I.J. Information Technology and Computer Science, **2015, 03, 48-53**
Published Online February 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.03.07

Database Performance Optimization –A Rough Set Approach

Phani Krishna Kishore. M
Department of Information Technology, GVP College of Engineering (Autonomous), Visakhapatnam, India
Email:kishorempk73@gvpce.ac.in

Leelarani Ch.[1]
[1]Department of Computer Applications, GVP College of Engineering (Autonomous), Visakhapatnam, India
Email: leelaranichilukoti@gmail.com

Aditya. P. V. S. S.[2]
[2]Department of Information Technology, GVP College of Engineering (Autonomous), Visakhapatnam, India
Email: adityapvss@gmail.com

Abstract- As the sizes of databases are growing exponentially, the optimal design and management of both traditional database management systems as well as processing techniques of data mining are of significant importance. Several approaches are being investigated in this direction. In this paper a novel approach to maintain metadata based on rough sets is proposed and it is observed that with a marginal changes in buffer sizes faster query processing can be achieved.

Index Term- Database Management Systems, Rough Sets And Data Mining

I. INTRODUCTION

In most of the traditional data base management systems the data is organized in a structured format based on relational data base model, in which the data is stored in the form of tables designed and optimized using a specific normal form. A structured query language is used to retrieve the relevant information from the data base. With the rapid changes in the technology and with the ever growing internet connectivity web based querying of the data bases has been the order of the day. The information is being generated from various sources rapidly; simultaneously the sizes of data bases are growing exponentially. To handle the data effectively and in less time, various optimization techniques at various levels are being implemented [14,15].

As the sizes of data bases are running into terabytes, better approaches are required to bring in a paradigm shift in the design of the data base management systems. In this direction works are initiated by several researchers. Distributed data bases cater to the need to some extent.

Also in case of data mining concepts, extraction of the hidden information from huge data bases in real time has become a challenging task. Developing methods that unifies and interleaves the traditional data base management as well as extracting useful information from the data bases is important. In this direction rough set theory approach offers a promising frame work. Also frame works that maintain metadata effectively are important. There are two approaches, one design of new mechanism altogether to design, store and retrieve the data, second designing a mechanism that acts on top of existing data bases and optimizing the performance. In the first approach new data bases can be built but migrating from the existing set up to new mechanisms is both time consuming and involves financial implications. The second approach is suited for both existing and new databases as well. In this direction a novel method is proposed in the present paper by introducing a layer on top of the existing data bases that will enhance the performance and helps the transition from older data bases into new formats.

A. Related Work

With the exponential growth in the sizes of the data bases, different methodologies and performance evaluation of different mechanisms to speed-up the query processing is gaining importance, as it has commercial value. With the advent of rough set theory several researchers had thrown light in this direction. Rayne et al [1] attempted to evaluate the performance of models based on rough set theory on very large database. They have tested using Oracle running on Windows NT and observed that rough set theory provides a good frame work for querying on large data bases. Rasha Osman et al [2] in their work they presented a detailed survey on the performance of data base design and systems. They presented a categorization of queuing network performance models of data base systems available in literature. Fares N. Almari et al [3] identified the importance of effective measures to test the performance of various mechanisms and examined the performance claims of vendors Oracle VM sever and VMware on the scalability issues. Taniar [4, 5] in his invited talk he presented how parallelism is effective in data-intensive applications and how to develop faster capabilities to support them in terms of indexing, special algorithms on parallel systems and object oriented schemes. T.

Kramberger [6] studied measurements of data base responses on different file systems according to query load.

Xiaohua Tony Hu, T. Y. Lin, Jianchao Han [7] in their work proposed a new rough sets model based on data base systems. They redefined the main ideas of rough set theory in the context of data base theory and take the advantage of set-oriented data base operations. They explored the use of redact and core concepts of rough sets and observed that their model is scalable and efficient when compared with the existing models.

Initiatives are taken up for using rough set theory to model the data bases. Dominik Slezak et al [8] in their work proposed a new method to represent metadata in terms of columns and compress the data into knowledge granules. Based on the query the relevant granules are decompressed and information is retrieved. To implement the procedure the existing tools are not suitable and an exclusive compiler has to be developed.

Dominik Slezak et al [8] also implemented similar techniques in the name of Bright house for Ad-hoc queries on warehouses.

Srinvas K G and Jagadish M et al [9] in their work they propose a hybrid model using rough sets and genetic algorithms for fast and efficient query answering. Basically rough sets are used to classify and summarize data sets and genetic algorithms are used for association related queries and feedback for adaptive classification.

B. Motivation for the Work

In the present paper, it is proposed to bring a layer that maintains metadata designed on the rough set theory frame work so that query processing is much faster with marginal changes in the buffer sizes, so that all the existing data bases can be optimized to the new functionality.

The remaining part of the paper is organized as follows.

In section II, the rough set preliminaries are presented. In section III the proposed method is illustrated. In III.A, the step by step process of the implementation of the idea with an example is described. In III.B, the result analysis and performance evaluation on a sample data base is incorporated. In section IV conclusion are given.

II. Rough Set Preliminaries

Let U be the Universe and let R be an equivalence relation defined on U. Let U/R denote the set of all equivalence classes. For any $X \subseteq U$, the lower approximation, upper approximation are defined by

$L(X) = \cup \{[x]_R / [x]_R \subseteq X\}$,

$U(X) = \cup \{[x]_R / [x]_R \cap X \neq \phi\}$ respectively.

The following is an equivalence relation defined through indiscernibility relation which is being used for attribute reduction,

$IND_B = \{(x_i, x_j) \in U^2 / a(x_i) = a(x_j), \forall a \in B\}, B \subseteq A$.

The equivalence classes are termed as information granules and the level of granularity determines the information quality in terms of approximations. The equivalence relation defined on the data determines the view on the data base. Different equivalence relations produce different outputs. Hence selection of equivalence relation is more important. Data driven or relevant equivalence classes can be defined to suit the specific requirements of the real time data bases.

III. Proposed Method

In the present context, Let $D = (T_1, T_2, \ldots, T_n)$ be a data base with tables $T_1, T_2, \ldots, T_n$ which are designed through a specified normal form.

Let A= {a_1, a2, an} denote the set of all attributes that appear in all the tables of the data base.

The idea is to discretize ranges of the attributes through a specific equivalence relation one for each of the attributes, thus generating information granules. This can be achieved in several ways. Methods like clustering with a suitable distance metric, equi-depth or equi-width binning may be used to generate these information granules.

Based on the granules produced as described above metadata is generated for each of the table T_i indicating the presence of the tuple information for each granule for each attribute. Views are generated for each of such metadata item. The advantage with view lies in the fact that the view definition only resides until it is called, once it is generated it can be treated as good as a base table. Updates on views are also possible.

Once a query is entered, it is parsed in the metadata layer and the suitable ranges of the attributes involved in the query are identified, and the query is processed only on the specific granules that contain the information, to produce the output of the query. The advantages of views lies in the fact that, as long as the data base is running views are as good as base tables and on any update, delete or insert the views will be updated automatically.

Another advantage of the proposed method is that, it can be used on any existing data base without any modification to the data base. Only relevant parts of the tables will be brought to the buffer there by reducing the processing time.

As mentioned earlier there are various discretization methods for the attribute ranges. However in the present case equi-width binning is considered to demonstrate the discretization. For the attribute a_i, let $D_1= [d_1\ d_2]$, $D_2= [d_3\ d_4]$, $D_{k-1}= [d_{k-1}, d_k]$ be the discretized intervals, that forms the partition of the range of the attribute a_{ir}.

The discretization can be based on equi-depth where the range of the attribute is divided equally. In this approach in some ranges the density of tuples may be more in some it may be sparse. This type of binning is suitable if the data is evenly distributed.

If the equi-depth discretization is made, where the ranges are progressively divided by fixing limits on the number of tuples in each view, the number of views will be varying dynamically and suitable for static data bases.

Statistical methods, clustering methods can also be employed based on probability distribution of the attribute values to optimize the storage.

Given a table T, attribute ‘a’, tuples t_n, t_m in T, define the relation,

$R_a = \{(t_n, tm) \in T^2 / d_i \leq a(t_n), a(t_m) \leq d_{i+1}$ for some $1 \leq i \leq k\}$.

Clearly R is an equivalence relation. U/R_a denotes the equivalence classes namely the bands produced by the partition over the range of the attribute ‘a’. The equivalence classes represent data tables which are in line with the original data base design but represents only part of the table(s). These constitute the data granules.

Given a table T, and given an attribute ‘a’ of T, let the range of ‘a’ has been divided into k disjoint parts, then metadata is created in terms of the ranges of ‘a’ and the set of tuples of T corresponding to each range. For each such resultant metadata, a view is created. Thus for entire data base, that is for each of the attribute of each table corresponding views are created. For a given query, if the lower approximations in each attribute satisfy the query requirements, the query is processed on the corresponding views only in the buffer; otherwise the output is generated from the upper approximation.

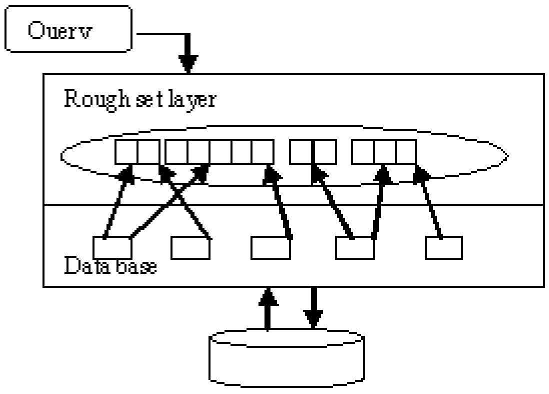

Fig. 1.

A. Process

Input: Database, selected method of partitioning of attribute values.

Step1: For each attributes a_i of each table T_j apply discretization method and produce sub-ranges.

Step2: Create reference tables for each attribute of each table T_j containing the sub-ranges and the tuples present in that range.

Step3: Create views in the data base for each of the ranges specified in step 2.

Input: Query, Lower approximation, Upper approximation

Step 4: Identify the ranges of the attribute which are entirely contained in the ranges specified in the query for the lower approximation/ Upper approximation.

Step 5: Identify the views as created in step 3 and as per the ranges identified in step 4.

Step 6: Process the query on the views to get the lower / upper approximation.

Consider the following example to understand the process. Let us consider an example table as follows:

Suppose a query is passed to find whose salary is greater than 50000 and whose department id is less than 201 then the following values are retrieved,

Table 1. Sample Database

TUPLE ID	EMP ID	NAME	SALARY	DEPT ID	JOB ID
T1	54	Rahul	30000	300	300
T2	01	Tarun	21000	100	200
T3	03	Prannoy	10000	200	100
T4	58	Aditya	58000	100	111
T5	05	Priyan	15010	200	200
T6	28	Praveen	34010	100	300
T7	48	Divya	34010	600	300
T8	13	Harini	33090	400	300
T9	40	Chitti	45000	100	111
T10	56	Prudhvi	56000	200	111
T11	57	Shilpa	47000	500	400

Table 2. Result of the query

TUPLE ID	NAME	SALARY	DEPT ID
T10	Prudhvi	56000	200
T4	Aditya	58000	100

Table 3. Employee ID Attribute

EMP ID	TUPLE ID
1-10	T2.T3.T5
11-20	T8
21-30	T6
31-40	T9
41-50	T7
51-60	T1,T4,T10,T11

Table 5. Salary Attribute

SALARY	TUPLE ID
10000-20000	T3,T5
20001-30000	T1, T2
30001-40000	T6,T7,T8
40001-50000	T9,T11
50001-60000	T4,T10

Table 6. Department Attribute

DEPT ID	TUPLE ID
0-100	T2,T4,T6,T9
101-200	T3,T5,T10
201-300	T1
301-400	T8
401-500	T11
501-600	T7

Table 7. Name Attribute

NAME	TUPLEID
A-C	T4,T9
D-F	T7
G-I	T8
J-L	-----------
M-O	-----------
P-R	T1T3,T5,T6,T10
S-U	T2,T11
V-X	-----------
Y-Z	-----------

Table 4. JobID Attribute

JOB ID	TUPLE ID
100-200	T2,T3,T4,T5,T9,T10
201-300	T1,T6,T7,T8
301-400	T11

Lower and upper approximations-approximate querying:

Suppose a query is given to find employees whose salary is greater than 10,005 and less than 23,000 then we have to utilize two equivalence classes, 10,000-20,000 and 20,001-30,000 to retrieve the output of the query. A new view will be generated from the two equivalence classes which is the upper approximation. Once the view is generated, query operations are done on that view which is now called upper approximation.

From the above table the following metadata can be created and the corresponding views so that any query recalls the corresponding views from this metadata.

B. Implementation and analysis

From the above table all the tuples are identified into the respective ranges and as illustrated below and views are created correspondingly. For implementing the above process a sample data base of size 0.83GB is taken with the schemas as mentioned above with additional tuples.

For different types of queries the performance in terms of buffer space occupied and the time it takes to compute are calculated in ordinary (without rough set method) and using the rough set approach. The following are the observations.

The program is run on Intel core i5 processor with 4 GB RAM and CPU @ 2.50 GHz running on Microsoft Windows 7 Professional operating system when all except vital system services are halted. The SQL 10g expression edition is used to demonstrate the procedure.

We have considered queries in four different categories of queries during the implementation; the types of queries are as follows:

The first 4 queries are simple regular queries. The next 4 queries are conditional queries. The next 4 queries are nested queries and the remaining are queries that operate on joins.

Table 8. Buffer Sizes and Time Consumption Analysis

	Normal execution		Experiment-I		Experiment-II		Experiment-III	
S. No.	Time taken under Normal Execution Conditions (Ms.)	Buffer Utilized under Normal Conditions (bytes)	Time taken under Rough set Theory Execution Conditions (Ms.)	Buffer Utilized under Rough set Theory Conditions (bytes)	Time taken under Rough set Theory Execution Conditions (Ms.)	Buffer Utilized under Rough set Conditions (bytes)	Time taken under Rough set Theory Execution Conditions (Ms.)	Buffer Utilized under Rough set Conditions (bytes)
1	100	5,36,66,144	110	4,89,39,300	82	4,36,12,360	93	4,78,60,448
2	104	6,26,07,528	89	4,76,13,480	83	4,47,11,372	79	4,48,47,504
3	109	5,82,18,328	80	4,91,84,632	91	4,70,68,956	85	4,74,67,232
4	112	6,22,14,312	121	4,63,02,760	90	4,48,42,444	97	4,78,77,260
5	115	6,48,37,908	81	5,00,68,512	95	4,64,79,132	92	5,04,31,452
6	118	6,62,65,044	95	5,33,75,512	96	4,90,50,136	98	5,26,57,964
7	120	6,67,02,672	84	5,38,49,364	94	5,01,13,812	95	5,02,51,656
8	132	5,81,19,168	88	4,79,39,448	87	4,63,64,872	87	4,68,88,344
9	138	6,33,92,248	101	5,29,99,108	90	4,74,94,084	90	4,48,42,444

10	142	6,53,26,416	85	4,91,84,632	80	4,81,00,720	85	4,87,78,916
11	143	6,36,56,104	90	4,89,71,212	100	5,42,42,580	95	4,89,71,212
12	146	6,35,25,032	78	4,69,07,684	78	4,82,10,816	78	4,82,64,748
13	163	6,30,34,368	88	4,93,64,428	83	4,94,98,848	85	4,91,84,632
14	167	6,45,56,796	50	5,10,83,464	41	5,15,57,316	56	5,13,69,784
15	179	6,46,22,332	51	5,06,09,612	66	4,93,96,340	53	5,27,12,416
Total	1988	94,07,44,400	1291	74,63,93,148	1256	72,07,43,788	1268	73,24,06,012

C. Observations

- The total time taken for normal execution of 15 queries is1988 milliseconds with an average of 132.53 milliseconds.
- The average time taken for rough set based execution 1271.66 milliseconds with an average of 84.77 milliseconds.
- Thus on the average time consumption has been reduced by 36.037%
- The total buffer utilization for normal execution of 15 queries is 94,07,44,400 bytes with an average of 6,27,16,293.33 bytes per query (59.81Mb)
- The total buffer utilization (average of three runs) under rough set execution for 15 queries 73, 31, 80,982.66 bytes with an average of 4, 88, 78,732.155 bytes per query (46.61Mb).
- On the average the buffer utilization has been reduced by 22.06%

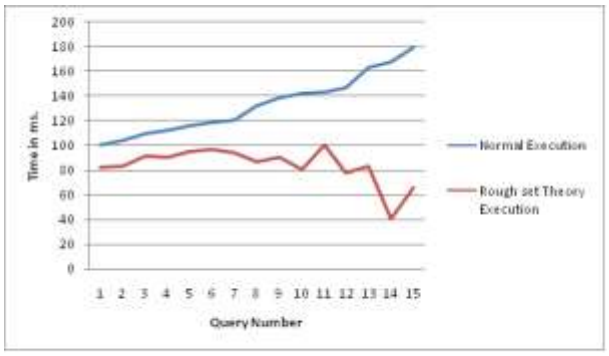

Fig 2. Time consumption

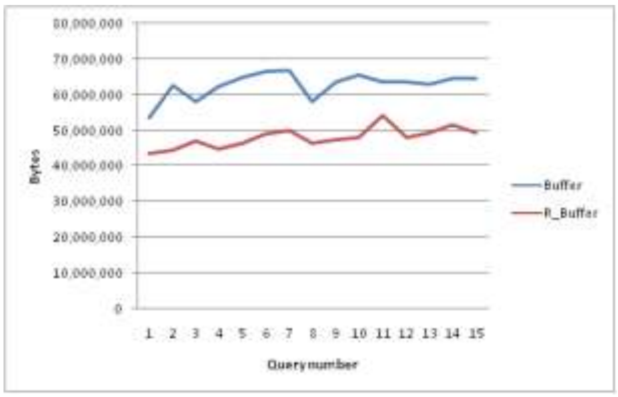

Fig 3. Buffer utilizations

The time utilization chart for Normal execution and with that of rough set based execution is as given below (Fig 2). The first 4queries are simple regular queries. The next 4 queries are conditional queries. The next 4 queries are nested queries and the remaining are queries that operate on joins.

The buffer utilization chart for Normal execution and with that of rough set based execution is as given below (Fig 3). The first 4 queries are simple regular queries. The next 4 queries are conditional queries. The next 4 queries are nested queries and the remaining are queries that operate on joins.

It is observed that the buffer allocation starts at different values for each of the runs. Hence the relative buffer utilization is observed by subtracting the subsequent values of buffers from the starting value (i.e.) with the initial value as reference and observed the following (Fig 4)

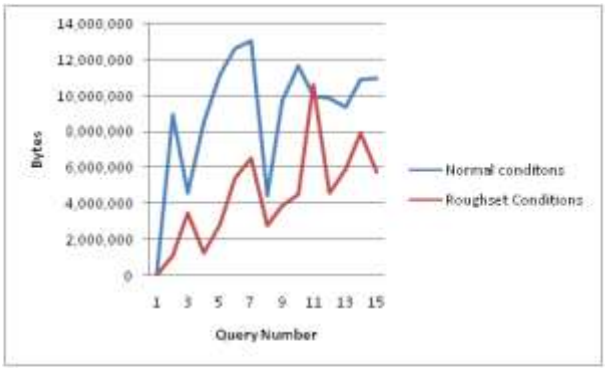

Fig 4. Relative Buffer Utilization

It is observed that the average buffer variations under rough set implementation are44, 37,225.87 bytes against 90, 50,149.333 bytes, so that Variation in buffer utilization is observed to be or the order of 50.97%.

Apart from the above observations the following are also observed.

- When queries are implemented multiple times on same granules comparatively better space and time utilization than normal execution is observed.
- Buffer spaces and time consumption are comparatively higher to that of the other queries based on rough sets when they are operating for the first time on any view.
- The buffer space and time taken over a period of time gets stabilized once all the views are generated.

IV. Conclusion

In this paper an attempt has been made to test the improvement in performance of the data base querying using rough set approach in terms of time and the buffer sizes. It is observed that both buffer sizes, as well as query operational time showed significant improvement. As a tradeoff between the space and time, the proposed new approach is suitable to increase the efficiencies in terms of the time as well as space.

Another advantage of the method lies in the fact that the procedure can be implemented on any existing data base without causing any loss or migration from the existing as this works a layer above the database.

The increase in the performance with respect to time can be attributed to the fact that data that is to be brought into the buffer is less when compared to the data being used in ordinary approach and also operations take less time to retrieve from the information available in buffer when compared with the retrieval from the permanent storage.

When the views are generated in distributed environment then Parallelism can also be increased since queries operating on different ranges of the same attribute can access the data from the respective views in parallel.

References

[1] Rayne Chen, T.Y. Lin, "*Supporting rough set theory in very large databases using Oracle",* RDBMS, 0-7803-3687-9/9601996 IEEE, pp.332-337Z. Pawlak. Rough sets, *"Theoretical aspects of reasoning about data",* Kluwer, 1991.

[2] Rasha Osman, William J. Knottenbelt, *"Database system performance evaluation models: A survey"*, Performance Evaluation 69 (2012) 471–493.

[3] Fares N. Almari, PavolZavarsky, Ron Ruhl, Dale Lindskog, Amer Aljaedi, *"Performance Analysis of Oracle Database in Virtual Environments"* , Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, 2012,pp.1238-1245.

[4] David Taniar, *"High Performance Database Processing"*, Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications, 2012, pp-6.

[5] Taniar, C. H. C. Leung, W. Rahayu, and S. Goel, *"High Performance Parallel Database Processing and Grid Databases",* John Wiley & Sons 2008.

[6] T. Kramberger, D. Cafuta, I. Dodig, *"Database System Performance in Correlation with Different Storage File Systems",* MIPRO-2012, May 21-25, 2012, Opatija, Croatia,913-918.

[7] Xiaohua Tony Hu, T. Y. Lin, Jianchao Han, *"A New Rough Sets Model Based on Database Systems"*, Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Lecture Notes in Computer Science Volume 2639, 2003, pp. 114-121.

[8] Ślęzak D., Wróblewski J., Eastwood V., Synak P, "*Bright house: An Analytic Data Warehouse for Ad-hoc Queries.*" Proc. of VLDB 2008, pp. 1337 - 1345. 2008.

[9] Srinivas K G and Jagadish M, Venugopal K R, L M Patnaik, *" Data Mining Query Processing Using Sets and Genetic Algorithms",* Proceedings of the 2007 IEEE symposium on Computational Intelligence and Data Mining (CIDM2007).

[10] Altigran S. da Silva, Alberto H. F. Laender, Marco A. Casanova, *"An approach to maintaining optimized relational representations of entity-relationship schemas",* Conceptual Modeling - ER '96, Lecture Notes in Computer Science Volume 1157, 1996, pp. 292-308.

[11] T. Apaydin, G. Canahuate, H. Ferhatosmanoglu, A.S.Tosun, *"Approximate Encoding for Direct Access and Query Processing over Compressed Bitmaps",*. VLDB 2006: 846-857.

[12] Z. Pawlak. Rough sets: *"Theoretical aspects of reasoning about data",* Kluwer, 1991.

[13] Z. Pawlak, A. Skowron. *"Rudiments of rough sets"*, Information Sciences 177(1): 3-27, 2007

[14] Xin Wang, Shuyi Wang, Pufeng Du, Zhiyong Feng, "*CHex*: *An Efficient RDF Storage and Indexing Scheme for Column-Oriented Databases*", IJMECS, vol.3, no.3, pp.55-61, 2011.

[15] Sanjay Kumar Yadav, Gurmit Singh, Divakar Singh Yadav, *"Mathematical Framework for A Novel Database Replication Algorithm"* , IJMECS, vol.5, no.9,pp.1-10,2013.DOI: 10.5815/ijmecs.2013.09.01

Authors' Profile

M. Phani Krishna Kishore, Professor, Department of Information Technology, Gayatri Vidya Parishad College of Engineering(Autonomous), Madhurawada, Visakhapatnam,530048.

Ch. Leela Rani, Graduate Student, Department of Computer applications, Gayatri Vidya Parishad College of Engineering (Autonomous), Madhurawada, Visakhapatnam, 530048.

P. V. S. S. Aditya, **,**Under Graduate Student, Department of Information Technology, Gayatri Vidya Parishad College of Engineering (Autonomous), Madhurawada, Visakhapatnam, 530048.

How to cite this paper: Phani Krishna Kishore. M, Leelarani Ch., Aditya. P. V. S. S.,"Database Performance Optimization–A Rough Set Approach", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.3, pp.48-53, 2015. DOI: 10.5815/ijitcs.2015.03.07