concept hierarchy query, which usually takes the form

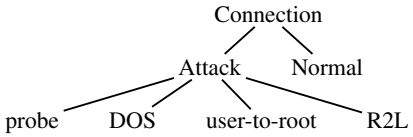
```
<Concept_Hierarchy_Definition_Statement>
     ::= define hierarchy< hierarchy_name>
         [for <attribute_or_dimension> ]
         on <relation_or_cube_or_hierarchy>
         as <relation_description>
         [where <condition>]
```

Concept hierarchies allow the mining of knowledge at multiple levels of abstraction. They define a sequence of mappings for a set of a low-level concepts to higher-level more general concepts. A concept hierarchy is represented as a set of nodes organized in a tree, where each node in itself, represents a concept. The common types of hierarchies are

- schema hierarchies which express semantic relationship between attributes.
- set-grouping hierarchies that organizes values for a given attribute or dimension into groups of constants, typically used for defining small sets of object relationships.
- operational-derived hierarchies is based on operations specified by users, experts or the data mining systems.

- rule-based hierarchies that occur when either the whole concept hierarchy or a portion of it defined by a set of rules.

Since problem of concept hierarchies involve classification of attribute values with multiple levels of abstraction genetic algorithm can be used with a slight modification of including multiple types of chromosomes each defining the strength of classification for each level of abstraction.
It can be observed that the number of features that have to be used to describe a particular class in the hierarchy may be different from one another. Consider the example of classification of connection types in KDD99-Cup Dataset, the number of attributes(features) that are needed to predict a normal connection is only four, while predicting the exact class of attack requires more than six attribute values. The concept hierarchy for KDD99-Cup dataset is shown below.



The difference in the number of bits to represent features demands the use of chromosomes with different length and a non-conventional type of crossover. A simple approach to take is to consider each path of the concept hierarchy as a series of classification where the support for a lower level concept *'p'* is based on the percentage of data objects belonging to its immediate higher level concept. The automatic building of concept hierarchies is close to concept maps and ontologies [14], [16].

VI. Experimental results

Experiments are performed on three real-life data sets taken from UCI Machine Learning Archive [17]. The characteristics of data sets are summarized below:

1) *KDD 99 Cup data:* The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. The attacks fall into four categories as DOS(denial of service), R2L, U2R and probing. The datasets contain a total of 24 training attack types. The attributes describe the basic features of TCP connections, content features within the connection suggested by domain knowledge and traffic features.
2) *Forest Cover-type:* It is a Geographical Information System data representing forest cover type like pine, fir, etc. found in US. The variables are cartographic and remote sensing measurements. It contains 10 dimensions, seven classes and 586,012 samples. All the attribute values are numerical.
3) *Iris:* A data set with 150 random samples of flowers from the iris species setosa, versicolor, and virginica. There are four features all of which are numeric.

A. Classification

The classification accuracy is defined as the percentage of data points whose class labels are correctly predicted. The classification accuracy of the datasets considered above is shown in Table 5.

Table 5: Classification Accuracy of all datasets

Dataset	Accuracy(%)
KDD 99	98.3
Covertype	64.2
Iris	97.6

The size of summarized tables expressed as percentage of size of the original dataset is shown in Table 6. n_a denotes the number of attributes that are the part of the summarized tables.

Table 6: The sizes of summarized tables

Dataset	n_a	$size$
KDD 99	12	7.1%
Covertype	6	6.8%
Iris	4	2.6%

(i) Relation between l and classification accuracy

Figure 7 shows the variation of classification accuracy versus minimum streak length l for KDD99 Cup data, Forest Cover-type data and Iris databank. As the minimum length of the information streak is increased the resolution of the classification decreases, hence the drop in accuracy.

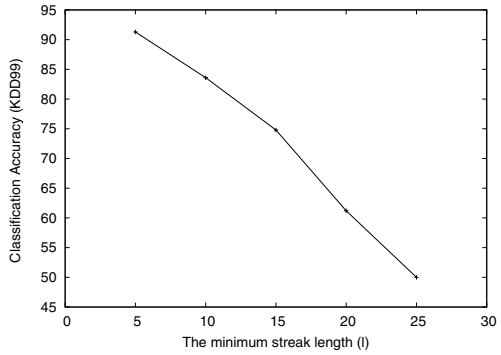


Fig. 7. Variation of Accuracy versus minimum streak length

(ii) Aggregate functions

A random set of queries were generated involving aggregate functions count, aggregate and max/min to test the performance of the proposed methods. For count and aggregate operations the error is the difference between the values obtained, while for max/min operations it is the percentage of times the correct max/min value was returned. Table 7 shows the comparison of execution times for three aggregate functions. t denotes the average time that was taken to execute the queries directly on the database and t_s

is the average time of execution when summarized tables were used. The last column shows the % error which is the deviation of the numerical value obtained from the actual value.

Table 7: Comparison of execution times for Aggregate Function

Dataset	query-type	t (secs)	t_s (secs)	% Error
KDD 99	count	40.3	2.1	10.2
	avg	51.3	2.2	11.6
	max/min	43.3	2.1	23.0
				height
Cover	count	63.1	5.1	14.3
	avg	71.3	6.2	13.6
	max/min	61.3	5.2	29.6
				height
Iris	count	0.5	0.02	10.1
	avg	0.45	0.02	9.6
	max/min	0.32	0.02	14.1

B. Accuracy of Concept Hierarchies

The accuracy of the hierarchies is calculated with respect to each level. If p^{ij} denotes the set of tuples at j^{th} class in the i^{th} level and c^{ij} the corresponding number that have been classified correctly, then the accuracy of i^{th} level is

$$\frac{\sum_{j=1}^{n} C_{ij}/P_{ij}}{n}$$

where n is the number of classes at that level. Overall hierarchy accuracy is expressed as average of all the levels. The experiments are performed on only two of the data sets, the results of which is shown in Table 8.

Table 8: Average concept hierarchy accuracy

Dataset	Accuracy
KDD 99	95.9%
Covertype	61.2%

C. Analysis of improvement with GA feedback

The datasets are increased by 10% and experimented with GA feedback. A slight improvement is seen in two data sets and there is no change observed in Iris dataset due to the absence of formation of new rules. The results are tabulated in Table 9.

Table 9: Classification Accuracy with GA feedback

Dataset	Classification Accuracy
KDD 99	98.9
Covertype	66.2
Iris	97.6

VII. Conclusion

In this paper we have proposed an intelligent query answering system using rough sets and genetic algorithms. The flexibility involved in building the summary tables of rough sets makes the system more scalable. The system is fast even with acceptable level of accuracy as justified in the experiments. Reinforcement of reducts with genetic algorithms leads to adaptive classification. The proposed framework can be used as alternative to system catalog. In future work, the system can be extended to other types of DMQL queries, since it is handling only a subset of queries.

Acknowledgments

The Project is partially supported by the AICTE, as a part of Career Award of Young Teachers(AICTE File No.: F. No.1-51/FD/CA/(9)2005-06) to Mr.K.G. Srinivasa, who is presently working as a faculty in Department of Computer Science and Engineering, M. S. Ramaiah Institute of Technology, Bangalore – 560 054, India.

References

[1] Z. Pawlak,*"Rough Sets,"* Int'l J. Computer and Information Sciences, Vol.11, 1982.

[2] Lingras. P,*"Application of Rough Patterns,"* Rough Sets in Data Mining and Knowledge Discovery, Series Soft Computing, Physics Verlag, 1998.

[3] Lingras. P and Davis. C, *"Application of Rough Genetic Algorithms,"* Computational Intelligence, 2000.

[4] A. Skowron and C. Rauszer, *The Discernibility Matrices and Functions in Information Systems*, Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, pp.331-362, Dordrecht:Kluwer Academic, 1992.

[5] Holland. J. H, *"Adaption in Natural and Artificial Systems,"* Series Soft Computing, Physics Verlag, 1975.

[6] S. M. Weiss and C. A. Kulikowski, *Computer Systems that Learn:Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* San Maeto, CA: Morgan Kaufmann, 1991.

[7] S. U. Guan and S. Li,*Incremental Learning with Respect to New Incoming Input Attributes*, Neural Processes. Lett., Vol 14, no. 3, pp.241-260, 2001.

[8] L. Su, S. U. Gain, and Y. C. Yeo, *Incremental Self-Growing Neural Networks with the Changing Environment*, J. Intell. Syst., Vol.11, No. 1, pp.43-74, 2001.

[9] K. A. DeJong and W. M. Spears, *Learning Concept Classification Rules using Genetic Algorithms*, in Proc. 1991 Int. Joint conf. Artificial Intelligence, 1991, pp.651-656.

[10] T.Y.Lin and R. Chen,*"Finding Reducts in Very Large Databases,"* Proc. Joint Conf. Information Science Research, p.350-352, 1997.

[11] Sheng-Uei Guan and Fangming Zhu, *"An Incremental Approach to Genetic-Algorithms-Based Classification,"* IEEE Trans.Systems, Man and Cybernitics-Part B:Cybernitics, Vol.35, No.2, April 2005.

[12] Dominik Slezak and Jakub Wroblewski *"Order Based Genetic Algorithms for the Search of Approximate Entropy Reducts"* Springer-Verlag Berlin Heidelberg, 2003.

[13] R. H. Shumway.*Applied Statistical Time Series Analysis.* Englewood Cliffs, NJ: Prentice Hall, 1988.

[14] Franciso Edson Lopes da Rocha, Julio Valente da Costa Jr and Eloi Luiz Favero *"A New Approach to Meaningful Assessment using Concept Maps Ontologies and Genetic Algorithms,"* Proc. of the first Intl.Conference on Concept Mapping.

[15] Srinivasa K G, Karthik Sridharan, P Deepa Shenoy, Venugopal K R and L M Patnaik, *"Dynamic Migration Model for Self Adaptive Genetic Algorithms,"* In Proc. of Intl. Conf. on IDEAL, 2005, pp. 555-562.

[16] Costa Jr. V., Rocha F. E. L., and Fevero E. L., *"Linking Phrases in Concept Maps in Study on Nature of Inclusivity,"* In Proc. of First Intl. Conf. on Concept Mapping, Navarra, Spain, 2004.

[17] C. L. Blake and C. J. Merz, t*"UCI Repository of Machine Learning Databases,"* Univ of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/ mlearn/,1998.