# Map Reduce based REmoving Dependency on K and Initial Centroid Selection MR-REDIC Algorithm for clustering of Mixed Data

Khyati R. Nirmal[1], K.V.V Satyanarayana[2]
Department of Computer Science and Engineering,
K L Education Foundation,
Vaddeswaram -522502,
Guntur Dist., A.P, India

***Abstract*—In machine learning, clustering is recognized as widely used task to find hidden structure of data. While handling the massive amount of data, the traditional clustering algorithm degrades in performance due to size and mixed type of attributes. The Removal Dependency on K and Initial Centroid Selection (REDIC) algorithm is designed to handle mixed data with frequency based dissimilarity measurement for categorical attributes. The selection of initial centroids and prior decision for number of cluster improves the efficiency of REDIC algorithm. To deal with the large scale data, the REDIC algorithm is migrated to Map Reduce paradigm,and Map Reduce based REDIC( MR-REDIC) algorithm is proposed. The large amount of data is divided into small chunks and parallel approach is used to reduce the execution time of algorithm.The proposed algorithm inherits the feature of REDIC algorithm to cluster the data.The algorithm is implemented in Hadoop environment with three different configuration and evaluated using five bench mark data sets. Experimental results show that the Speed up value of data is gradually shifting towards linear by increasing number of data nodes from one to four. The algorithm also achieves the near to closer value for Scale up parameter, while maintaining the accuracy of algorithm.**

***Keywords*—Machine learning; clustering; similarity measurement; initial centroid selection; number of clusters; map reduce paradigm**

## I. Introduction

In the current era of machine learning the strategy of unsupervised algorithm to group data without prior knowledge is widely adopted in the application of data segregation. Like the applications in the field of web mining, text mining, image processing, stock prediction, signal processing, biology and other fields of science and engineering the algorithms of clustering had been applied for better results. [1] [2]

Clustering targets to find out hidden structure from underlying dataset, without any prior information, here the labels are not associated with data. As an outcome of this the clusters are formed having minimum within cluster distance and maximum between cluster distances. Here one representative of a group is chosen as cluster centroid.

A number of strategies have been proposed in last several years, for solving the clustering problems efficiently[3] The two broad categories of clustering algorithm are: Hierarchical Clustering and Partitional Clustering. In Partitional Clustering K Means Clustering has practiced the facts of simple mathematical formation, ease of implementation and fast coverage[4]. This will enlarge the application field of the algorithm.

Conversely the results generated of K Means Clustering coverage the local optimal based on initial clusters which may or may not be endure from global optimum solution. The Hamming Distance measurement is applicable for numerical dataset only, however the attributes of real world data is not restricted to numerical attributes, it consist of the categorical attributes as well. This is an additional obstacle to adopt the K Means Clustering Algorithm.

For categorical attributes the K Mode Clustering Algorithm has supplanted the Hamming distance measure with simple matching dissimilarity measurement and derived with the alternate solution. To extend the K Mode Clustering algorithm for mixed attributes the K Prototype Clustering algorithm is proposed, which is in cooperation with both the distance based measurement. [5] [6]

The K prototype Clustering algorithm is one of the approaches for clustering of mixed attributes. To enhance the performance of K prototype clustering algorithm, various provisions have been proposed in last decades. The emphasis of this paper is to augment one more approach in the same direction.

The Section 2, the REDIC K Prototype Clustering algorithm along with necessity of migrating the algorithm to MapReduce Paradigm is elaborated. The MapReduce REDIC K prototype Clustering algorithm is proposed In Section 3, the experimental setup and result of the proposed algorithm are given in Section 4.

## II. Background Knowledge

In the Clustering of Mixed data using the K prototype clustering algorithm, the numerical attributes and categorical attributes are separated and functioned separately using two different similarity measurements. It implements Euclidean Distance measurement for numerical attributes and Hamming distance for categorical attributes. The Hamming distance measurement results in 0 if two categorical attributes are similar and results 1 if the results are dissimilar. This reasoning may not give the better result in many real world data set[7].

As the extension of K Prototype Clustering algorithm, The K center Algorithm has been proposed and proved that the algorithm contributes improved results by considering the frequency of attributes in consideration [8].

The effectiveness of this algorithm is contingent on the initial centroids selected, and on the other side the algorithm requires professional knowledge to choose the value for parameter K [9].

REDIC K Prototype Clustering algorithm is proposed in [10], which has precisely concerns the above stated issues, and suggested the alternate strategy.

## *A. REDIC K Prototype Clustering*

The similarity between two categorical attributes is largely depends on the relative frequency of the common values for particular attributes.[11]. For this reason the frequency based method for similarity measurement of categorical attributes must be adopted.REmoval Dependency on K and Initial Centroid Selection (REDIC) K Prototype Clustering algorithm is proposed with a novel frequency based similarity measurement for categorical attributes. [12] Using simple furthest point heuristic (Maxmin) initialization decreases the clustering error of k-means from 15% to 6% on average [13]. This strategy is adopted in REDIC algorithm while choosing the initial centroids. (REDIC) K prototype clustering is proposed, which will have three contribution: 1) Frequency based dissimilarity measurement for the categorical attributes. 2) Select the initial centroids by calculating most significant attributes. 3) Incremental approach for deciding number of clusters. The preliminary used in the algorithm are defined as:

**Preliminary 1.** Similarity between two Categorical Attributes $Cdist(c_i, c_j)$
Consider any two categorical instances $C_i$ and $C_j$ of n instances.

$$Cdist(c_i, c_j) = FOC(c_i) - FOC(c_j) \tag{1}$$

where $FOC(c_i)$ is defined as

$$FOC(c_i) = \frac{no\ of\ occurrence\ of\ c_i}{total\ no\ of\ instances\ n} \tag{2}$$

**Preliminary 2.** Similarity between two Instances $Dist(I_i, I_j)$

$$\begin{aligned} &Dist(I_i,\ I_j) =\ NDist(n_i,\ n_j) + CDist(c_i,\ c_j) \\ &where \\ &NDist =\ Dis\tan ce\ of\ Numerical\ Attributes\ (n_i,\ n_j) \\ &CDist =\ Dis\tan ce\ of\ Categorical\ Attributes\ (c_i,\ c_j) \end{aligned} \tag{3}$$

**Preliminary 3.** Initial Centroid Selection:
The instances having minimum or maximum row factor will be consider as initial cenroids. Here out of $n$ attributes, 1 to $m$ are numerical attributes and $m$ to $n$ are categorical attributes.

$$Row\_Fact(i) = \sum_{a=1}^{m} num_i\ +\ \sum_{a=m}^{n} FOC(i) \tag{4}$$

$$Set\ of\ Initial\ Centroids\ CN =\ \{CN_1, CN_2,\ ...\ ,\ CN_k\} \tag{5}$$

where $k$ = number of instances having minimum or maximum value for Row Factor

**Preliminary 4.** Initial Value for Number of Cluster(k):
The cordiality of set $CN$ is consider as number of initial centroids.

$$k = |CN| \tag{6}$$

**Preliminary 5.** Decision parameter for Cluster Refinement:
Consider the Cluster $C_i$ having cluster centroid $CN_i$ and instances are $I_1, I_2, ..., I_q$

$$\delta_i = min(dist(I_i, C_1), ..., dist(I_i, C_k)) \tag{7}$$

Using these five preliminaries the clusters are refined and formed, here it is observed that the algorithm is computationally simple not much expensive. The evaluation results are also better than the K prototype algorithm.

REDIC Algorithm is designed for the dataset of small size, to deal with the large dataset the substitute option should be formulated. Here the Map Reduce Paradigm is preferred to speed up the REDIC algorithm.

## *B. Map Reduce Paradigm*

To process the large scale data the Map Reduce is designed and processed. [14] Automatic parallelization and task assignment reduce the overhead while deploying the algorithm to the paradigm. Here only two phases are essential to parallelize the algorithm namely: Map and Reduce. For both of the phases the input and output is in the form of $< key, value >$ pair. Match phase accepts the input in $< key, value >$ pair and produces the intermediate list in $< key, value >$ format only. By grouping and shuffling operation this list will be rearranged as per the intermediate key value. The Intermediate list is in the form of $< key, (value1, value2, \ldots, valueN) >$ . Reducer accepts intermediate list as an input and produce the final values according to the algorithm. In the Figure 1, the functions of Map and Red are explained with block of data. The library of MapReduce Paradigm splits the input file in number of blocks. The copies of the input files are stored to the nodes of the paradigm. One superior copy of input files with data and metadata is maintained and referred as the master. The master node does the task allotment automatically without user interference. The rest of nodes are the slave nodes, which performs the task assigned by master node. A slave node who is assigned a map work reads the input file and convert the file into $< key, Value >$ pair. The Mapper function performs the grouping and shuffling and creates the intermediate list. The reducer function reads the intermediate list and performs the sorting operation for mapping of different task to appropriate reducer task. A slave node who is assigned a reducer work iterates over sorted data and assigns the intermediate key to appropriate output values. After successful implementation of algorithm the output is stored into different files of reducer.

## III. Proposed Algorithm: Map Reduce based REDIC K Prototype Clustering

To handle the categorical data of large scale, the Map Reduce based REDIC K Prototype Clustering algorithm is proposed. In MR REDIC Algorithm the distance between two categorical instances are calculated using frequency based method and the initial centers are selected by calculating the row factor of each instance. This will eliminate user

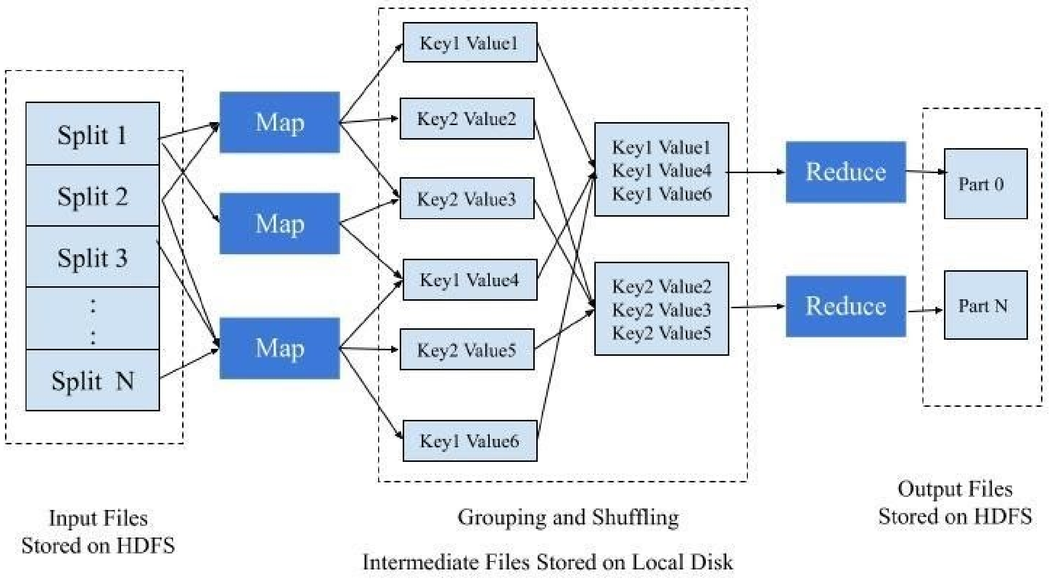


Fig. 1. Map Reduce Paradigm

dependency to choose the number of cluster priory and also improve the result in comparison with random centroid selection method. As shown in the figure 2, MR REDIC algorithm works in two Phases:

The map phase accepts the part of mixed dataset which is stored on HDFS as an input and calculates the row factor for each instance. The instance having minimum or maximum value for row factor is selected as initial centroids. This decides the number of clusters. The distance between each centroid to each instance is calculated and the instance is assigned to the cluster which have minimum distance . Here Cluster Index work as a key and the associated instances of that cluster work as a value of that key.

In reducer phase cluster refinement is done. Here the delta value is calculated for each cluster and it is compared with the distance between cluster centroid and each instance , If this value is less then only particular instance will remain in that cluster else it will be considered non promising instance. Such instances are eliminated from cluster and new cluster with that centroid will be created. This process will be continued till all the instances will be allotted to appropriate cluster and there will be no revision in cluster refinement. Here again Cluster Index work as a key and the associated instances of that cluster work as a value of that key.

The order to migrate REDIC algorithm to Map Reduce paradigm, MR REDIC algorithm is proposed. The Job class does the initialization of the job along with the directory path for Input and Output . The input dataset is stored on HDFS, which will be split and assigned to Mapper Class for further processing. The Job class of MR-REDIC constructs a global variant centers which is a null array( C) , later it will store the information about centers of the clusters. The task is distributed to the datanodes by job class using inbuilt libraries, and reducer class will refine the clusters and update the value for number of cluster parameter.

The Global Center array, offset key and sample value is

```
Algorithm 1: Job Class
Input: I:Input path, O: Output path,W:Intermediate path
Output: a set of k Clusters with allotted instances,
        Execution Time
Method:
Input Path I;
Output Path O;
Create Job ;
Create Cluster Center Array C= $\emptyset$ ;
Set Mapper class;
Set Reducer Class ;
globalF = false ;
while globalF == True do
    Update Input path = W ;
    Start Job
end
data from W to O ;
return value of k and execution time
```

assigned as an input to Mapper Class. Initially when this array is empty the Row value $\varrho$ is calculated by using the method proposed in [12] and stored in the set R. The smallest and largest value from the R is extracted in $\varrho_{min}$ and $\varrho_{max}$. The initial value for number of cluster k is set to the count of min and max value. The initial centroids are stored in $CN_i$ variable, in next step distance from every instance to every centroid $dist(I_i, C_k)$ is calculated by using formula defined in [12] . The instance is assigned to the cluster having minimum cluster centroid distance. The intermediate $< Key, Value >$ pair is maintained where Key is the Cluster Index and Value is the value all the attributes of of particular instance.

The Reducer class will refine the clusters by reallocating the

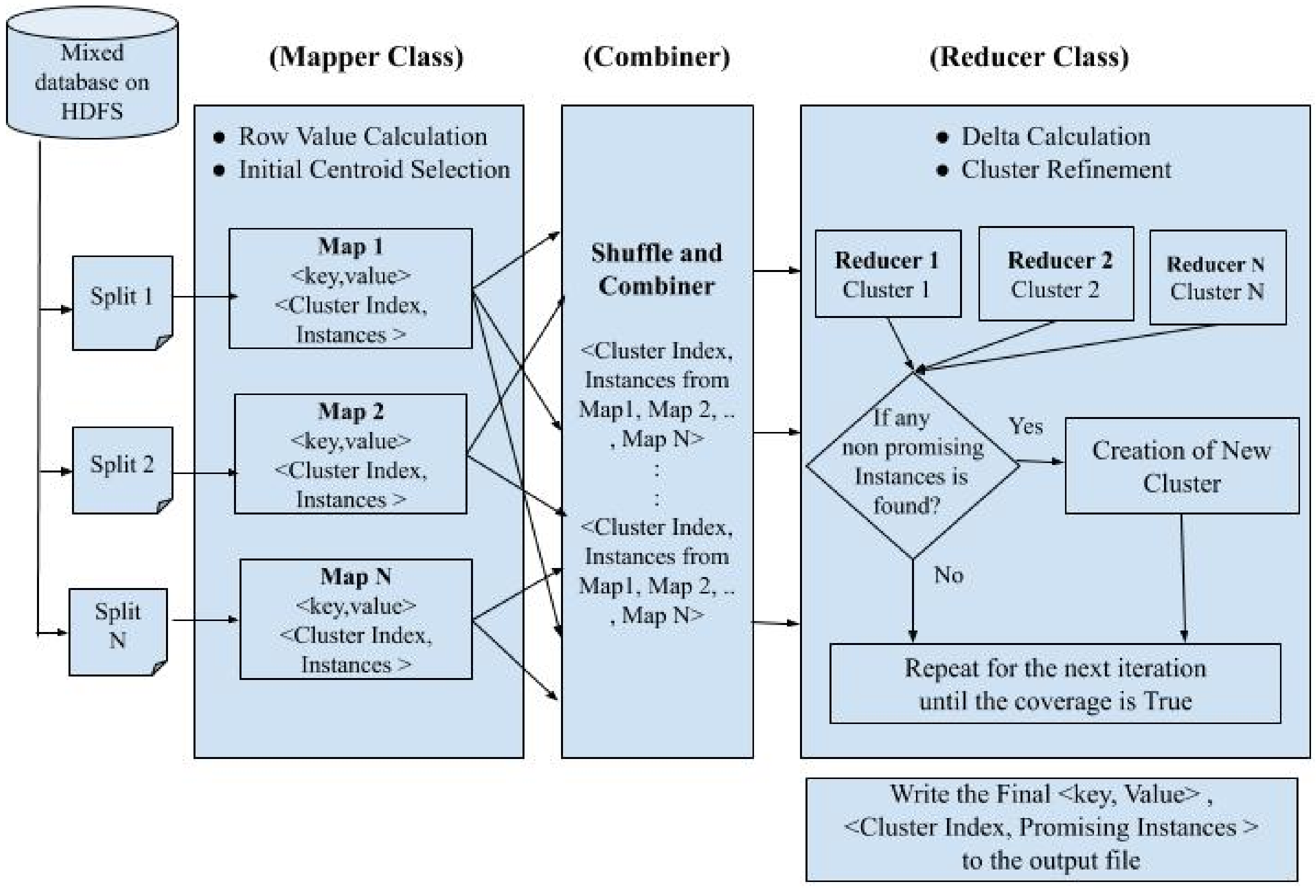

Mixed database on HDFS
(Mapper Class)
(Combiner)
(Reducer Class)
Row Value Calculation
Initial Centroid Selection
Split 1
Split 2
Split N
Map 1 <key,value> <Cluster Index, Instances >
Map 2 <key,value> <Cluster Index, Instances >
Map N <key,value> <Cluster Index, Instances >
Shuffle and Combiner
<Cluster Index, Instances from Map1, Map 2, .. , Map N>
<Cluster Index, Instances from Map1, Map 2, .. , Map N>
Delta Calculation
Cluster Refinement
Reducer 1 Cluster 1
Reducer 2 Cluster 2
Reducer N Cluster N
If any non promising Instances is found?
Yes
No
Creation of New Cluster
Repeat for the next iteration until the coverage is True
Write the Final <key, Value> , <Cluster Index, Promising Instances > to the output file

Fig. 2. Map Reduce Paradigm for MR-REDIC Algorithm

instances to appropriate clusters, and it also creates the new cluster and update the value of K also. The intermediate list produced by Mapper Class is assigned as an input to Reducer Class. For refinement of clusters the decision value parameter $\delta_k$ is calculated using the formula proposed in [12]. The values of $dist(I_i, C_k)$ will compare with $\delta_k$. If $dist(I_i, C_k) > \delta_k$ then that particular instance $I_i$ is not promising for cluster $C_k$ . The instance $I_i$ will be removed from the cluster $C_k$ and the new cluster will be formed and centroid value of both the cluster will be updated. The final $< Key, Value >$ pair is maintained where Key is the updated Cluster Index and Value is the value all the attributes of of particular instance. Finally the results file of the Reducer Class will be stored to the Output Directory path set by Job Class.

## IV. Result and Analysis

The MR-REDIC is implemented in Hadoop architecture with a single node and multi node cluster environment. The dataset used for validation are varies in size to measure out the execution time and speed up of the proposed algorithm. This section divides into three parts:The experimental setup and configuration of nodes, the details of the considered dataset, and the experimental result.

### A. Experiment Setup

For implementation of the proposed algorithm using Map Reduce paradigm, Hadoop 2.6.0 and Java version 1.7.0 is considered. Operating system used is Ubuntu 14.04.The experiment was carried out on different a Hadoop cluster.

| | |
|---|---|
| **Configuration 1:** | Name node and Data Node on a single machine |
| **Configuration 2:** | One Name Node and Two Data Node |
| **Configuration 3:** | One Name node and Four Data Nodes |

The nodes in the Hadoop cluster are configured with Intel $i3\alpha3.64GHZ$ processor, $2GB$of RAM for each node and 500GB of hard disk with a measured bandwidth for end-to-end TCP sockets of $100MB/s$.

### B. Dataset Description

The dataset used in this experiment are downloaded from the kaggle and UCI Repository. To evaluate the performance for the execution time the dataset that of different size and mixed attributes are chosen.
**Poker Hand data set:**This data set is having multivariate characteristics with 1025010 instances and 11 numbers of

**Algorithm 2:** Mapper Class

```
Input: Global variable centers C[ ], the offset key, the
       sample value
Output: <key,value > pair,
        key: The index of the closest center point
        value: he values of all attributes of particular
        Cluster Index
Method:
Construct the sample instance from value;
if $C$ [] is null then
    for $\forall$ row do
        Calculate $\varrho$ ;
        $\varrho_{min} = \min \varrho_1, \varrho_2 ... \varrho_i$ ;
        $\varrho_{max} = \max \varrho_1, \varrho_2 ... \varrho_i$ ;
        $\varrho_{min}, \varrho_{max} \in R$;
    end
    $k = |R|$
end
else
    $CN_i = \{CN_1 , CN_2 ..., CN_p\}$;
    Calculate $dist(I_i, C_k)$;
    Assign instance to the cluster $C_i$ to $CN_i$ having
      minimum cluster distance;
end
key $\rightarrow$ Cluster Index ;
value $\rightarrow$ The values of all attributes of particular Cluster
  Index ;
return <key,value > pair
```

**Algorithm 3:** Reduce Class

```
Input: Key is the index of the cluster, Value is the
       Instance Value
Output: <key,value > pair, where
        key: the index of the cluster
        value: The values of all attributes of new
        Cluster Index
Method:
Calculate $\delta_k$ for Cluster $C_k$ ;
if $dist(I_i, C_k) > \delta_k$ then
    Remove $I_i$ from cluster $C_k$ ;
    k= k+1;
    Create Cluster $C_{k+1}$, where $I_i \in C_{k+1}$ ;
    update $C_n$ set globalF = True;
end
Key $\rightarrow$ cluster Index ;
Value $\rightarrow$ The values of all attributes of new Cluster
  Index ;
return <key,value >
```

attributes. Each instance of dataset is an example of a hand which consist of strategy of how five playing cards are to be drawn from a deck of 52 cards. [15]

**Indian Census dataset:** This dataset consist of 156 attributes of mixed type. It gives the information about population based on demographic data for each district of India.

**Magic Gamma Telescope Dataset** The imaging technique used by telescope, captures the high energy gamma particles, as gamma rays emits radiation by charged particles in an electromagnetic shower. Every event of this radiation is described by the various parameters, like major axis of ellipse [mm] minor axis of ellipse [mm] etc,. out of which 10 , 10-log of sum of content of all pixels etc. Here 10 different numerical attributes are considered. The instances are is divided into two major categories :gammas (signal) and hadrons (background). [16]

**House Sales in King County, USA** [17] This dataset having the information about house sold prices for USA between May 2014 and May 2015. It contains the record of 21614 houses with 21 different properties like, area, longitude, latitude etc.

**Toy Dataset:** This toy dataset comprised of 150000 instance with 6 attributes. Size of the dataset is 5 MB. Here each instance is described by the features like age, gender, income, city etc. [18] The brief introduction of dataset is given in table I

### C. Evaluation Parameters and Results

In these experiments three evaluation measurements are considered: Execution time. Speed up and Scale up .

*1) Execution Time:* The execution time of the proposed algorithm MR REDIC is compared with the REDIC algorithm, here the comparison is done with 3 different configurations. For Configuration 1, Data Node and Name Node is on single machine, for Configuration 2 One Name Node and Two Data Nodes are considered, Further in Configuration 3 One Name Node and 4 Data Nodes are considered. By observing the values of table II, it is verified that the execution time is decreasing gradually for each configuration.
The figure 3 shows the execution time is decreasing as number of data nodes will increase.

*2) Speed up: :* To measure out the Speed up , the number of nodes are increased in every configuration by keeping the fix size of

$$Speedup = \frac{T_1}{T_n}$$

$T_1$ is execution time on 1 node and $T_n$ is execution time for the $n$ node. [19]

The linear Speed up is the ideal case of Map Reduce paradigm. For example , the speed up of algorithm is increasing from 1 to n while splitting the task from 1 machine to n machines. the In real time it is difficult to achieve, as by increasing the number of nodes, the communication overhead will also be considered. To evaluate the Speed up parameter of the proposed algorithm, five different dataset and the 3

TABLE I. DESCRIPTION OF DATASET

| Dataset | Size in MB | No of Instances | Total Attribute | Numerical Attribute | Categorical Attribute |
|---|---|---|---|---|---|
| Poker Hand | 0.61 | 1025010 | 11 | 5 | 6 |
| Indian Census | 1.3 | 1909 | 156 | 151 | 5 |
| Magic Gamma | 1.5 | 19020 | 11 | 11 | 0 |
| House Sales | 2.4 | 21614 | 20 | 19 | 1 |
| Toy | 5 | 150000 | 6 | 3 | 3 |

TABLE II. EXECUTION TIME OF MR-REDIC

| Dataset | Execution Time in seconds | | | |
|---|---|---|---|---|
| | | Proposed Algorithm MR REDIC | | |
| | REDIC | Configuration 1 | Configuration 2 | Configuration 3 |
| Poker Hand | 5.23 | 2.45 | 1.18 | 0.58 |
| Indian Census | 6.85 | 2.32 | 1.19 | 1.02 |
| Magic Gamma | 8.00 | 2.40 | 1.19 | 0.39 |
| House Sales | 11.89 | 6.26 | 6.20 | 4.35 |
| Toy | 29.67 | 8.13 | 7.45 | 4.05 |

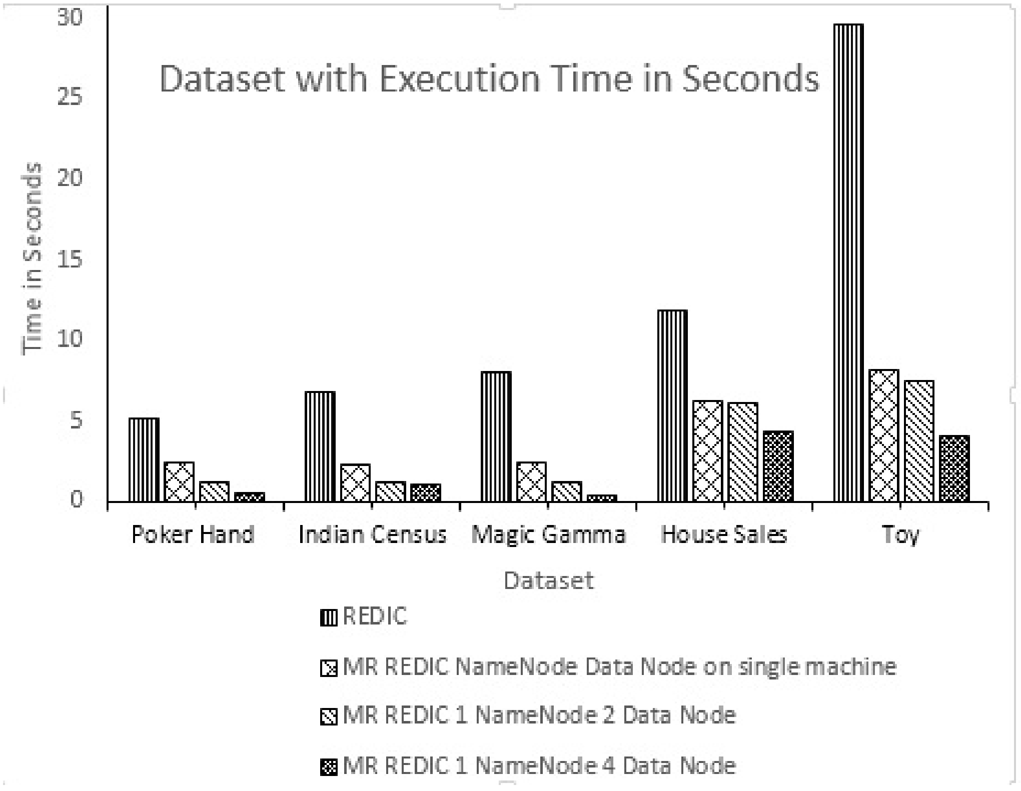


Fig. 3. Comparative Analysis of Execution time

different configuration are considered.From the below figure 4 it is observed the Speed up value is slightly drifting from linear in case of 1 Node to 2 Node, but while considering the Speed up from 2 nodes to 4 nodes it almost closer to linear.

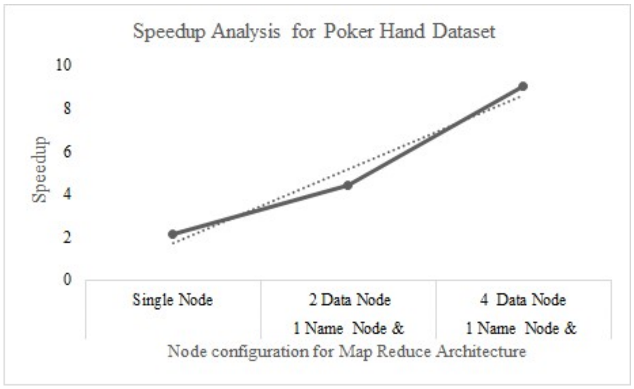
*3) Scale up:* To measure out the Scale up , the size of dataset is increased by keeping the fixed configuration of Data Node and Name Node. It is evaluated by using formula,

$$Scaleup = \frac{TD_n}{T2D_n}$$

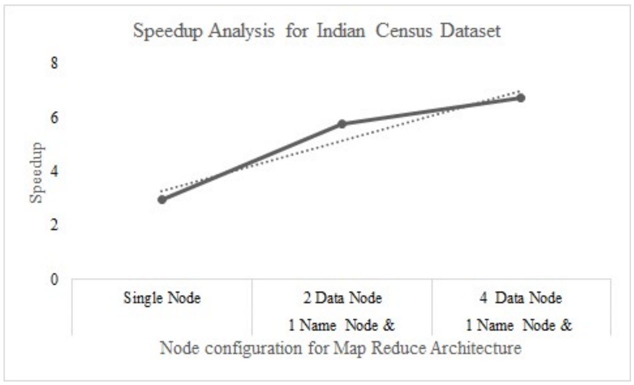
$TD_n$ is execution time of dataset of size D for n Data Nodes
$T2D_n$ is execution time of dataset of size 2D for n Data Nodes [19]

The constant value of Scale up for each size of dataset the ideal case of Map Reduce paradigm.In real time it is difficult to achieve, as execution time depends on the different values of attributes for particular instances.

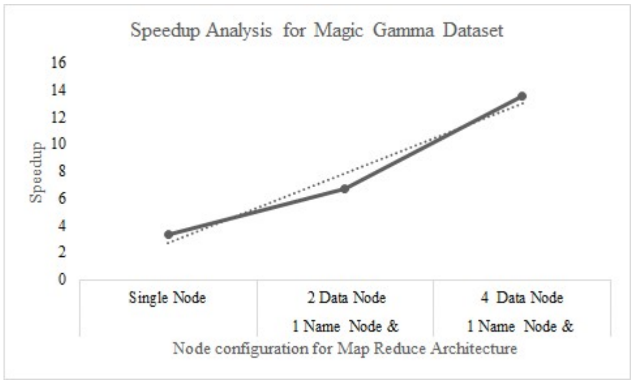
The scale up analysis is calculated in table IV. To evaluate the Scale up parameter of the proposed algorithm, Toy dataset is considered. Here the three instances of dataset with 1MB,2MB, 4 MB sizes are considered. The execution time for different instances of dataset is recorded in table III. From



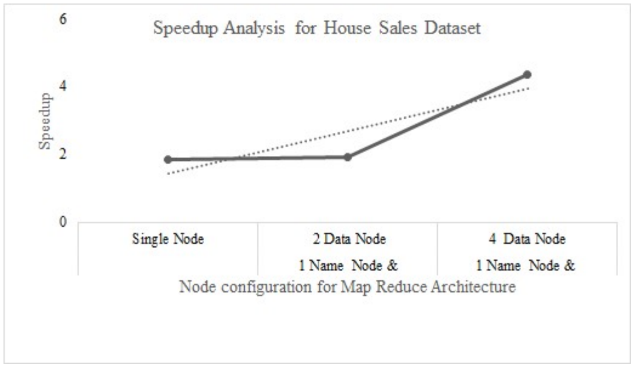
(a) Poker Dataset



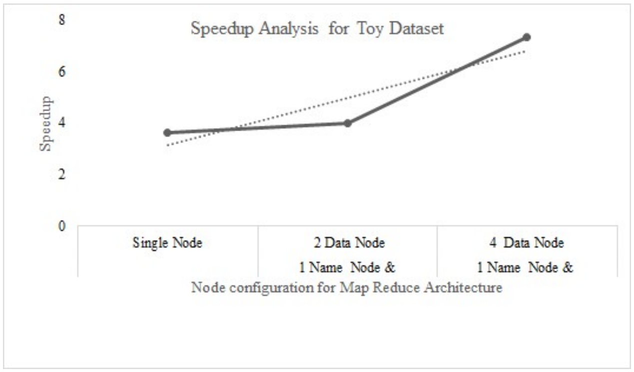
(b) Indian Census Data



(c) Magic Gamma Data set



(d) House Sales Data set



(e) Toy Data set

Fig. 4. Speed up Analysis for different dataset

TABLE III. EXECUTION TIME FOR DIFFERENT INSTANCES OF TOY DATASET

| | Execution time in seconds | | |
|---|---|---|---|
| **Instances of Toy DataSet** | **Single Node** | **1 Name Node and 2 Data Node** | **1 Name Node and 4 Data Node** |
| 1 MB | 1.61 | 1.39 | 1.23 |
| 2 MB | 3.34 | 2.67 | 2.52 |
| 4 MB | 6.62 | 5.43 | 4.98 |

TABLE IV. SCALE UP ANALYSIS OF TOY DATASET

| Measurement for Speed up Parameter | Single Node | 1 Name Node and 2 Data Node | 1 Name Node and 4 Data Node |
|---|---|---|---|
| 1 MB to 2MB | 0.48 | 0.52 | 0.48 |
| 2 M to 4MB | 0.50 | 0.49 | 0.49 |

the table IV, it is observed that In different configurations of Map Reduce architecture with variation in size the value of Scale up parameter ranges from 0.48 to 0.52, which is nearer to constant. The graph for the same result is shown in the Figure 4.

## V. Conclusion

In this paper,the MR-REDIC algorithm is proposed, in which the REDIC K-prototype algorithm is migrated to Map Reduce model for making it suitable to create cluster of mixed data having large scale. The REDIC K-Prototype Clustering is efficient due to its simple calculation and eliminating the dependency on prior parameters. It has proved its efficiency for real time mixed data-sets. But it requires more time for large scale data, so here its parallel version using Map Reduce paradigm is proposed. Experimental results were conducted with five benchmark data-sets and three different configuration of Map Reduce paradigm. In Speed up comparative analysis it has shown the near to linear increase approach, and in Scale up comparative analysis the algorithm gives almost constant value.n In future the different similarity measurement for numerical attributes can be integrated with MR REDIC algorithm.

## References

[1] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster Analysis*. John Wiley & Sons, Ltd, Jan. 2011. [Online]. Available: https://doi.org/10.1002/9780470977811

[2] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[3] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[4] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 7, pp. 881–892, 2002.

[5] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data mining and knowledge discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[6] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li, "Automated variable weighting in k-means type clustering," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 657–668, 2005.

[7] A. Ahmad and S. S. Khan, "Survey of state-of-the-art mixed data clustering algorithms," *IEEE Access*, vol. 7, pp. 31 883–31 902, 2019.

[8] W.-D. Zhao, W.-H. Dai, and C.-B. Tang, "K-centers algorithm for clustering mixed type data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2007, pp. 1140–1147.

[9] S. Harous, M. Al Harmoodi, and H. Biri, "A comparative study of clustering algorithms for mixed datasets," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*. IEEE, 2019, pp. 484–488.

[10] K. R. Nirmal and K. Satyanarayana, "Redic k prototype clustering algorithm," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 18, pp. 5027–5036, 2018.

[11] Z. He, S. Deng, and X. Xu, "Improving k-modes algorithm considering frequencies of attribute values in mode," in *International Conference on Computational and Information Science*. Springer, 2005, pp. 157–162.

[12] K. R. Nirmal and K. Satyanarayana, "Redic k-prototype clustering algorithm for mixed data (numerical and categorical data)," *International Journal of Recent Technology and Engineering*, vol. 7, no. 6, pp. 1–6, 2019.

[13] P. Fränti and S. Sieranoja, "How much can k-means be improved by using better initialization and repeats?" *Pattern Recognition*, vol. 93, pp. 95 – 112, 2019.

[14] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[15] "Uci machine learning repository: Poker hand data set," http://archive.ics.uci.edu/ml/datasets/Poker+Hand, (Accessed on 01/12/2019).

[16] "Uci machine learning repository: Magic gamma telescope data set," https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope, (Accessed on 01/12/2019).

[17] "House sales in king county, usa — kaggle," https://www.kaggle.com/harlfoxem/housesalesprediction/data, (Accessed on 01/12/2019).

[18] "Toy dataset — kaggle," https://www.kaggle.com/carlolepelaars/toy-dataset, (Accessed on 01/12/2019).

[19] X. Yue, W. Man, J. Yue, and G. Liu, "Parallel k-medoids++ spatial clustering algorithm based on mapreduce," *arXiv preprint arXiv:1608.06861*, 2016.