A
Dissertation
On

Parallel Meta-heuristic Algorithm for Clustering

Submitted in Partial Fulfillment of the Requirement
For the Award of the Degree of

Master of Technology

in

Computer Science and Engineering

by

Nupur Bansal
University Roll No.:- 2K14/CSE/11

Under the Esteemed Guidance of

Dr. Kapil Sharma
Associate Professor, Computer Science and Engineering Department, DTU

2014-2016

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY

DELHI – 110042, INDIA

ABSTRACT

Data mining involves the extraction of hidden patterns from raw data. Knowledge mining from raw data is data mining. Cluster Analysis is an important part of Data Mining activities. Analyzing the similarity in data helps to find many useful patterns of data to find relevant results. Clustering finds application in many areas such as pattern analysis, decision-making, grouping and machine-learning examples, including document retrieval, data mining, pattern classification and image segmentation.

Today, the amount of data available has increased manifold. The analysis of this large amount of data is a computationally intensive task and requires a lot of execution time. Hadoop MapReduce is a new framework that allows parallelization, fault tolerance, node balancing and data distribution in library. It consists of user-defined map and reduce tasks. Use of MapReduce for clustering has seen a rise recently so that large datasets can be mined easily.

Likewise, in this thesis we present a Parallel Meta-Heuristic Algorithm for clustering large amount of data. A K-Bat algorithm has been developed that uses advantages to two traditional algorithms and gives comparably good results. The proposed method uses the dynamic exploration and exploitation capability of the bat algorithm. It removes its defect of inappropriate timing of exploitation activity. It uses K-means structure to give initial population to the proposed algorithm. The parallel structure of this algorithm is then proposed. After testing it on benchmark datasets, it shows extremely efficient performance. It achieves best fitness results as well as best execution time.

ACKNOWLEDGEMENT

I would like to express deep gratitude to my project supervisor Dr. Kapil Sharma for providing me the opportunity of taking up this project and continuously guiding me throughout the dissertation program. I am extremely obliged to him for the support, advice and encouragement he has provided me without which the project could not have been a success.

I am also grateful to Dr. O.P Verma, HOD, Computer Science and Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University library and staff for providing the right academic resources and environment for this work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents, friends and seniors for constantly encouraging me during the completion of work.

Nupur Bansal
University Roll no: 2K14/CSE/11
M.Tech (Computer Science and Engineering)
Department of Computer Science and Engineering
Delhi Technological University
Delhi – 110042

Computer Science and Engineering Department

DELHI TECHNOLOGICAL UNIVERSITY

DELHI -110042

www.dce.edu

CERTIFICATE

This is to certify that the dissertation entitled "**Parallel Meta-heuristic Algorithm for Clustering**" is a bonafide record of work done by **Nupur Bansal, Roll No.- 2K14/CSE/11** at **Delhi Technological University** for the partial fulfillment of the requirement for the degree of **Master of Technology** in **Computer Science and Engineering**. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

Date: _____

(Dr. Kapil Sharma)
Associate Professor and Project Guide,
Department of Computer Science and Engineering,
Delhi Technological University.

Table of Contents

List of Figures

List of Tables

Chapter One: **INTRODUCTION**

Data Mining is the process of extracting hidden patterns from raw data. Thus, data mining can also be described as mining of knowledge from raw data. The Information Industry makes available huge amounts of data. Unless this data is converted into useful information, it is of no use. Thus it becomes necessary that this huge data is analyzed and some useful information is extracted from it. For the mining of information many steps are also needed to be performed, such as Data Cleaning, Data Transformation, Data Integration, Pattern Evaluation and Data Presentation. After all these processes have been performed, the extracted information would be useful for us in many real life applications.

Clustering or cluster analysis is a part of the vast data mining process. Cluster analysis according to the source [1] is the grouping of a collection of similar patterns into clusters. The properties used to group are represented as points in a multidimensional space or as vectors of measurements. The patterns are grouped together on the basis of similarity with each other, or they are differentiated on the basis of dissimilarity. Thus, the patterns which are clustered together have more similarity to each other and large dissimilarity to those in other clusters. Clustering finds application in many areas such as pattern analysis, decision-making, grouping and machine-learning examples, including document retrieval, data mining, pattern classification and image segmentation.

K-Means clustering is the most popular technique used to cluster data. Even though it was first proposed in 1957, it is till date the most widely used data clustering technique. It is an incremental approach that proceeds by updating cluster centroids at each step. The updation of any cluster centroid is done by taking the average of all points that lie in that cluster. K-Means algorithm is able to give very good results when the dataset is unimodal. But in case of multi-modal data, this algorithm often gets stuck in a local optimum. This happens because of the way in which the initial centroids are assigned

randomly. In the work [2], the authors provide many modifications to the algorithm to overcome this defect.

Latest trend such as this source [3] , suggests a shift of research towards nature-inspired algorithms for clustering purpose. All stochastic algorithms with local search and randomization are now named as 'meta-heuristic'. The randomization in the meta-heuristics explores the search space and generates arbitrary solutions to give a global solution after some iteration. On the other hand the local search part focuses on a specific region to achieve convergence. Few of the nature-inspired metaheuristics existing in clustering are genetic algorithm (GA), simulated annealing (SA), particle swarm optimization (PSO), ant colony optimization (ACO), artificial bee colony (ABC), bat algorithm (BA) and many more. The journal [4] provides an extensive study of nature-inspired algorithms. These meta-heuristics are able to provide improved clustering solutions due to good exploration and exploitation behaviors.

A shift towards nature-inspired algorithms has also exposed the fact that they are very computationally intensive. This is a major drawback of using the meta-heuristics. To overcome this drawback, the parallelization of algorithm has taken a rise. Distributed algorithms have been developed for the existing algorithms so that the computationally intensive workload can be distributed among many processors. This leads to decrease of processing time. Thus the power of these algorithms can be harnessed easily by parallelization of algorithms.

In our research work, we have taken up the objective of proposing an efficient algorithm for data clustering that can not only give good solutions but also be able to scale with increase of input size. To fulfill this objective, we have taken up the Bat Algorithm for optimization. This algorithm was first presented in [5] by X.S. Yang and has shown comparative solutions to optimization problems. For our thesis, we used this algorithm and combined it with the K-means algorithm so that faster convergence and better results

can be obtained, and getting stuck in local minima can be avoided. Thus a new K-Bat algorithm has been proposed. Next this algorithm has been parallelized on Hadoop. MapReduce framework of Hadoop has been used to parallelize the K-Bat algorithm so that large datasets can be faster. Thus, this dissertation presents a "Parallel K-Bat Algorithm for Clustering of Large Datasets".

1.1 Motivation

An elementary problem that often arises in a variety of fields like pattern recognition, machine learning, image processing and statistics is the clustering problem Clustering is an important part of exploratory data mining. Many algorithms exist to overcome this problem. One of them is K-means. But it has short-coming of getting stuck in local optima. To get improved result we have moved to the use of meta-heuristic algorithms. Meta-heuristics provide the advantage of exploration and exploitation in a search space. This leads to better global and local search operation.

In this dissertation, we put emphasis on developing a parallel meta-heuristic algorithm to minimize computational efforts of clustering.

1.2 Research objective

The objectives of this research work are as follows:

- To develop an algorithm for efficient clustering of numerical data.
- To use nature-inspired algorithm as a clustering technique taking motivation from the previous work done in this area.
- To improve the algorithm developed so that it can be applied to low dimensional as well as high dimensional data.
- To develop a parallel implementation of the sequential algorithm so that it can be applied to very large datasets.

- To use Hadoop MapReduce framework to run the parallel algorithm and test its results against time.

1.3 Thesis Organization

We start this dissertation with Chapter 1 that gives the Introduction to the work. Chapter 2 provides the literature survey of past work done. The works of scholars in fields of clustering, meta-heuristics, and parallel implementations have been studied. Chapter 3 gives the research methodology used to reach the resultant algorithm. It emphasizes on the defects of the original K-means and Bat algorithms. Chapter 4 gives the details of the proposed algorithm. In the chapter 5, our proposed work has been evaluated on 5 benchmark datasets. The final Chapter 6 provides the conclusion of our work and the future scope for this work.

Chapter Two: **LITERATURE SURVEY**

Extensive survey has been done on different clustering techniques and many nature-inspired algorithms have also been applied for this. We present a literature review of past work.

The survey [6] presents all data clustering algorithms and showcases their uses in the traveling salesman problem, in different benchmark data sets, and in bioinformatics. Also topics like cluster validation are evaluated.

The journal paper [3] provides a review of many meta-heuristic nature inspired algorithms for clustering. It is observed that the traditional partitional algorithms are simpler computationally but often the solution gets trapped in local minima and so inaccurate results are provided. The nature inspired algorithms use their population to explore the entire search space and ensure the achievement of optimal partition.

The work [7] proposes a clustering technique based on genetic algorithm called GA-clustering. Strings of real numbers which are the chromosomes represent the cluster centers and these centers are updated by using the GA's searching capability.

An efficient clustering technique based on genetic algorithm utilizing the principles of K-Means clustering algorithm is described in the paper [8]. This algorithm exploits the search capability of K-Means along with avoiding its limitation of getting stuck at local optimals. Its superiority has been demonstrated extensively and a real life application has been presented in the classification of pixels of satellite images taken from Mumbai city.

Two new approaches for data clustering using Particle Swarm Optimization have been proposed in [9]. PSO is used to find the centroids of all clusters. A variant is also presented that uses K-means clustering to initialize the swarm and then PSO refines the cluster centers that have been calculated using K-Means.

The paper [10] presents a Particle Swarm Optimization (PSO) algorithm for document clustering. The PSO clustering algorithm utilizes a global search in the search space against local searching done by K-means clustering. For the experiments, PSO, K-means and a hybrid PSO algorithm has been applied on 4 text document datasets. The results show that the hybrid PSO does better clustering than the K-means.

In the article [11], a new algorithm for optimization called Gravitational Search Algorithm (GSA), has been introduced. GSA is based on Newtonian physics and the search agents are taken as isolated masses. Newton's law of Gravity and of mass interactions is utilized. The gravitational force is used to update the solutions of. The GSA provides superior results in most cases and comparable ones in others when compared with PSO.

An efficient algorithm for cluster analysis has been presented in [12]. This new algorithm is based on heuristic search and gravitational search algorithm. The algorithm, called GSA-HS, uses GSA to find a nearly optimal solution for the problem and then the solution is improved by applying a heuristic search algorithm. Comparison is done with K-means and PSO and the new algorithm gives clusters of high quality.

The work [13] presents two algorithms integrated along with the K-Means algorithm in order to cluster data. This study thus validates the efficiency of the hybrids and measures the quality of the clustering results which are produced by these two hybrids. The algorithms which have been used are Firefly Algorithm, Cuckoo Algorithm, Bat Algorithm and the Ant Colony optimization algorithm. It is observed that the Cuckoo and

Bat algorithms achieve much better objective fitness value than the ACO and Firefly algorithms. Overall, the results show that all four meta-heuristic clustering algorithms take less time to execute and achieve higher accuracy in clustering than the basic K-means Algorithm. This observation shows that K-means is enhanced by nature-inspired optimization algorithms and their integration speeds up the time for searching of good centroids.

Another paper [14] introduces a new meta-heuristic algorithm for optimization based on the phenomenon of the black hole. Two significant advantages have been observed for the proposed algorithm. Firstly, it has an easy to implement simple structure. Secondly, no parameter tuning is required. The proposed algorithm has been applied to the clustering problem and it outperforms many other test algorithms.

The research paper [15] presents a survey on the Particle Swarm Optimization (PSO) algorithm and the variants of PSO applied to clustering of high-dimensional data. It has been studied that the modified PSO and its hybrids with other algorithms such as K-means, Genetic Algorithm, Ant Colony Optimization, etc. can be applied to solve the problem of clustering high-dimensional data and also give better results. Implementing such algorithms for clustering of data of high dimensionality, results in better cluster formation, thus leading to better analysis and prediction of data.

In the paper [16], the gravitational search algorithm has been modified and the number of clusters has been determined by automatic clustering using gravitational search algorithm (ACGSA). The proposed approach has used two new concepts which are the dynamic threshold setting and the weighted cluster centroid computation. Experimental results show that the proposed algorithm is able to calculate the correct number of clusters. This new algorithm helps to find the optimal cluster number such that they are well separated as well as more compact.

In journal paper [17] the new black hole optimization method has been analyzed and found to be only a simplification of the Particle Swarm Optimization applied with inertia weight. The paper shows that the performance of this approach is for maximum number of considered problems much inferior to that of PSO with inertia weights.

The journal paper [18] proposes a new classifier which employs Gravitational Search Algorithm to find the best position results. The performance of this new classifier when compared with Artificial Bee Colony, Particle Swarm Optimization and nine other algorithms confirm that it is effective and efficient method for classification.

MapReduce has been applied widely to many clustering algorithms to process very large data. Review of this work is presented below:

In the paper [19] , the authors explore MapReduce for clustering of very large data. They present the Best of both Worlds (BoW) method, that chooses an appropriate strategy by spotting the bottlenecks. It is shown that BoW has many desirable features. This work also reports experiments on real as well as artificial large data sets.

The paper [20] explores a hybrid of hierarchical clustering algorithms with Map-Reduce for grouping of Internet users based on their web logs. Feature selection is done by a co-occurrence based technique and batch updating is done to reduce the IO overhead. This technique decreases the execution time as well as the number of iterations to nearly 1/15.

In [21], Black Hole algorithm has been parallelized and MapReduce algorithm proposed so that as input data size increases, the algorithm scales well. MapReduce Black Hole (MRBH) algorithm uses the characteristics of the BH approach and thus no parameters are to be manually set. Several datasets have been used along with varying number of nodes. Results of the experiments show that as the number of nodes is increased, significant speedup is obtained.

Different research works done on bat algorithm and its improvised versions are as follows:

In the study [22], bat algorithm which is a new meta-heuristic optimization algorithm, has been used to solve optimization tasks with constraints. BA is verified by applying to many benchmark problems and has shown better performance than various other existing algorithms for this task. The paper also analyzes the unique search features of the bat algorithm.

The paper [23] discusses a new method for attribute reduction based on Bat Algorithm and Rough Set Theory. In this work, numerical experiments have been conducted on 13 datasets to check whether the proposed idea can perform well. When compared with other algorithms for attribute reduction the proposed BAAR algorithm has shown superior performance. BAAR uses the strength of existing algorithms for rough set theory and an added feature of the bat algorithm which is inspired by the echolocation behavior of micro bats. The BAAR algorithm has used 14 parameters which is the highest number of parameters compared to other methods used.

Another resource [24] introduces a new algorithm K-Means and Bat Algorithm (KMBA), which identifies the initial centroid of every cluster. The proposed algorithm uses the bat algorithm and thus the echolocation behavior of bats to find the distance between each data object and centroid. This method first calculates the cluster center based on Bat algorithm and then the clusters are formed by using the K-Means algorithm. The results of the proposed algorithm illustrate that it gives better result than the existing K-Means and BA algorithm.

The paper [5] has presented a new Bat Algorithm (BA) for solving problems of engineering optimization. Bat Algorithm has been found to be very efficient as seen by

simulating several benchmark engineering problems. The BA performs better than many existing algorithms for nonlinear constrained tasks. It is more powerful than other existing methods such as Genetic Algorithm and Particle Swarm Optimization.

The paper [25] proposes a bat-inspired algorithm for feature selection as it can be molded to an optimization problem. The bat algorithm has been converted to a binary version so that bats can occupy binary coordinates on the search space. A feature has been encoded as a string of bits. The proposed binary bat algorithm outperforms the other techniques in 3 cases out of 5 datasets, and turns out to be the second best in the rest 2 datasets.

The paper [26] develops a new bat swarm optimization algorithm called chaotic bat algorithm. This algorithm removes the problem of premature convergence of conventional bat. The loudness in the new chaotic BSO is calculated by multiplying a chaotic map function by a linearly decreasing function. The results of this chaotic bat algorithm approve its performance over conventional BA.

Many modifications to the Bat algorithm have already been developed and used in widespread applications. Some of these improvements have been presented in resources such as [27], [28], [29], [30], [31], [32] and [33]. All these modified algorithms have been presented so as to eliminate drawbacks of bat.

Chapter Three: **RESEARCH METHODOLOGY**

3.1 K-Means Algorithm

K-means algorithm is the one of the simplest unsupervised learning algorithm for data clustering problems. It is based on an iterative scheme that employs an incremental approach to find the local solution. It runs for a fixed number of iterations or till a convergence point is reached. The objective is to minimize a clustering error that is taken as the sum of intra-cluster distances for this work.

K-means starts with initially placing the cluster centers at random positions and at each iteration moves the cluster centers towards a better solution. After each input data has been assigned to a particular cluster in an iteration, the cluster centers are modified by assigning to them the average of all input data that lie in that cluster.

Suppose we are given a dataset of 'N' data inputs and we have to divide these N inputs into 'K' clusters. The dataset is represented as $X = \{x_1, \ldots, x_N\}$, $x_N \in R$. We are using the Euclidean distance as the clustering criterion that has to be minimized. The cluster centers are represented as $C = \{c_1, \ldots, c_K\}$ and the error to be minimized is :

$$E = \sum_{i=1}^{N} \sum_{j=1}^{K} I\,(x_i \in C)\, \|x_i - c_j\| \qquad (1)$$

Where I (x) = 1 if x is true and 0 otherwise.

The centroid updation formula for K-means is:

$$C_i = \frac{1}{n_i} \sum_{j=1}^{N} x_j * I(x_j \in C_i) \qquad (2)$$

Each cluster centroid is updated by average of all data in that cluster. n_i is the number of data points in cluster i.

The pseudo code of the K-means algorithm is as follows:

Input: D (dataset), K (number of clusters)
Output: Final clusters

1: Initialize K cluster-centers with random values.
2: while termination condition is not satisfied do
    3: Assign all data instances to the nearest cluster center.
    4: Update cluster centers by calculating mean.
5: end while

The huge popularity of K-means algorithm is owing to its computational simplicity. However, it has certain limitations as listed by [34]:

1. It has high sensitivity to selection of initial cluster centroids.
2. It often gets trapped in some local minima because of its hill climbing strategy.
3. It is sensitive to outliers and noise.
4. It is not suitable for clusters that have different forms and density.
5. It is only applicable for the data collection where the calculation of average is describable (numerical data only).

3.2 Bat Algorithm

Proposed in [5], the Bat algorithm is a novel approach to optimization. It is based on echolocation behaviour of bats through which they search their prey. This capability of microbats of echolocating is quite fascinating as the bats can not only find their prey but also distinguish between different types of insects even in complete darkness.
Microbats are a famous type of bats because they extensively use echolocation to detect prey, locate their roosting crevices and avoid obstacles in the dark. These bats emit very large sound pulse and then listen for echo that comes back from surrounding objects. These emitted pulses vary in characteristics and are related to their hunting strategies, which depends on the species. Their signal bandwidth also varies depending on the

species. Typically the range of frequency for most bats is in range between 25kHz and 100kHz, but some species emit higher frequencies ranging upto 150 kHz. Astonishingly, the pulse emitted by microbats can be as loud as 110 dB. Fortunately, these loud pulses lie in the ultrasonic region. The loudness of pulses also varies with distance from the prey, being quietest when closer to the prey. These short pulses indicate the amazing signal processing capability of the micro bats.

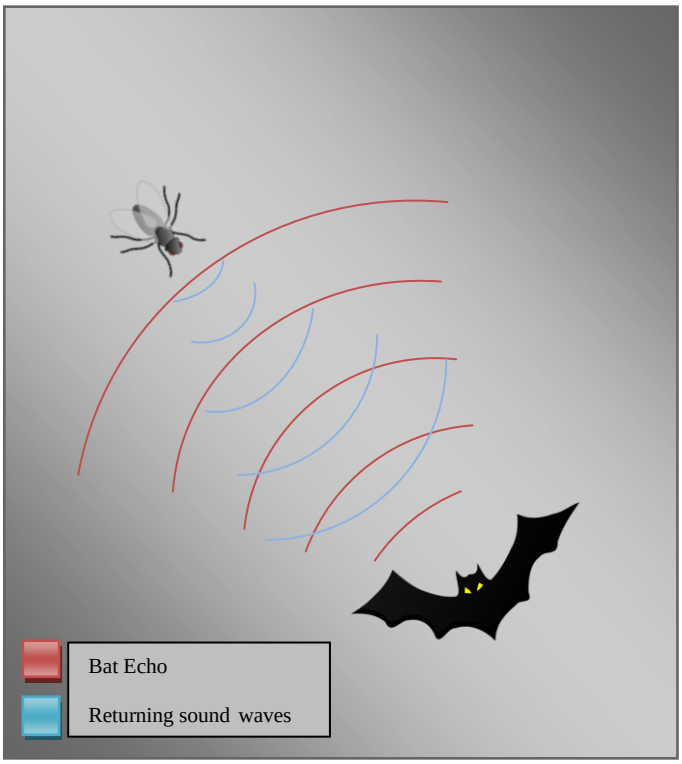

Figure 3-1. Pictorial Representation of Echolocation behaviour of Microbats

Microbats can even manage to avoid obstacles which are as small as a thin human hair. Studies also show that microbats use time delay between the emission and the detection of the echo, time differences between their own two ears, and the loudness variation of the echoes to build up a 3 dimensional scenario of their surrounding. This unique echolocation character of microbats can be used in such a manner that it can be associated with objective function that has to be optimized, and this makes it possible to form new optimization algorithm called the Bat Algorithm (BA).

Following rules are used for the algorithm:

1. All bats will use echolocation to detect distance, and they also will 'know' the difference between food and the background barriers;

2. Bats randomly fly with velocity vel_i at position pos_i with a fixed frequency $freq_{min}$, varying wavelength λ and loudness L_0 in search of prey. They automatically adjust the frequency (or wavelength) of their pulses and also adjust the rate of pulse emission rate $\in [0 , 1]$, depending on proximity of their target;

3. The loudness is varied from a large (positive) L_0 to a constant minimum value L_{min}.

4. Also, we do not have to necessarily use the wavelengths only, instead, we can also fix the wavelength and vary the frequency. For simplicity, we assume $f \in [0 , f_{max}]$ or $f \in [f_{min}, f_{max}]$.

5. The rate of pulse will be taken from the range of [0 , 1] where 0 is same as no pulses at all, and 1 equals the maximum rate of pulse emission.

The position update equations of the bat algorithm are :

$$freq_i = freq_{min} + (freq_{max} - freq_{min}) * rand \quad (3)$$

$$vel_i^t = vel_i^{t-1} + (pos_i^{t-1} - pos^*) * freq_i \quad (4)$$

$$pos_i^t = pos_i^{t-1} + vel_i^t \quad (5)$$

Here, pos^* is the current global solution that has been obtained after comparison of all bat fitness.

The new equation of local search around the best is as follows:

$$pos_{new} = pos^* + \varepsilon * L' \quad (6)$$

Here, ε is a random number in the range [-1, 1]. L' is the average value of the loudness of all bats in the population.

The equations for update of loudness and pulse rate are:

$$L_i^{t+1} = \alpha * L_i^t \quad (7)$$

$$rate_i^{t+1} = rate_0(1 - e^{-\gamma t}) \quad (8)$$

As suggested by the paper that presented the bat algorithm, α and γ are taken as 0.9. t is the number of the iteration that is being run at present. Pseudo code of the bat algorithm is:

1. Initialize the parameters such as α, γ, r_0, population size, max_iterations.
2. Read input file containing the input dataset.
3. Read input file containing the number of clusters, dimension, number of input data and range of attributes.
4. Initialize position, velocity, rate, L for all bats.
5. Find the fittest bat. Assign to best.
6. t=1
7. while(t<=max_iterations)
    a. For each bat_i ϵ population
        i. Update values of f_i, v_i, x_i using equations (3) to (5)
        ii. If Random>=$rate_i$
            a. Local search around the best solution using equation (6)
        iii. Calculate $fitness_i$ of bat_i by putting data into clusters.
        iv. If $fitness_i$ <= best
            a. Update best
            b. Update bat position
            c. Decrease Loudness
            d. Increase Pulse Rate emission
    b. t++
8. Return best
9. End

3.3 Big Data and Hadoop MapReduce

Big data is a catchphrase, or buzzword, which is used for a massive amount of data that can be either structured or unstructured. Big data is so large in size that processing it is difficult and almost impossible using conventional software techniques and databases. Today in maximum ventures the volume of produced data is very big or/and it is moving too fast that it now exceeds the current capacity of processing.

In spite of all these problems, if big data is captured and analyzed properly then it has potential to assist all companies in improving their operations and making their decisions faster and more intelligently. Thus big data is captured, formatted and manipulated, stored and analysed so that using it a company can improve its operation by increasing revenues, getting and retaining customers.

It may seem that the word big data is used mainly in reference to the large volume of data, but that is not the case always. The term big data is often used by vendors to refer to the technology such as tools and the processes that an association requires so that it can handle the big amounts of data and provide its storage facilities. Web search organizations are believed to have coined the term “big data” when they needed to query huge aggregations of distributed loosely-structured data. A big data example might be petabyte (1,024 terabytes) or exabyte (1,024 petabytes) of data which consists of millions of billions to trillions of data records of billions of people—all coming from different sources (for e.g. sales, web, contact centers, mobile data, social media, and so on). The data is mostly loosely or un-structured data that means it is often inaccessible and incomplete.

Today, the amount of data has increased manifold and its processing has become a huge problem. This big data is usually so large that its computations need to be distributed across thousands of machines so that computations can be finished in reasonable time

period. Also there are the issues of parallelizing the computation, and distributing data, and handling of failures which require large as well as complex codes to be dealt with. As a solution to this problem, a new abstraction has been designed that allows simple computations along with hiding the untidy details of fault-tolerance, parallelization, load balancing and data distribution in a library. This abstraction is conceptualized from map and reduce primitives present in Lisp and in other languages. Most of computations involve applying map operation to each "record" in the input. This computes a set of intermediate key and value pairs. Then a reduce operation is applied to all the values that share same key, so that the derived data is combined appropriately. This model of user specified map-reduce operations allow large computations to be parallelized easily and fault tolerance to be handled by re-execution.

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage:

Map stage : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage : This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

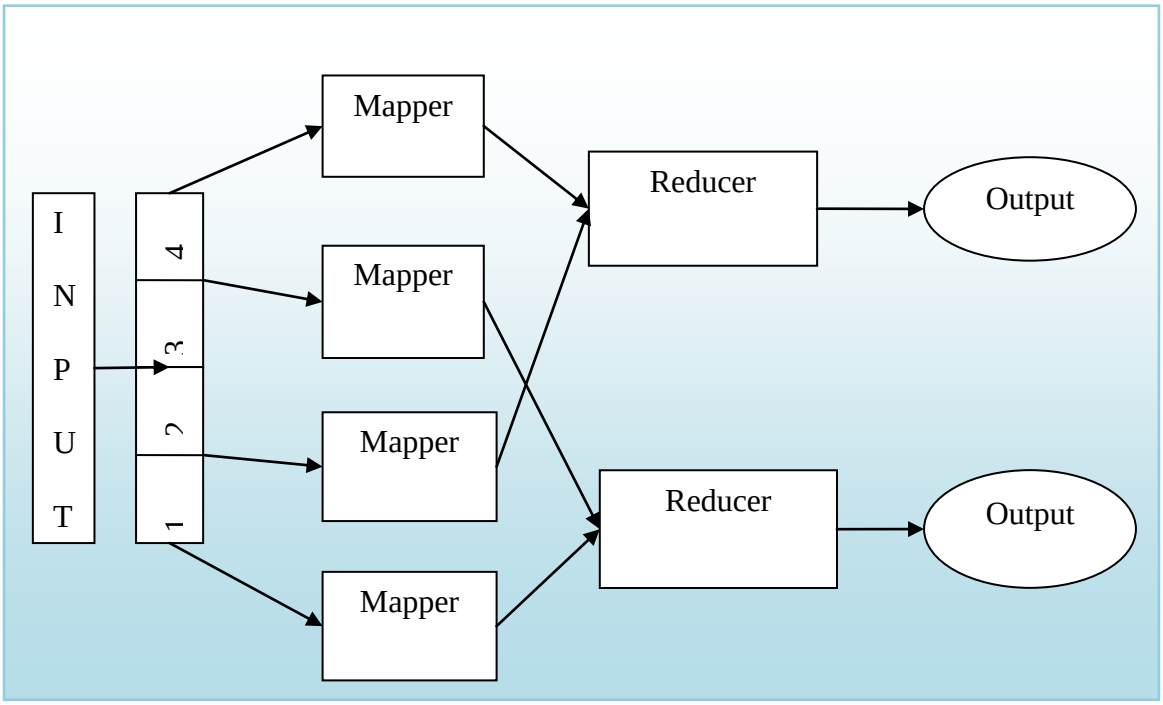

Figure 3-2. MapReduce Architecture

During any MapReduce job, Map and Reduce tasks are sent to the appropriate cluster servers by Hadoop. All the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes are managed by the framework. Network traffic is reduced because most of the computing takes place on nodes with input data on local disks.

After the completion of these tasks, the cluster will collect and then reduce the data to convert to an appropriate result, and then it is sent it back to Hadoop server.

The MapReduce framework operates on (key, value) pairs. Thus, the input to the job is provided as a set of (key, value) pairs and the output is also produced a set of (key, value) pairs. These (key, value) pairs can be of different forms for input and output.

The Writable interface should be implemented so that these key and value classes are available in serialized manner by the framework. Additionally, the reducer performs the sorting and shuffling phase. To facilitate this, the key classes need to implement the Writable-Comparable interface.

Chapter Four: **PROPOSED ALGORITHM**

In this chapter we present the algorithm that has been developed to overcome the defects of the original bat algorithm and solve the problem of data clustering efficiently as the size of input data is increased. Thus a scalable algorithm called the K-Bat has been proposed in this work. This proposed algorithm has been also extended on MapReduce framework of Hadoop to cope with the complexities related to massive datasets.

4.1 Problem Statement

Our objective of the thesis work is to group input data into a fixed number of clusters such that either it has minimum intra-cluster distance or maximum inter-cluster distance. Also, as the size of input data will increase, our algorithm should be able to scale as well. Thus, it should be able to give optimal output in reasonable amount of time.

4.2 Proposed Solution

As a solution to the problem of exploration and exploitation of the bat algorithm, a new algorithm called the K-means Integrated Bat Algorithm has been proposed. This algorithm also called K-Bat algorithm overcomes the problem of Bat algorithm by using K-means to provide an initial set of centers. This results in an effective algorithm that avoids local minima in most of the cases.

4.2.1 K-Bat Algorithm

This algorithm is derived from K-means algorithm and bat algorithm. It starts with taking initial bat population by running K-means a fixed number of times (taken as 5 here) and then performing the bat algorithm. Also certain parameters of the bat algorithm have been updated for better performance.

The position update equations of the algorithm are same as bat algorithm:

$$freq_i = freq_{min} + (freq_{max} - freq_{min}) * rand \quad (3)$$

$$vel_i^t = vel_i^{t-1} + (pos_i^{t-1} - pos^*) * freq_i \quad (4)$$

$$pos_i^t = pos_i^{t-1} + vel_i^t \quad (5)$$

Here, x^* is the current global solution that has been obtained after comparison of all bat fitness.

The new equation of local search around the best is as follows:

$$pos_{new} = pos^* + \varepsilon * 0.001 \quad (9)$$

Here, ε is a random number in the range [-1, 1]. Also, the average of loudness of bats has been skipped and instead the factor 0.001 has been used to multiply to the best solution. This has been done because searching for best solution should be done as close to the best value as possible. This is the base of exploitation.

The equations for update of loudness and pulse rate are:

$$L_i^{t+1} = \alpha * L_i^t \quad (7)$$

$$rate_i^{t+1} = rate_0(1 - e^{-\gamma t}) \quad (8)$$

In these equations, α is taken as 0.9 and γ is taken as 0.001 so that better exploitation can be done. Value of $rate_0$ is taken 0.5 for better exploration.

Compared with the existing meta-heuristics, the bat algorithm has the advantage of dynamic control of exploitation and exploration by performing a local search around the best solution. When a random number generated becomes greater than the pulse rate value, the algorithm switches to exploitation around the best solution. But this structure of exploration and exploitation is not balanced as it should be. At the beginning of the algorithm, first exploitation is done and then exploration but it should be the other way round.

Pseudo-code of the K-Bat algorithm is presented below:

1. Initialize the parameters such as α, γ, $rate_0$, population size, max_iterations.
2. Read input file containing the input dataset.
3. Read input file containing the number of clusters, dimension, number of input data and range of attributes.
4. For each bat in the population do
    a. Run K-means 5 times and assign the centroids obtained to a bat
5. Initialize velocity, rate, L for all bats.
6. Find the fittest bat. Assign to best.
7. t=1
8. while(t<=max_iterations)
    a. For each bat_i ϵ population
        i. Update values of $freq_i$, vel_i, pos_i using equations (3) to (4)
        ii. If Random>=$rate_i$
            a. Local search around the best solution using equation (9)
        iii. Calculate $fitness_i$ of bat_i by putting data into clusters.
        iv. If $fitness_i$ <= best
            a. Update best
            b. Update bat position
            c. Decrease Loudness
            d. Increase Pulse Rate emission
    b. t++
9. Return best
10. End

The K-bat algorithm is presented by the below figure:

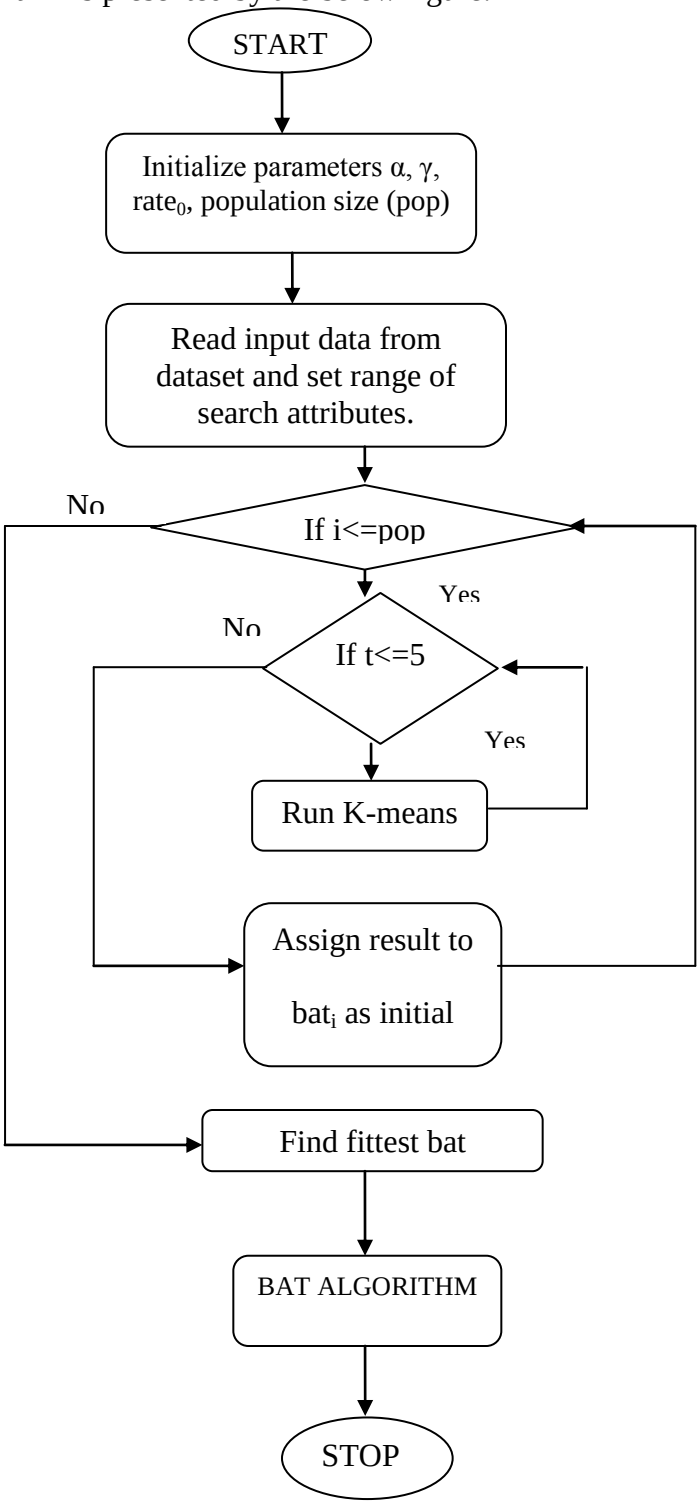

Figure 4-1. Flowchart of K-Bat algorithm

The bat algorithm is presented by the below figure:

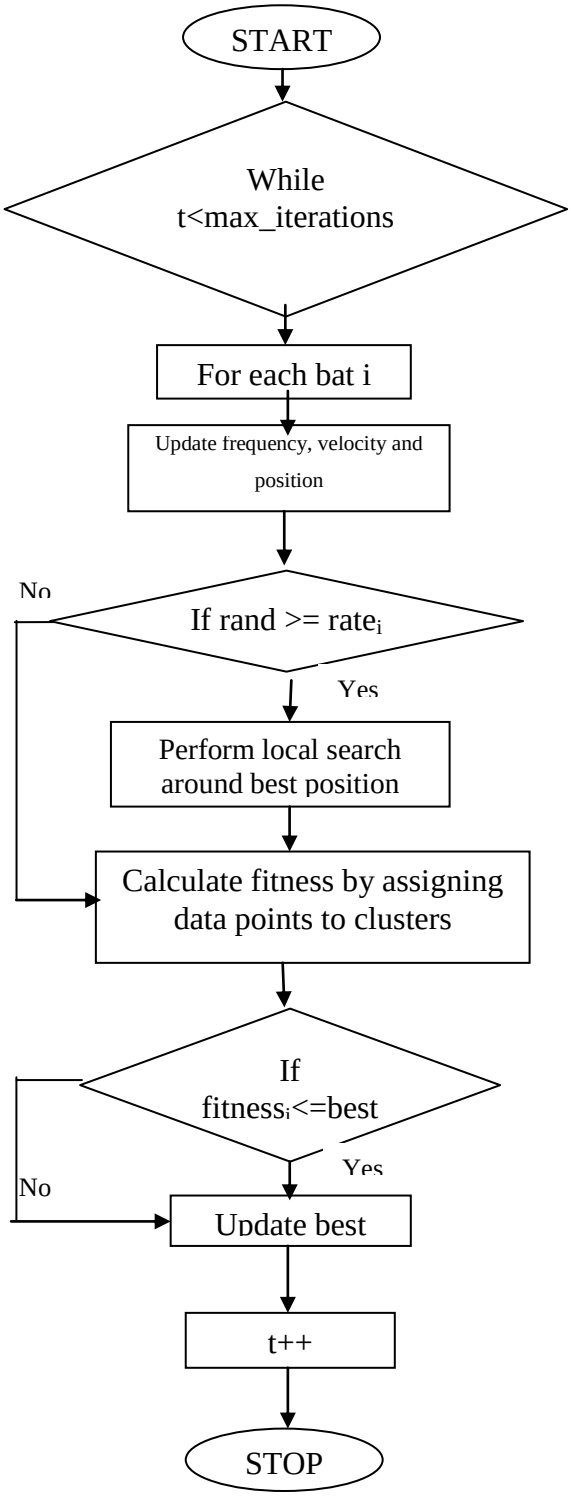

Figure 4-2. Flowchart of Bat Algorithm used in K-Bat

4.2.2 MapReduce Modules

The proposed algorithm has been extended and another algorithm has been provided for implementation on MapReduce. This provides a parallel implementation of K-Bat algorithm. The aim of implementing our algorithm on MapReduce is to improve the capability of the algorithm to mine large-scale datasets and to efficiently update the population. For this purpose, we divide our proposed K-Bat algorithm into 3 different parts:

1. Mapper
2. Reducer
3. Merge Module

We aim to divide our population among the mappers so that large datasets take less time when run on large number of machines. As population is divided among different mappers on different processors, the time used to execute is decreased.
According to our observation, the calculation of fitness is the most computationally intensive part of the algorithm, thus this part is divided among the mappers. This decreases computation time and cost to a large extent.

1. ***Mapper:***

   The job of our mapper is to calculate the minimum distance between a data point and each bat. Thus, for each bat, mapper finds the distance of a data point to each centroid. Then it compares those distances and emits only the minimum distance for one bat and one data point. The mapper reads input that is provided by the user as a text file. This large dataset is split into smaller sub datasets before it is sent to all mappers. This file is read in the form of values one by one; where one value represents one data point of the dataset. After the calculation of the minimum distance of a data point from the bat, the mapper emits the bat ID and the computed minimum distance value as output. The pseudo code of this entire process of Mapper is presented below:

Input: Key- dataID *Output: Key- batID*

Value- data *Value-*

Mindistance

```
Map (Key : dataId, Value : data)

    a. bat_list = read ( file )
    b. For each bat
        a. Mindistance= getMinDistance (bat, data)
        b. write (badID, Mindistance)
```

2. Reducer:

The work of our reducer is to club the outputs obtained from the mapper. The mapper outputs the distances of all points from a bat. To calculate the fitness of a bat, all these distances need to be added. The sum of these values gives the sum of intra-cluster distances which we aim to minimize.

A reducer consists of an automatic step of shuffling and sorting. All the outputs of a mapper are sorted and those which have the same key value are put together as one key value pair. Here, value becomes an iterable comma separated list of all values associated with one key.

In our module, the reducer sorts the batID and the results with same batID are put together. Thus, the input of the reducer is the key: value pair, where key is made of batID and value is made of all data points' distances from that bat. Then the reducer performs the action of summing these values and checking if there is an improvement in the fitness of a bat by comparing it to previous fitness. On improvement of fitness, the bat properties are updated otherwise the changes are discarded. The reducer then outputs the (bat, fitness) as (key, value) pair.

The pseudo code of this reducer module is given below:

Input: Key- batID *Output: Key- bat*

Value- Mindistance *Value- bat.fitness*

```
Reduce (Key: batID, Value: Mindistance)

    a. For each dist in Mindistance list
        a. new_fitness+=dist
    b. if(new_fitness > old_fitness)
        a. update old_fitness=new_fitness
        b. update position vector
        c. decrease loudness
        d. increase pulse rate
    c. writeToFile (new_bat, new_fitness)
    d. write (new_bat, new_fitness)
```

3. Merge Module:

After our mappers and reducers have finished working, the merge module is used to find the best result from the output of the reducer. The merge module takes as input all the fitness values of bats and compares the bats' fitness to find out the bat with the best i.e. minimum fitness value. This value is compared with the global best solution and it is updated. Thus, the merge module merges the solution from all reducer functions to give the best fitness value.

The pseudo code of the merge module is given below:

```
Merge (bat, fitness)

    a. Find minimum fitness
    b. If( minFit < best)
            a. best=minFit
```

These three modules form the base of the MapReduce implementation of the K-Bat algorithm. Apart from these the Driver code drives the entire implementation and connects all the parts of the algorithms together.

The Driver code starts by initializing the parameters of the K-Bat algorithm. Then, it derives the initial centers to be fed to the bat initial population by the K-means algorithm. This is the main step of our improved algorithm. The algorithm loops for a number of iterations. The position, velocity and frequency updation steps are also performed in the Driver module. It outputs the final result of the best fitness value.

Chapter Five: **SIMULATION AND RESULTS**

The K-Bat algorithm is simulated on Hadoop 2.7.2 platform on 5 benchmark datasets. We have also compared proposed algorithm with K-means and Bat algorithms. K-Bat is outperforming all other algorithms not only in time but also in the quality of clustering.

5.1 Simulation Setup

To simulate the three algorithms – Bat algorithm, k-means algorithm and the proposed K-Bat algorithm the following configurations of systems have been used:

1. Intel Corei5-4570 processors with processing 3.20GHz × 4.
2. 32-bit configuration, 500 GB hard disk space
3. Ubuntu 14.04

The Hadoop configurations used to run the MapReduce implementation are:

1. Hadoop version 2.7.2.
2. Replication value of 3.

The source [5] proposed the Bat Algorithm originally. Thus the parameters used for simulation are as defined in the paper. These parameter values are:

Table 1. Bat Parameters.

Parameters	**Values**
Population size (pop)	40
f_{min}	0.0
f_{max}	10.0
alpha	0.9
gamma	0.9
r_0	0.9

The parameters used in the proposed K-Bat algorithm have been updated to get the best results and improve exploration and exploitation ability of the algorithm. The parameters used are as follows:

Table 2. K-Bat Parameters

Parameters	Values
Population size (pop)	40
f_{min}	0.0
f_{max}	10.0
alpha	0.9
gamma	0.01
r_0	0.5

5.2 Datasets

Five benchmark datasets have been used in the implementation and testing of the proposed algorithm. These are Iris, Glass, Wine, Magic, and Pokerhand. All provide a variety in the type of datasets used to test our algorithm because they vary largely in the number of attributes as well as in the number of data values in the dataset. They are described below:

Table 3. Dataset Information

Dataset	Number of attributes	Number of clusters	Number of data values
Iris	4	3	150
Glass	9	7	214
Wine	13	3	178
Magic	10	2	19020

Pokerhand	10	10	1,000,000

1. Iris: This is a flower dataset used to differentiate the variation of Iris flowers of three related species. Each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor) consists of 50 data points and four features.
2. Glass: This dataset consisting of 9 attributes uses these attributes to identify the type of glass used for construction of windows, containers, tableware and headlamps.
3. Wine: A chemical analysis of wines which were grown in Italy but derived from three distinct cultivars. The analysis computed the quantities of 13 components in each of the 3 varieties of wines.
4. Magic: This is the Magic gamma telescope data found in 2004. It is used to distinguish patterns which are belonging to primary gamma from those which are caused by cosmic rays.
5. Pokerhand: Each data point is an instance of a hand made up of five playing cards which have been drawn from a deck of 52 cards. Two things (rank and suit) define each attribute thus forming total of 10 attributes.

5.3 Results

The function that has been minimized is the sum of intra-cluster distances for all clusters. This is taken as the clustering error and thus this value is minimized. This has been performed by K-means algorithm, Bat algorithm and K-Bat algorithm and results obtained have been presented in given table.

Table 4. Results of simulation on K-means, Bat, K-Bat

		K-Means Algorithm	Bat Algorithm	K-Bat Algorithm
Iris	Best	97.3259	105.9406	96.6555
	Average	105.7290	119.1983	96.6555
Glass	Best	215.6775	342.2624	199.1234
	Average	227.9778	380.9874	202.4873
Wine	Best	16,555.6794	16,768.6586	16,396.0349
	Average	16,963.0449	17,094.8874	16,461.384
Magic	Best	1,650,401.6848	2,205,689.8209	1,645,851.7511
	Average	1,660,321.2459	2,635,126.5478	1,647,415.2991
Pokerhand	Best	6,666,863.8264	6,675,069.5800	6,031,523.3012
	Average	6,874,382.9723	6,698,532.9821	6,055,345.8123

These results clearly show that the K-Bat algorithm surpasses both base algorithms in terms of the fitness value. The best values as well as the average values obtained for each dataset by the K-Bat algorithm is better than those obtained by the Bat algorithm and the K-means algorithm.

5.4 Execution Time Graphs

1. ***Iris Dataset:***

   The result of this dataset reaches the best value within 50 iterations of the K-Bat algorithm. Thus 50 iterations have been run and time has been evaluated for Bat and K-Bat algorithms as the number of nodes increase.

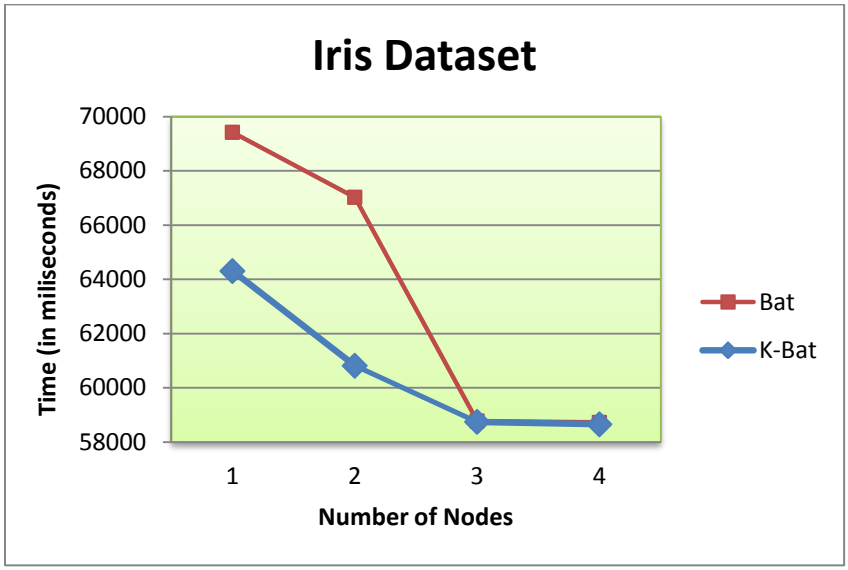

Figure 5-1. Number of Nodes VS Time graph for Iris Dataset

It can be observed from the graph above that since iris is a small dataset, the use of more number of blocks does bring much alteration in the execution times of the algorithms.

2. ***Glass Dataset***:

The result of this dataset reaches the best value within 50 iterations of the K-Bat algorithm. Thus 50 iterations have been run and time has been evaluated for Bat and K-Bat algorithms as the number of nodes increase.

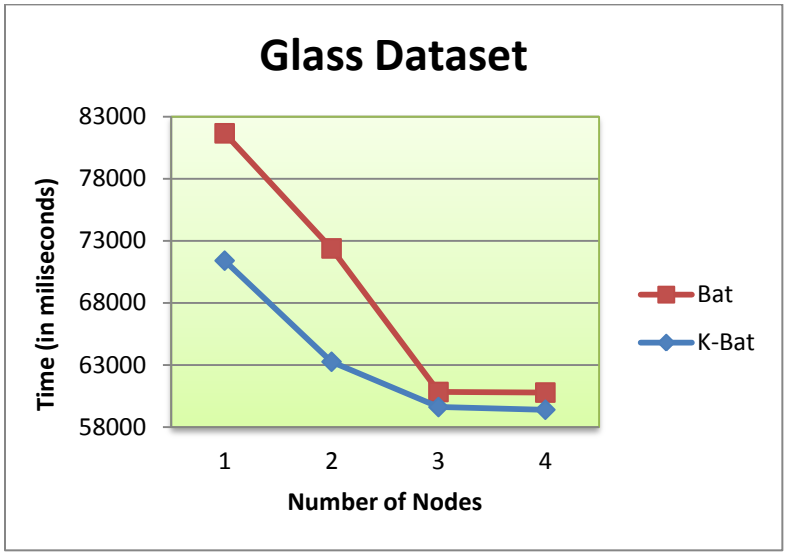

Figure 5-2. Number of Nodes VS Time graph for Glass Dataset

It can be observed from the graph above that since glass is also a small dataset, the use of more number of blocks does bring much alteration in the execution times of the algorithms.

3. *Wine Dataset:*

The result of this dataset reaches the best value within 100 iterations of the K-Bat algorithm. Thus100 iterations have been run and time has been evaluated for Bat and K-Bat algorithms as the number of nodes increase.

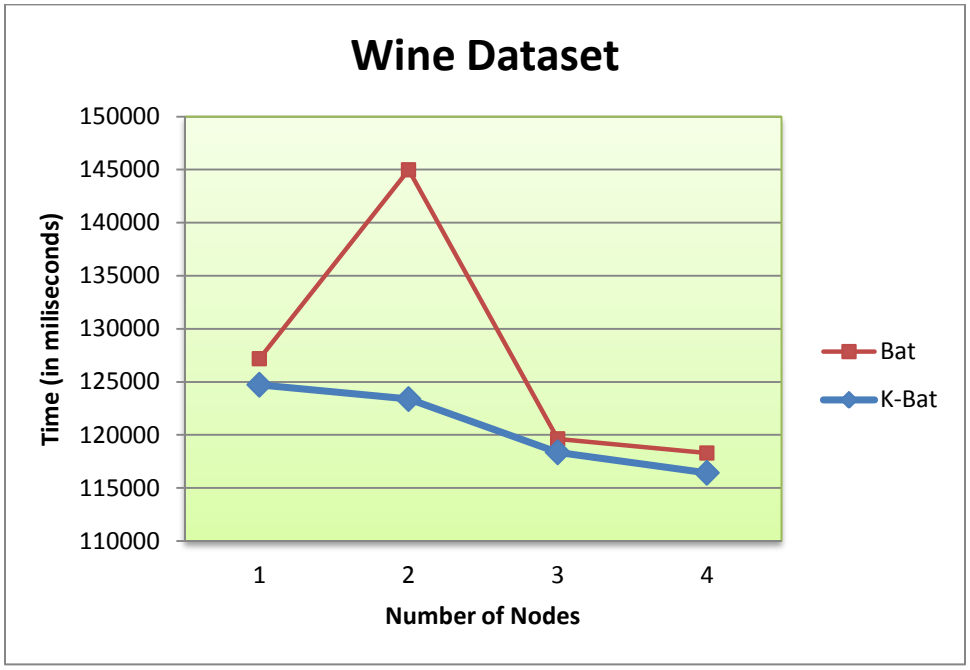

Figure 5-3. Number of Nodes VS Time graph for Wine Dataset

This dataset too has less number of data values and thus the variation in execution time with the number of nodes on Hadoop is not large.

4. ***Magic Dataset:***

The result of this dataset reaches the best value within 100 iterations of the K-Bat algorithm. Thus100 iterations have been run and time has been evaluated for Bat and K-Bat algorithms as the number of nodes increase.

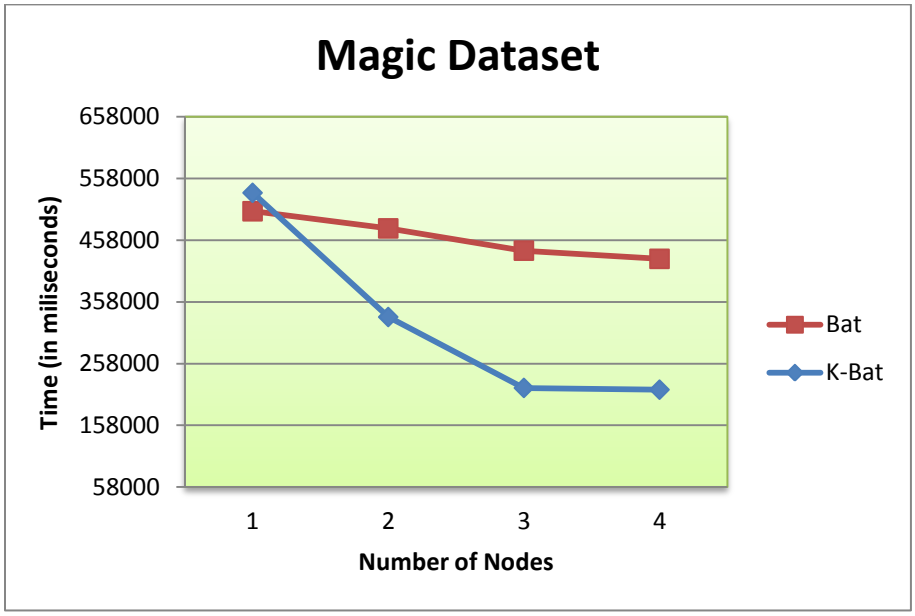

Figure 5-4. Number of Nodes VS Time graph for Magic Dataset

This dataset is medium sized and thus as the number of nodes increase from 1 to 4, the time almost decreases to less than half because of the division between mappers. This example clearly shows the effect of using MapReduce for larger datasets.

5. ***Pokerhand Dataset:***

The result of this dataset reaches the best value within 50 iterations of the K-Bat algorithm. Thus, 50 iterations have been run and time has been evaluated for Bat and K-Bat algorithms as the number of nodes increase.

This is the largest dataset that has been used for our review work in this thesis and gives the best explanation of the decrease in time as the number of nodes on the cluster is increased. Since we have taken a smaller size of block, this dataset

forms 21 blocks for distribution and gives maximum efficiency as number of nodes are increased.

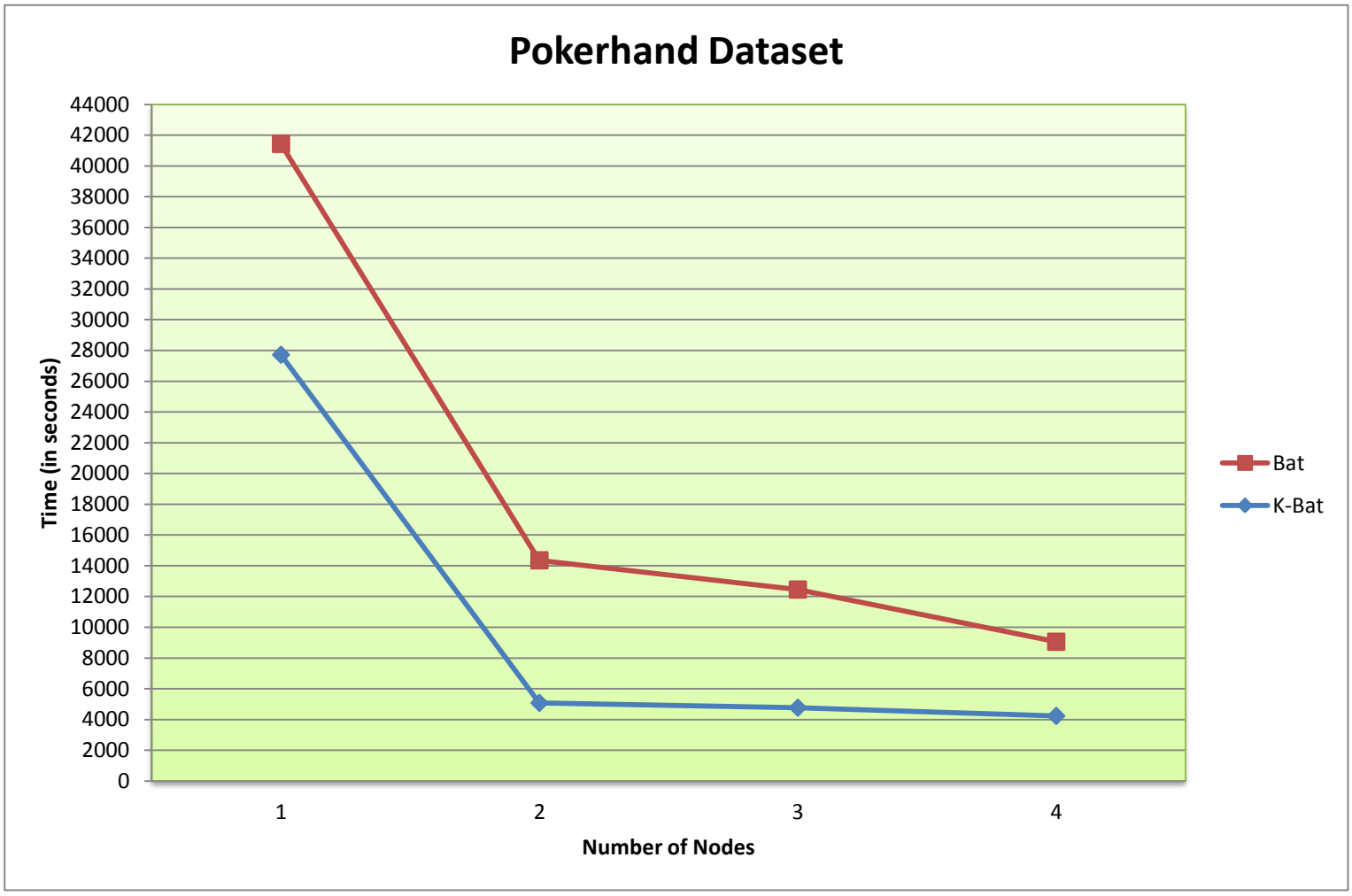

Figure 5-5. Number of Nodes VS Time graph for Pokerhand Dataset

5.5 Convergence Graphs

Convergence graphs have been made for the datasets that represents how fast the fitness value reaches convergence with the number of iterations. 1000 iterations have been run for all datasets except the Iris dataset that reaches best value within the limit of 100 iterations. These graphs show the efficiency of our proposed algorithm to reach the best value faster. K-means algorithm, Bat algorithm and K-Bat algorithm have been compared for this result.

1. ***Iris Dataset:***

   100 iterations of Iris have been run on all three algorithms and their convergence graphs have been plotted.

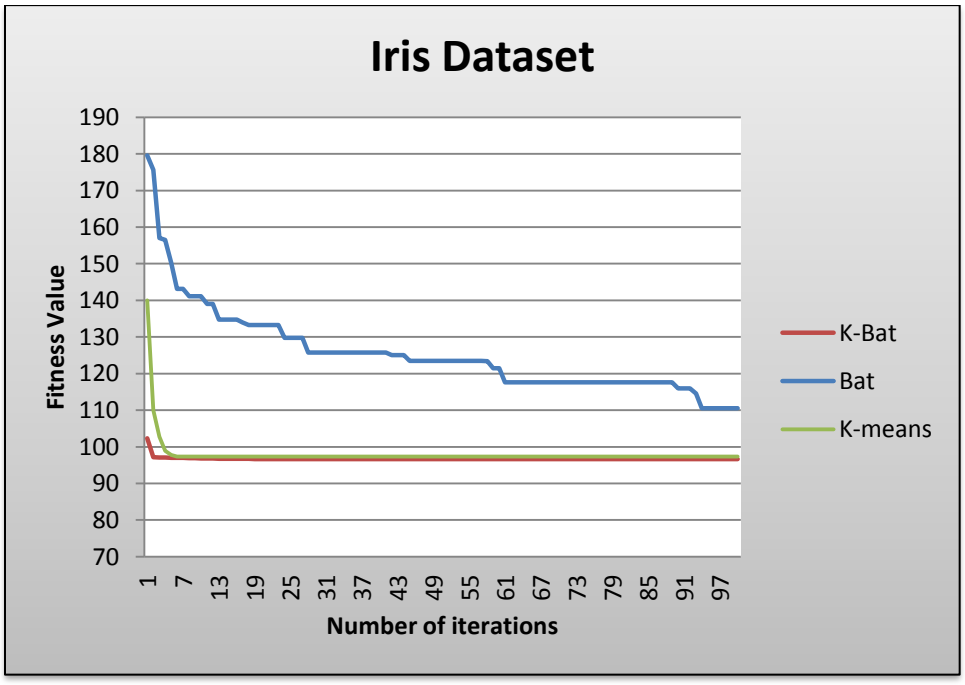

Figure 5-6. Number of iterations VS Fitness Value Graph for Iris Dataset

Above graph shows that our proposed algorithm reaches the convergence point the fastest. The K-means algorithm also reaches the result fast but the value obtained by K-Bat (96.6555) is better than the value obtained by K-means (97.3259).

2. ***Glass Dataset:***

1000 iterations of Iris have been run on all three algorithms and their convergence graphs have been plotted.

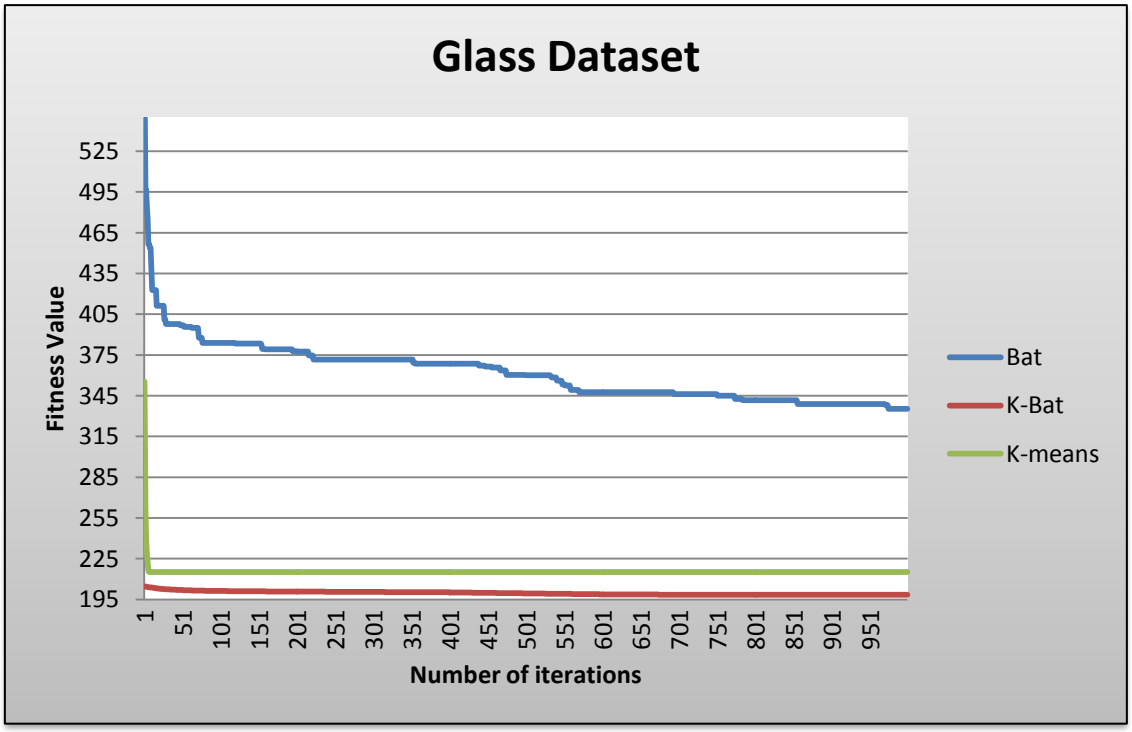

Figure 5-7. Number of iterations VS Fitness Value Graph for Glass Dataset

K-Bat and K-means both reach fast convergence value with K-Bat giving better solution than K-means algorithm.

A closer look at the difference in results for the two algorithms can be better seen in the above graph shown in maximized way below:

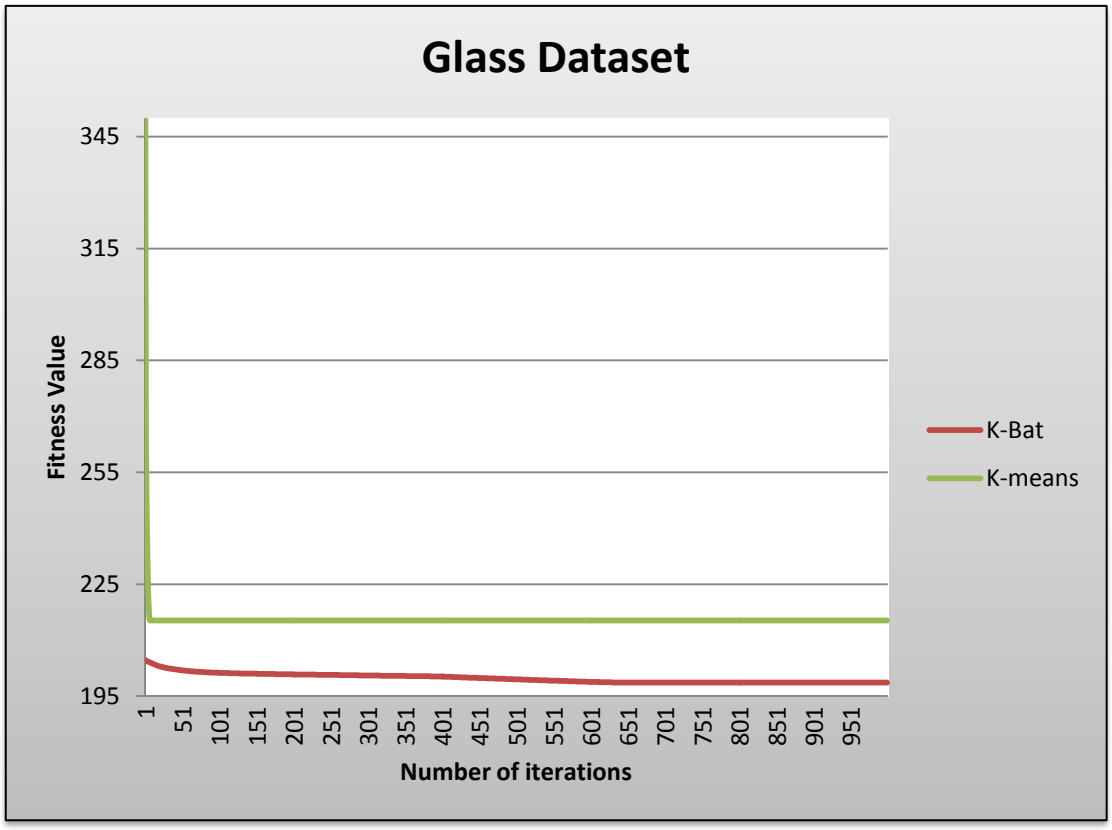

Figure 5-8. Number of iterations VS Fitness Value Graph for Glass Dataset (Closer Look)

3. ***Wine Dataset:***

1000 iterations of Iris have been run on all three algorithms and their convergence graphs have been plotted.

This distinctive wine dataset shows very clear difference between the curves of the three datasets. The K-Bat can be seen to obtain the best value when compared to other 2 base algorithms. In fact K-Bat starts at the point at which K-means will converge to its best value. This shows the power of this algorithm.

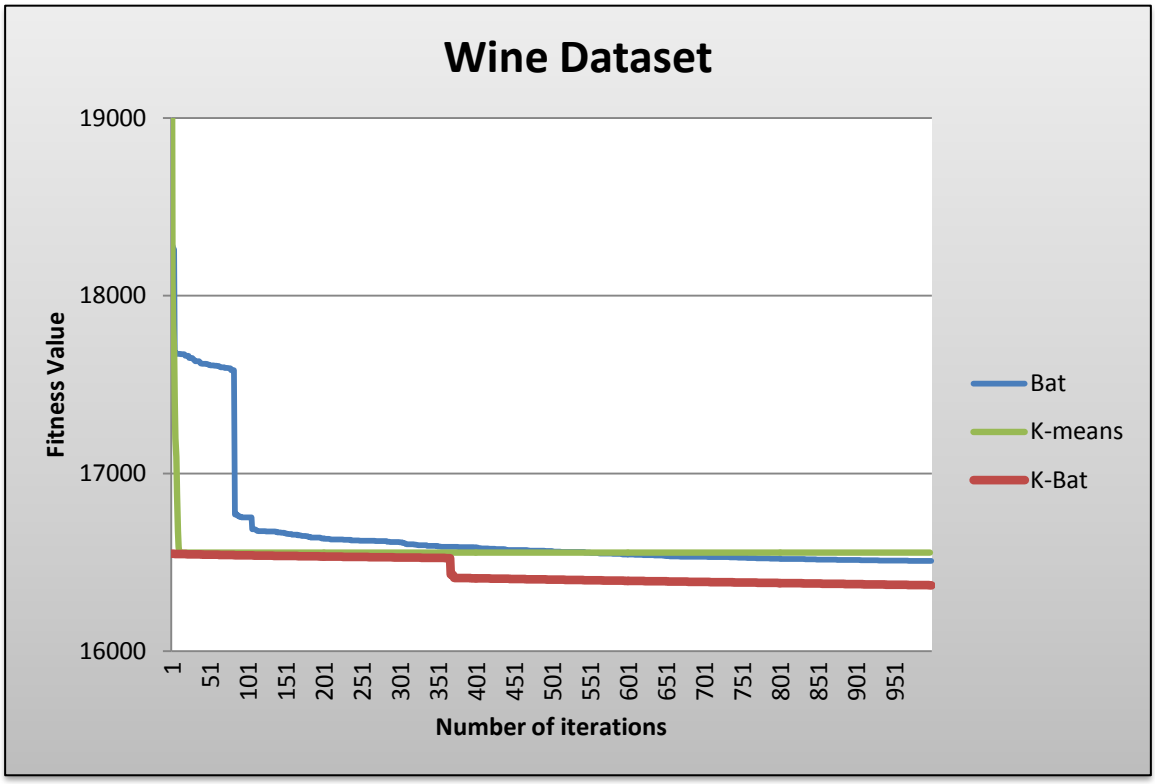

Figure 5-9. Number of iterations VS Fitness Value Graph for Wine Dataset

This distinctive wine dataset shows very clear difference between the curves of the three datasets. The K-Bat can be seen to obtain the best value when compared to other 2 base algorithms. In fact K-Bat starts at the point at which K-means will converge to its best value. This shows the power of this algorithm.

4. ***Magic Dataset:***

1000 iterations of Iris have been run on all three algorithms and their convergence graphs have been plotted.

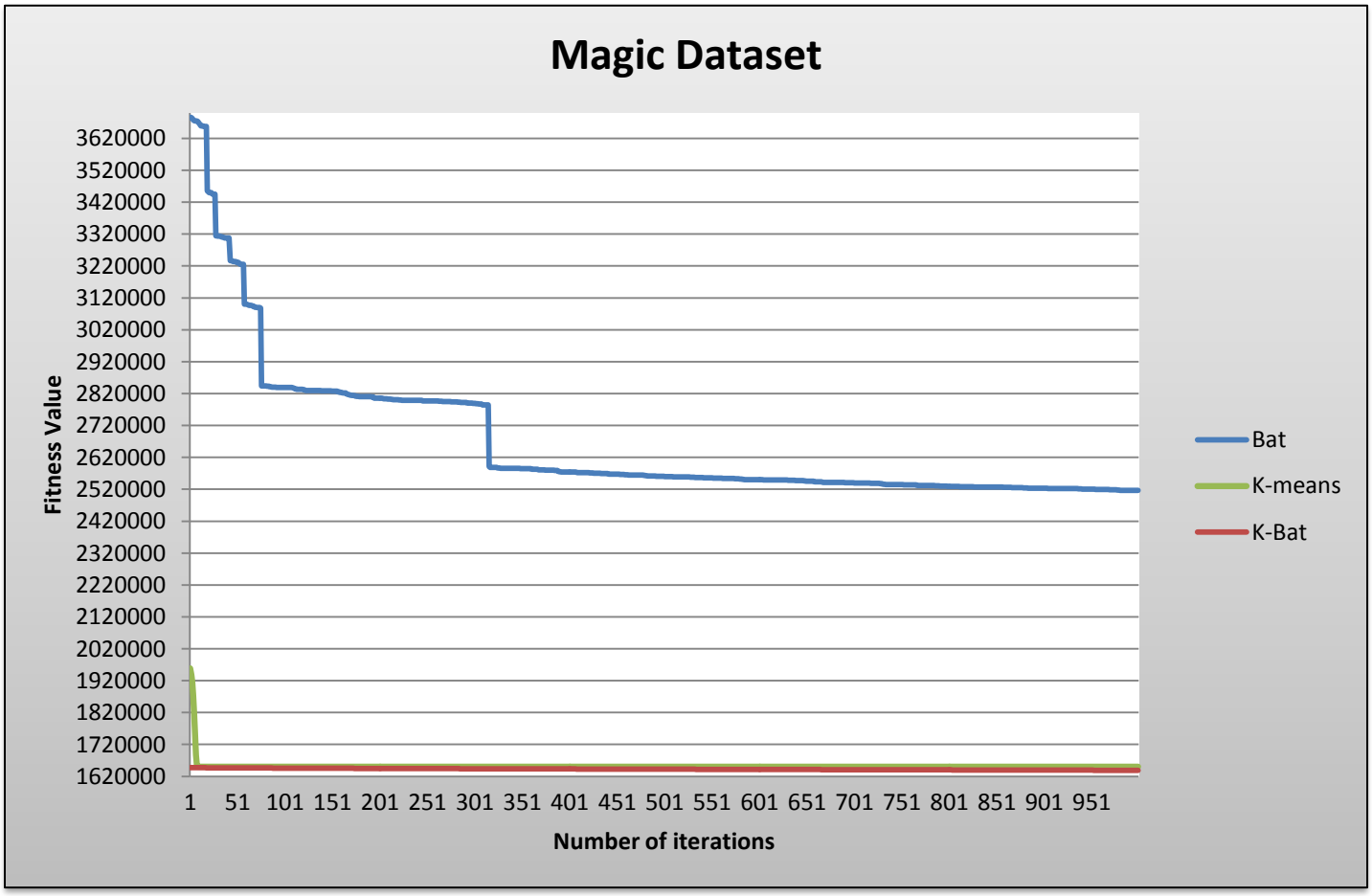

Figure 5-10. Number of iterations VS Fitness Value Graph for Magic Dataset

This graph shows the poor exploration character of the basic Bat algorithm as it gets stuck in some local minima value. K-Bat and K-means converge to almost similar but not the same values. K-Bat gives a better result as can be seen from graph given below:

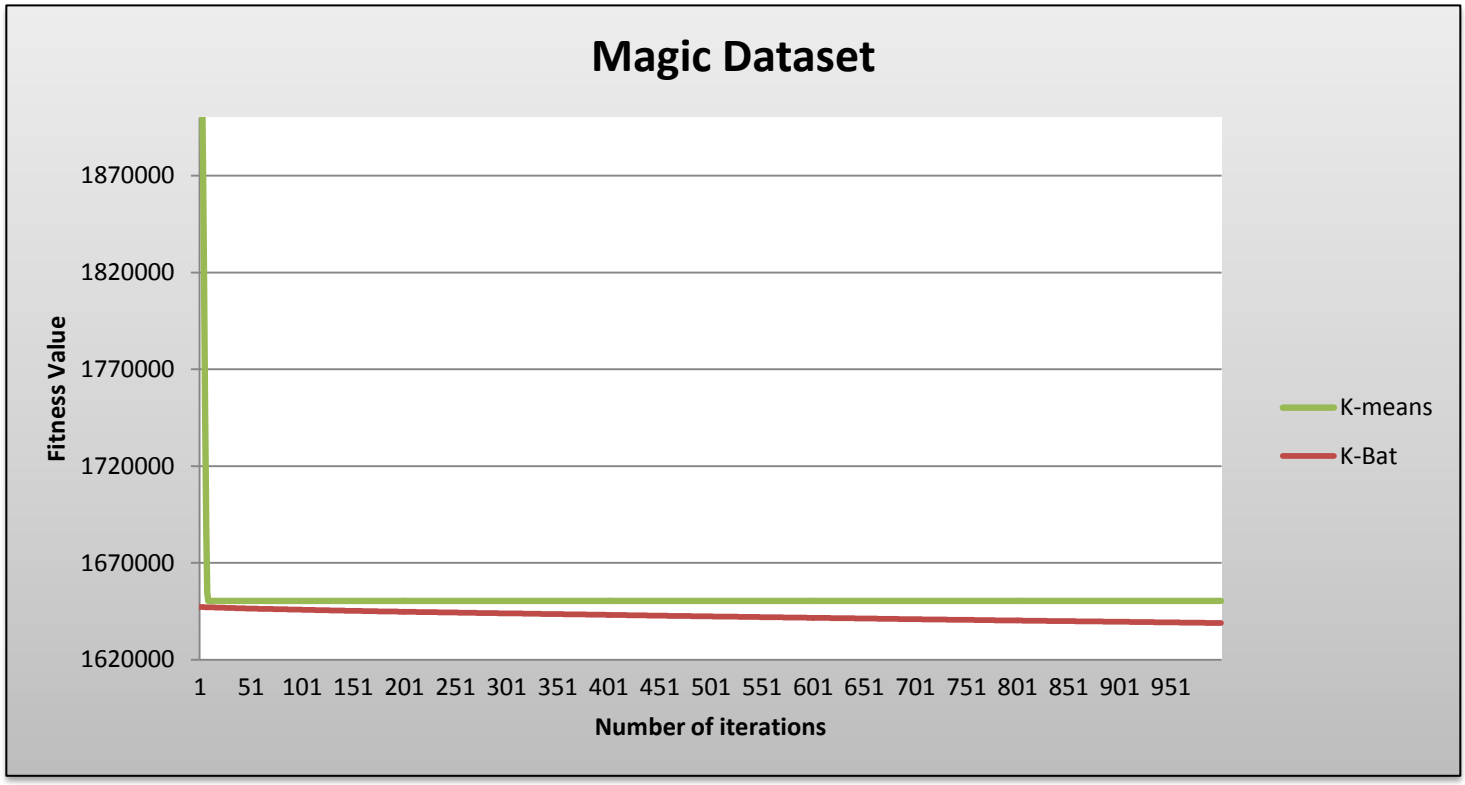

Figure 5-11.Number of iterations VS Fitness Value Graph for Magic Dataset (Closer Look)

This graph shows that K-Bat algorithm achieves better result on the Magic dataset than the K-means algorithm. It can be observed that K-Bat starts lower than K-means. K-means takes a steep drop but is not able to reach values close to K-Bat.

Chapter Six: **CONCLUSION AND FUTURE SCOPE**

We dedicate our work to finding an efficient solution to the clustering problem. Cluster analysis is an important part of any Data Mining process. To group similar data together, and mine important information from them, cluster analysis is applied. We have developed the K-Bat algorithm to perform clustering for computationally large inputs. Our proposed structure combines the advantages of the famous K-means and Bat algorithms, while eliminating the defects of the two. It is tested on 5 benchmark datasets and it outperforms both these algorithms to give exceptionally good results.

Additionally, a MapReduce structure of the same algorithm has also been developed. This structure is able to solve clustering problem for extremely large datasets. When input size increases, K-Bat algorithm is able to scale well. It is tested on a Hadoop cluster setup of 4 systems. It is observed that as the number of nodes on the cluster is increased, the execution time decreases. This shows that K-Bat on MapReduce is able to perform clustering faster than when run sequentially. Thus, parallelization of the K-Bat algorithm has been successfully achieved. Future work in this area is that K-Bat algorithm can be applied to real datasets to exploit the strength of this algorithm.

REFERENCES

[1] A. Jain, M. Murty and P. Flynn, "Data Clustering: A Review," ACM Computing Surveys, Vol. 31, No. 3., 1999.

[2] A. Likas, N. Vlassis and J. J. Verbeek, "The global k-means clustering algorithm," *The Journal of Pattern Recognition,* 2003.

[3] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary Computation, Elsevier,* 2014.

[4] M. Dixit, N. Upadhyay and S. Silakari, "An Exhaustive Survey on Nature Inspired Optimization Algorithms," *International Journal of Software Engineering and Its Applications,* vol. Vol. 9, pp. 91-104, 2015.

[5] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations:International Journal for Computer-Aided Engineering and Software,* vol. 29, 2012.

[6] R. Xu and D. Wunsch II, "Survey of Clustering Algorithms," *IEEE Transactions On Neural Networks,* vol. Vol. 16, no. No.3, 2005.

[7] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *The Journal of Pattern Recognition Society,* 2000.

[8] S. Bandyopadhyay and U. Maulik, "An evolutionary technique based on K-Means algorithm for optimal clustering in RN," *Information Sciences Journal,* 2002.

[9] D. v. d. Merwe and A. Engelhrecht, "Data Clustering using Particle Swarm Optimization," in *IEEE*, 2003.

[10] X. Cui, T. E. Potok and P. Palathingal, "Document Clustering using Particle Swarm Optimization," in *IEEE*, 2005.

[11] E. Rashedi, H. Nezamabadi-pour and S. Saryazdi, "GSA: A Gravitational Search Algorithm," *Information Sciences*, 2009.

[12] A. Hatamlou, S. Abdullah and Z. Othman, "Gravitational Search Algorithm with Heuristic Search for Clustering Problems," in *3rd Conference on Data Mining and Optimization*, 2011.

[13] S. Fong, S. Deb, X.-S. Yang and Y. Zhuang, "Towards Enhancement of Performance of K-Means Clustering Using Nature-Inspired Optimization Algorithms," *The Scientific World Journal*, vol. 2014, p. 16, 2014.

[14] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Information Sciences*, 2012.

[15] A. A. A. Esmin, R. A. Coelho and S. Matwin, "A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data," in *Springer Science+Business Media Dordrecht 2013*, 2013.

[16] V. Kumar, J. K. Chhabra and D. Kumar, "Automatic cluster evolution using gravitational search algorithm and its application on image segmentation," *Engineering Applications of Artificial Intelligence, Elsevier*, 2013.

[17] A. P. Piotrowski, J. J. Napiorkowski and P. M. Rowinski, "How novel is the ‘‘novel’’ black hole optimization approach?," *Information Sciences,* 2014.

[18] A. Bahrololoum, H. Nezamabadi-pour, H. Bahrololoum and M. Saeed, "A prototype classifier based on gravitational search algorithm," *Applied Soft Computing,* 2012.

[19] R. L. F. Cordeiro, C. Traina Jr., A. J. M. Traina, J. López, U. Kang and C. Faloutsos, "Clustering Very Large Multi-dimensional Datasets with MapReduce," *KDD,* 2011.

[20] T. Sun, C. Shu, F. Li, H. Yu, L. Ma and Y. Fang, "An Efficient Hierarchical Clustering Method for Large Datasets with Map-Reduce," in *International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2009.

[21] C.-W. Tsai, C.-H. Hsieh and M.-C. Chiang, "Parallel Black Hole Clustering Based on MapReduce," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2015.

[22] A. H. Gandomi, X.-S. Yang, A. H. Alavi and S. Talatahari, "Bat algorithm for constrained optimization tasks," in *Neural Computing & Applications*, 2013.

[23] A. M. Taha and A. Y. Tang, "Bat Algorithm For Rough Set Attribute Reduction," *Journal of Theoretical and Applied Information Technology,* vol. 51, 2013.

[24] "An Optimized K-Means Clustering Technique Using Bat Algorithm".

[25] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa and X.-S. Yang, "BBA: A Binary Bat Algorithm for Feature Selection," in *SIBGRAPI*

Conference on Graphics, Patterns and Images, 2012.

[26] A. R. Jordehi, "Chaotic bat swarm optimisation (CBSO)," *Applied Soft Computing*, 2015.

[27] A. Alihodzic and M. Tuba, "Improved Bat Algorithm Applied to Multilevel," *Scientific World Journal*, 2014.

[28] N. S. Jaddi, S. Abdullah and A. R. Hamdan, "Optimization of neural network model using modified bat-inspired algorithm," *Applied Soft Computing*, vol. 37, 2015.

[29] A. Shukla, "A modified Bat Algorithm for the Quadratic Assignment Problem," in *IEEE*, 2015.

[30] S. Talatahari and A. Kaveh, "Improved Bat Algorithm For Optimum Design Of Large Scale Truss Structres," *International Journal Of Optimization In Civil Engineering*, 2015.

[31] I. Fister Jr., S. Fong, J. Brest and I. Fister, "A Novel Hybrid Self-Adaptive Bat Algorithm," *The Scientific World Journal*, 2014.

[32] G. Wang and L. Guo, "A Novel Hybrid Bat Algorithm with Harmony Search for," *Journal of Applied Mathematics*, 2013.

[33] S. Yılmaz, E. U. Kucuksille and Y. Cengiz, "Modified Bat Algorithm," *IEEE ELEKTRONIKA IR ELEKTROTECHNIKA*, vol. 20, 2014.

[34] A. B. Serapião, G. S. Corrêa, F. B. Goncalves and V. O. Carvalho, "Combining K-Means and K-Harmonic with Fish School SearchAlgorithm for data clustering task

on graphics processing units," *Applied Soft Computing*, 2016.

[35] M. Mahdavi, M. H. Chehreghani, H. Abolhassani and R. Forsati, "Novel meta-heuristic algorithms for clustering web documents," *Applied Mathematics and Computation*, 2008.