An Optimization of Bitmap Index Compression Technique in Bulk Data Movement Infrastructure

Manoj Kumar¹, Tarun Kumar Gupta² and Deepak Umrao Sarwe³
¹Department of Mathematics, Graphic Era Hill University Dehradun-248002, India
²Department of Mathematics, Chaman Lal Mahavidyalaya Landaura, Haridwar-247664, India
³Department of mathematics Mumbai University, Mumbai-400098 Maharashtra, India
E-mail: manojkumar@gehu.ac.in

Abstract. An index is an optional structure, associated with a table or clustered table that can sometimes speed up data access. By creating an index on one or more columns of a table, you gain the ability, in some cases, to retrieve a small set of randomly distributed rows from the table. This research aims to free the filled disk space after documents have been erased. We verify the current compression technologies and apply the data pump to the bitmap index compression. Bitmap is more effective for minimal unique value, according to the traditional wisdom index. However, it needs no Bitmap index at high cardinality or low cardinality by data pump. In this paper we recommend a pump function for space release in the database after documentation has been removed. This utility does not free disk space even though the elimination of records from the table is seen by the Bitmap Index. To free up the room and adjust the index point position, we have introduced data pump for compression. The data pump is commonly used in oracle database for logical backups. Finally, we study the bitmap index which is widely used for industrial purposes and address unanswered questions for the future.

1. Introduction

Oracle introduces a very powerful feature after the B-Tree index that was bitmap index. Bitmap index consist two-dimensional structure for every row with one column of table. In the bitmap index the distinct value is represented by each column. The 2-dimensional array of bitmap index shows all values in the index is product with number of records in table. Dependency of size of bitmap index is depends upon the number of distinct values, which are normally derived as a cardinality. Bitmap index is suitable for the low degree of cardinality means which column contains minimum number of unique values. example-if a column contains 2,00,000 (two million) unique records the bitmap index would not be suitable for this column. Here b-tree index would be efficient. Bitmap index is useful for minimum number of distinct values because its feature of bitmap index that it would be efficient on the minimum number of distinct values. Otherwise the bitmap index would not improve the query processing rather decrease the performance [1].

1.1 Index Compression

As we already know the compression technique is basically used to compress the size of table or reduce the storage area. Oracle already gives this feature for the table compression as well as for the index
compression. When index is created on particular column of a table then it contain some space, which requires some storage or disk space by this technique we can save our disk space because in industry the company have to purchase the disk space according to the size of the their data and pay according to their disk space. Here compression technique plays a vital role for the maintenance of data. Index compression also save the disk space here we can reduce the size of index up to some extent [2]. Oracle basically developed the index key compression feature from 9i database. Index key compression was as like same as the table and data compression. But to compress the index was little bit complex task as compare to table compression. Index compression applies at the column level where duplicate values are reduced by the compression technique. Index compression technique would be applied according to the requirement; DBA enables this feature according to the need. We can also disable the compression behaviour on index by dropping and recreating index [3].

Requirement for the index compression are:
- It would be better for the compression of index that index should be maintained on the least number of unique value columns.
- The large block size table would be superior for the compression; normally oracle provides 16k block size i.e. ideal because there is least chance of data loss.
- Compression on the less unique key column would be preferred for eliminating the duplicate values.
- Block size 32k would not reliable for the compression because it takes lots of time and creates load on the CPU due to which performance suffers a lot.
- Oracle usually takes maximum time to read 32k block as compare to the 16k block size.
- Index compression key also applicable for bitmap as well as b-tree index.
- Record values are defined by the individual bit value.
- Individual records are denoted by the individual bitmap.

2. Literature Review

2.1. Bitmap Index Required for Compression

Normally for the compression of bitmap index [4] we prefers the minimum distinct values column. Here it is generate the small size of index based tree for low cardinality column. If we create it on the maximum distinct values column then it takes maximum amount of time to grow. There is also one more way to minimize the size of bitmap to minimize the number of bitmaps generated on each column. Normally encoding technique is used to generate the minimum bitmaps on a particular attribute. These encoding techniques are:
- Interval encoding [5]
- Bit sliced encoding [6].

2.1.1 Interval Encoding

As name suggests interval encoding is basically derived in terms of intervals, intervals are defined for each column. If the records value meet with this interval, the bin value is automatically set to ‘1’ otherwise it is ‘0’. The values are arranged in such a way that near intervals can partly cover, in this scenario we have manifold buckets set to ‘1’ for base line value.

In the table 1 the record R1 value 5 describes in first block. Where the records R3 has value 6 which is describe in first and second both blocks. Here two blocks value is set to ‘1’. and for the records record R4 last two blocks set to ‘1’. 
Table 1. Bitmap Index Interval Encoding

| Records | values | Intervals |
|---------|--------|-----------|
|         |        | [5,6] [6,7] [7,8] |
| R1      | 5      | 1 0 0     |
| R2      | 8      | 0 0 1     |
| R3      | 6      | 1 1 0     |
| R4      | 7      | 0 1 1     |
| R5      | 5      | 1 0 0     |
| R6      | 8      | 0 0 1     |
| R7      | 6      | 1 1 0     |

2.1.2 Bit-sliced Encoding

Bit-sliced encoding basically generated using arithmetic operation. For each column the encoding in form of $2^{\leq x}$ where $x$ is denoted the degree of cardinality of the column [8]. Here all the values of column are defined in form $2^0$. If the value is generated ‘1’ then the bit value would be generate $2^0$. Refer table 2.

Figure 2. Bitmap index Bit-sliced Encoding

| Records | values | Bit-Sliced |
|---------|--------|------------|
|         |        | $2^0$ $2^1$ $2^2$ |
| R1      | 1      | 1 0 0     |
| R2      | 2      | 0 1 0     |
| R3      | 3      | 1 1 0     |
| R4      | 2      | 0 1 1     |
| R5      | 4      | 1 0 1     |
| R6      | 1      | 1 0 0     |
| R7      | 4      | 0 1 1     |

2.2 Bitmap Compression

There is also one encoding techniques that are equality encoding techniques but the major drawback of this encoding is non accuracy. Usually the number of column values is combined together and make single bitmap attribute. Interval coding are not sufficient because it is not able to answer queries with specified value of column. Usually bit-sliced encoding is able to minimize the number of bitmaps. But it is not efficiently handle the query processing as most of bitmaps required for accessing, mostly many times [9].

Causing the cost of processing increases. There is best method for the compression of bitmap index is to minimize size of individual bitmaps. For the compression we need to apply the logical operation on them. And before the logical operation we need to decode it. Logical operation performance will not affect by the compression [3][4] algorithm. If logical performance would least suffered then query processing would be fast [10].

2.3 Bitmap Compression Algorithm
There are many compression techniques available which provides the best compression approach and not affect the performance of logical operation [11].

2.3.1 Byte-Aligned Bitmap Code (BBC) Compression

Byte aligned algorithm is basically derived for the bitmap compression, which was dependent on the run length encoding. It was derived by the Antoshenkov and oracle hired this technology for the compression of bitmap index. The idea behind this technology to form a combination of bitmaps and convert into bytes and run length encoding apply on this. If bytes contain [12] all the same bits 1 or 0, then aligned byte is known as Gap-Byte. If bytes contain the mixture value like combination of 1’s or 0’s then it is known as Map-byte. There is one more byte which is known as control-byte, which keeps the record of neighbour bytes. If BBC contain only the combination of 0’s in Gap-bytes then it is known as single sided BBC. And if it is contains the combination of 0’s and 1’s in Gap-bytes then it is known as double sided BBC.

It has the better compression behaviour in comparison to the others. It contains the feature in which compress bitmap allow logical operation directly, there is no requirement before to decompress it. It’s also reduces the CPU cost. Because the logical operation speed on compressed bit is faster as compare to logical operation on uncompressed bit [13].

2.3.2 Word-Aligned Hybrid (WAH) Compression

Word aligned hybrid compression was improved version of byte-aligned code. It basically supports the word aligning rather than byte aligning. In some manner it is same as BBC, it also based on the run length encoding. In the present situation the processor are capable to execute the one word of memory in same time as one byte of memory. In this method [14] we follow the word rather than bytes. Its processing time is minimum as compare to the others and due to this it reduces the CPU cost. And this method mostly used in the oracle database to compress the size of Bitmap index [15]. Refer figure 1.

![Figure 1. Word Aligned Hybrid Compression](image)

For the compression of bitmap index firstly we divided the 31 bits of bitmaps into the groups. Same class neighbour group would be overlapped together. The overlapped word is known as fill word and
those word which cannot be overlap normally refers to unique word with its adjacent word known as literal word. We can compress the duplicate vale but we cannot compress the unique value like literal value would not be compress. Main point here to understand that how we will differentiate between the fill word and the literal word. To acquire the fill word we store the most significant second bit of fill word [16]. We already know that the compression bitmap directly support the logical operation and result of logical operation is stored in different compression bitmap. Word aligned compression are more reliable then the byte aligned bitmap code. Its fast executes the query as compare to others compression technique. Sometimes we find the large compression data due to the large unique data. Normally if we are using the scientific data.

3. Proposed Methodology

3.1. Data Pump

Data pump is used to take the backup of logical entity in the database. Data pump is only process in oracle database which can take backup of logical entities. Sometimes it is used to copy the database from one location to another location with very high speed. If user wants to take their table from one schema to another schema then also user uses this facility. Data pump also includes the export and import facility, where user can export their data and preserves it as a backup for future aspects. For taking backup through data pump we need to create a directory, where oracle logically write on that particular location and physical location also required where log file and dump file will store. Through data pump it will take backup of following entities:
- Database – it will export the full database, it will not include the schema in it only the Meta data as a part of object which are usually in data dictionaries.
- Schemas- it will export the full schema, if user don’t want to export the full database then user can export only their own schema. User can’t export other’s schema without privileges.
- Tables- it will perform data pump of particular table if user want to take the backup of particular table then it will use export table operation. User can export the table in its own schema or in other schema using remaps schema option [17].

This phenomenon is also useful to export/import table from one schema to another schema, this process basically known as the remap schema. Through the remap schema we can take the logical backup of table or different objects in different schema. Basically we can load or unload our data through data pump. It can also load or unload data directly to the path which is given by the user or to the external table. Through this feature our all data would be transfer to the external table. There are two types of method which is used to load/unload data through data pump:
- Direct path- where we have to unload our data to our particular path.
- External table - we create external table where we will transfer our whole data, through this facility we can transfer our data from database to external available table which is exist outside the database.

3.1.1 Data Pump Direct Path Load/Unload

Oracle database provides the direct path load/unload facility through data pump. When user performs export/import operation of data then by default direct path operation executes. There is also restriction of direct path load/unload, when there is column has a long data type. Then only and only direct path
load/unload option would be used [18]. It supports different kind of transformation and data types. Direct path load would not be useful for the following conditions:

- If a LOB column consist the domain index then direct path data pump would not use.
- A table which has multi partition and it consist global index.
- If a table is pre-existing in the database and it has an active trigger.
- If table is exist in a group.
- If a BFILE column exist in a table.
- If there is primary key/foreign key relationship in pre-existing table in database.
- If encrypted data column present in the table.
- An embedded VARRAY data column exist in a table.

3.1.2 External Table

External table method enables the reading of data outside the database and writing data from database to the external table outside to the database. Means through the external table we can transfer the data from non-oracle environment to the oracle environment. this is the best option to load all the data in particular table and transfer into oracle environment. Through external table we can speed up over loading/unloading capacity of objects.

- Normally we use the external for loading and unloading of large size table or large amount of data.
- For encrypted data column we basically prefer the external table method.
- If there is active trigger in the table then we use external table.
- External table is useful when the table content is different during loading and unloading.
- If table consist domain or global index then external table would be useful.

3.1.3 Loading or Unloading of Meta-Data

When use external table to export /import the large size of table. Then along with the table data, Meta data also parallel load unload. For this loading unloading process is distributed in the number of I/O execution process. Here the main problem is the Meta data. Basically Meta data is data about data. It explains here that how many records are there in the table what is size of table. What is the data type of the column? What is the maximum character size of particular column? when user is allotted the task to load unload the objects then user select the external table methods and for this operation user have to set parallel I/O operation till 10g database user can set parallel I/O execution value to 1.

3.2 Features Provided by the Data Pump

Data pump support following features to increase the performance of export and import which are the followings and easy to use.

- Its supports the maximum threads [19] during the operation of data pump. means we can allocate the number of processor for particular jobs and speed up our execution process.
- It has ability to start the data pump job, means if user want to take logical backup of their object then user can simply start the data pump export.
- Normally the data pump export and import are operate with one job at one time but if DBA has more than one client then he can attach more clients with the data pump utility without affecting
the current one. Here current job would not be affected due to the attachments of the multiple job at a time.

- We can perform the export import operation not only with the system available data but also the data available in the remote location through the network link.
- We are importing a particular object to different destination then we can also change the name of object and import it with the different name using REMAP_DATAFILE.
- It also supports the filter process in which we can filter a particular objects like we don’t want to include the index during the export operation on table. Then we can use EXCLUDE key word to exclude the index created on table. But if we want to unload the index and the table data as well then we will use the INCLUDE key word.
- Oracle database basically supports the OEM (oracle enterprise manager). This is basically a graphical operation utility from where we can easily monitoring our job performance.
- It also support command line interface which makes it more fast. because command line operations are more fast then OEM.
- It provides the data movement from the version of the database is only possible over the network. It also support the different version data movement.
- For data pump operation worker should have the privilege on particular object and particular schema. To take full advantage of the data pump the user should have authority on particular objects like table and index.
- Here master table is used to track the transferred data and Meta data [20].

3.3 Creation of Table

For performing practical related to the index. User has created a table of name result_final, it contains three column that is empno that is employee identity ,empname that is employee name and last is salary that basically represents the employee salary. These three-column data type’s basically are varchar (10), number (5). Refer figure 2.

![Figure 2. Creation of Table](image)

3.3.1 Data Types Varchar2 (30)

Varchar2 () data type supports character information. It is used for dynamic memory allocation. It is slightly different from the char data type. It will occupy the space of that segment which contains the data. Like if user has given the data type CHAR (30). Then the complete space 30 bytes would be
allocate for all the records like if data is available on 17 bytes then also it will occupy the 30 bytes.
And in the Varchar2(30) then it will allocate the size according to the requirement, if one records contains 15 bytes then only 15 bytes would be allocate for data from the available 30 bytes. Hence it is clear that Varchar2 (30) is used for dynamic memory allocation.

3.3.2 Number Data Type

Number data type support the number types of value. Like salary, employee_id, department_id, phone number.

3.4 Insertion in Table using PL/SQL Query

For creation of large table user mostly prefers the PL/SQL query, Normally user can also insert the random records in the table by the Cartesian products, but correct procedure to insert the millions of record in the table is PL/SQL method .here use the For loop for setting the parameter for lowest value to highest value. Here lowest value would be 1 and maximum value would be the total number of random records. Refer figure 3.

Step 1: Firstly use keyword begin to initiate the PL/SQL loop.
Step 2: Then set limit of for loop, means how many records user want to insert in the table.
Step 3: In this step use the DML (Data Manipulation Language) statement insert to insert the seventy millions records in the table result_final.
Step 4: Values keyword is used to assign the value in the three different columns here is simply insert value the normal count from 1 to 7000000 in empno column.
Step 5: The dbms_random.string (‘U’,30) is function which is used to generate the random upper case string of 30 character. This function basically calls a package which is normally used to generate the random character values in the database.
Step 6: The dbms_random.value (1000,7000) is also a function which is used to generate the random values between the limit assign to the parameter. Here it will generate the value of salary from the 1000 to 7000.
Step 7: Now use TCL (transaction control statement) COMMIT for the permanent insertion of records in the table.
Step 8: Finally ends the loop by end keyword.

```
SQL> /* Insert Seventy Millions Records in result_final Table*/
SQL> Begin
2   for i in 1..7000000
3     Loop
4     insert into result_final
5       values(i, dbms_random.string(‘U’,30), dbms_random.value(1000,7000));
6     if mod(i, 10000) = 0 then
7       Commit;
8     end if;
9     end loop;
10   end;
```

Figure 3. Insertion through PL/SQL in Table
3.5 Query execution without Bitmap index

When execute query without index then oracle performs the complete table scan and query performance suffers due to this. Firstly understand the what it mean of query performance suffer [21]. it basically derives in terms of time require for execution of query normally known as elapsed time now we see that how bitmap index are created on the table and what is the syntax to create the bitmap index. Refer figure 4.

Here user has created a table RESULT_FINAL in which user has inserted a 70 million records using PL/SQL query. Table basically consist the three types of column first is employee_id of number data types second is employee_name of varchar2 (30) data type and third is salary of number (10) data type. Here we are calculating the elapsed time of select query because select query requires full table scan. So it would be beneficial for measuring the query performance here we execute the select query to count the number of distinct value of column empno so oracle execute this query by full table scan because index is not created on this table and for calculating the elapsed time we execute a command i.e. set timing on so now it will show the elapsed time of every query [22].

4. Experiments & Results

4.1 Compression of Data

So compression for this process is more efficient. It decomposes the same values. Normally in the compression group of five same values define by the single value. In oracle 10g database we can compress the bitmap index up to 20 % and consume twenty percent disk space. Th details available in figure 5.
4.1.1 Export the Table for Compression

For the release the space which is still occupied after deletion of records. We will export table result_final and then delete the table from original location and then import it from the dump file. Refer figure 6.

4.1.2 Import the Table for Compression

After deletion of table we will import table from export. And now check the size of table test_final. How much space is released after import. Refer figure 7 table 3.
Figure 7. Import table result_final

Table 3. Commands for Import Export

| Operations   | Syntax                                                                 |
|--------------|------------------------------------------------------------------------|
| Export       | `oracle@sam ~$ expdp TOM/directory=dpdir234 dumpfile=tom.dmp table=result_final.` |
| Import       | `oracle@sam ~$ impdp TOM/directory=dpdir234 dumpfile=tom.dmp table=result_final.` |

Above utility shows that import/export operation only possible of his own schema. Here TOM is user which can export result_final table of its own schema. Oracle provides Remap facility to transfer table result_final from schema TOM to other schema through remap schema user can transmit [23] the objects of database.

4.1.3 Size of Table and Index after Export/Import

Now space would be release after export/import now the size would be minimized. Refer figure 8.
Here we can easily find out the initially size of table was 314 MB after export/import the size is reduced to 256 MB and initially size of index was 174 MB and now size is reduced to 144 MB. Refer figure 9.

Figure 9. Graphical Representation of Index Size

4.2 Comparison Between Elapsed Time of Different Result

Step 1: Elapsed time before bitmap index creation

In this operation we are counting the number of distinct records in the table result_final .there is seven millions records in table and the elapsed time require executing query without bitmap index. **Elapsed time is 34 seconds and 15 Milliseconds.** Refer figure 10.

Figure 10. Elapsed Time Before Bitmap Index Creation

Step 2: Elapsed time after bitmap index creation

In this process firstly we will create bitmap index on table result_final .and now check the elapsed time to execute query. **Elapsed time is 16 Seconds and 19 Milliseconds.** Refer figure 11.
Step 3: Elapsed time after bitmap index Compression

In this process we will check the elapsed time required for counting records after deletion of one million rows. Compression releases the space of table and index which is occupied after deletion.

Elapsed time is 11 Seconds and 53 Milliseconds. Refer figure 12.

According to the above results we can easy conclude that the elapsed time of execution query is decreasing gradually after creation of bitmap index and applying the compression technique on table. When we executed query without creation of bitmap index on table then we find elapsed time 34 second approx. and when we create bitmap index on table then elapsed time was 15 second. After applying compression technique or release space through data pump then found elapsed time 11 seconds approx. so we can easily conclude the execution time is reducing due to creation of bitmap index and compression technique. Here it is completely visible that query execution time is fifty percent reduced after creation of bitmap index. Now we can identified the how index is efficient for the query processing. Oracle doesn’t search all the records in case of index. Only scan some records of database.

5. Conclusion and Future Scope

In this paper, we present the experiment evaluation of Bitmap Index Compression and release occupied disk space of database objects like table and indexes after deletion of records. Industrial database frequently allows the bulk data insertion and deletion. In database deletion of millions records from the table doesn't release occupied disk space immediately. Next steps in our research will be to release the disk space along
with the deletion of records. Oracle database comprises the behaviour to occupy the disk space due segment related issue. Because once segment grow for available records. To utilize this disk space for useful records data rather than wasting it. This work used the data pump utility for the Compression of Bitmap index and table. Data pump utility performed for the logical backups in database. Limitations of this compression technique through data pump that we need to drop the table for release the disk space. The Experimental results, it is observed that in Compression of bitmap index or release the disk space is possible through the data pump utility. It will not release the space even after deletion of records. This is of special interest for the case to compress the bitmap index and table space along with the DML'S (Data Manipulation Language).

6. References

[1] Samy Chambi, Daniel Lemire, Owen Kaser, and Robert Godin. 2016. Better bitmap performance with roaring bitmaps. *Software: practice and experience* 46, 5 (2016), 709–719.
[2] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113.
[3] Giuseppe De Candia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vo-gels. 2007. Dynamo: amazon's highly available keyvalue store. In *ACM SIGOPS Operating Systems review*, Vol. 41. ACM, 205–220.
[4] François Deliège and Torben Bach Pedersen. 2010. Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. In *Proceedings of the 13th International conference on Extending Database Technology*. ACM, 228–239.
[5] Daniel Lemire, Gregory Ssi-Yan-Kai, and Owen Kaser. 2016. Consistently faster and small-er compressed bitmaps with roaring. *Software: Practice and Experience* 46, 11 (2016), 1547–1569.
[6] Peng Lu, Sai Wu, Lidan Shou, and Kian-Lee Tan. 2013. An efficient and compact indexing scheme for large-scale data store. In *Data Engineering (ICDE)*, 2013 *IEEE 29th International Conference on*. IEEE, 326–337.
[7] Memcached. 2018. memcached - A distributed memory object caching system. (2018). Retrieved 2018-10-20 from http://www.memcached.org/
[8] Apache ORC. 2018. Apache ORC, High-Performance Columnar Storage for Hadoop. (2018). Retrieved 2018-10-18 from https://orc.apache.org/
[9] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. 2010. The ha-doop distributed file system. In *Mass storage systems and technologies (MSST)*, 2010 *IEEE 26th symposium on*. IEEE, 1–10.
[10] Lefteris Sidirourgos and Martin Kersten. 2013. Column imprints: a secondary index struct-ure. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 893–904.
[11] Larson P-A, Clinciu C, Hanson EN, et al. SQL server column store indexes. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*; 2011; Athens, Greece.
[12] Idreos S, Groffen F, Nes N, Manegold S, Mullender S, Kersten M. MonetDB: two decades of research in column-oriented database architectures. *IEEE Data Engineering Bulletin*. 2012; 35(1):40-45.
[13] Aktas MS, Plale B, Leake D, Mukhi NK. Unmanaged workflows: their provenance and use. In: *Data Provenance and Data Management in eScience*. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2013:59-81.
[14] Weahama W. Strategies to improve query processing time in searching membership queries of virtual classroom by using DBIC. *Int J Comput Internet Manag.* 2012; 20(1):51-56. 47. 
Crainiceanu A, Lemire D. Bloofi: multidimensional bloom filters. *Inf Syst.* 2015; 54:311-324.

[15] Lin K-W, Byna S, Chou J, Wu K. Optimizing fastquery performance on lustre file system. In: *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*; 2013; Baltimore, MD.

[16] Mr. Ankit Kumar, Dr. Dinesh Goyal, Mr. Pankaj Dadheech, (2018), “A Novel Framework for Performance Optimization of Routing Protocol in VANET Network”, Journal of Advanced Research in Dynamical & Control Systems, Vol. 10, 02-Special Issue, 2018, pp-2110-2121, ISSN: 1943-023X.

[17] Mr. Pankaj Dadheech, Dr. Dinesh Goyal, Dr. Sumit Srivastava, Mr. Ankit Kumar, (2018), “A Scalable Data Processing Using Hadoop & MapReduce for Big Data”, Journal of Advanced Research in Dynamical & Control Systems, Vol. 10, 02-Special Issue, 2018, pp-2099-2109, ISSN: 1943-023X.

[18] Vengatesan, K. et al. "Analysis of Mirai Botnet Malware Issues and Its Prediction Methods in Internet of Things". Lecture Notes on Data Engineering and Communications Technologies 31. (2020): 120-126.

[19] Vimal, V. et al. "Artificial intelligence-based novel scheme for location area planning in cellular networks". Computational Intelligence. (2020).

[20] Kumar, A. et al. "Black hole attack detection in vehicular ad-hoc network using secure AODV routing algorithm". Microprocessors and Microsystems. (2020).

[21] Kumar, A. et al. "Comparative Analysis of Data Mining Techniques to Predict Heart Disease for Diabetic Patients". Communications in Computer and Information Science 1244 CCIS. (2020): 507-518.

[22] Vengatesan, K. et al. "Credit card fraud detection using data analytic techniques". Advances in Mathematics: Scientific Journal 9. 3(2020): 1185-1196.

[23] Preetha, J. et al. "Data mining technique based critical disease prediction in medical field". Advances in Parallel Computing 37. (2020): 104-108.