Baad: A Self-Optimizing Algorithm For Anomaly Detection

Adeel Shiraz Hashmi, Tanvir Ahmad

Abstract*: Anomaly/Outlier detection is the process of finding abnormal data points in datasets or data streams. Anomaly detection finds its application in various fields like network intrusion detection, fraud detection, fault detection, etc. There are many anomaly detection algorithms available in the literature but most of these algorithms require setting of some parameters which significantly affect the performance of the algorithm. These parameters are generally set by hit-and-trial, hence performance is compromised with default or random values. In this paper, the authors propose a self-optimizing algorithm for anomaly detection based on bat meta-heuristic, and named as Bat Algorithm for Anomaly Detection (BAAD). The proposed solution is a non-clustering unsupervised learning approach for anomaly detection. The BAAD algorithm belongs to the category of density-based algorithms, and aims to find the optimal value of neighborhood radius as done in the LOCI (Local Correlation Integral) algorithm for anomaly detection, though the approach used in the proposed solution is different. The algorithm is implemented on Apache Spark for scalability and thus the solution can handle big data as well and provides fast results. Experiments were conducted on various datasets, and the results show that the proposed solution is much accurate than the standard algorithms of anomaly detection.

***Index Terms*: Anomaly detection, bat algorithm, big data, parallel algorithms.**

I. INTRODUCTION

Anomaly detection is the process of identifying those patterns in data which do not conform to expected behavior. These patterns are called as anomalies or outliers. According to Hawkins [1], the definition of an outlier is: "An outlier is an observation that deviates so much from other observations (considered normal) as to arouse suspicion that it was generated by a different mechanism". Identification of these anomalies is significant in many cases, for instance, an abnormal traffic pattern in a computer network could mean that an unauthorized user is trying to compromise a system in the network. Anomaly detection finds application in a wide variety of domains such as intrusion detection in the field of networks and security, fraud detection in the field of banking and insurance, fault detection in wireless sensor networks, etc.

Swarm Intelligence [2] is a category of nature-inspired algorithms which mimic the behavior displayed by living (or even artificial) organisms while performing some task like foraging, mating, etc. Swarm intelligence algorithms can find an optimal solution for the problems when evaluation of all the possible solutions is not possible. These algorithms find their application in various areas including the field of anomaly detection.

In this era of big data, a machine learning tool should be scalable, fast and accurate. Apache Hadoop [3] is the most popular choice when scalability in data processing is desired. Apache Spark [4] is another tool which can be utilized for handling big data. Spark has been proven to be faster than Hadoop due to its in-memory analytics capabilities. Spark provides a SQL-like interface to analyze structured datasets, and it also provides APIs to develop new distributed machine learning algorithms. A Spark-based implementation of an anomaly detection algorithm makes it scalable and also reduces the turnaround time considerably.

This work deals with swarm intelligence for anomaly detection on a big data tool. The rest of the paper is structured as follows: Section II deals with the literature survey of anomaly detection and swarm intelligence, the original bat algorithm is discussed in Section III, Section IV presents the proposed bat algorithm for anomaly detection, Section V is dedicated to experimentations and results, and Section V gives the conclusion of the work done.

II. LITERATURE REVIEW

In this section the literature survey of anomaly detection algorithms and swarm intelligence is done. Swarm intelligence algorithms have been actively used in literature for many optimization problems, and can also be used to optimize the parameters of anomaly detection algorithms.

A. Anomaly Detection

The anomaly detection techniques can be broadly divided into three categories: visual/graphical, statistical and machine learning. The visual techniques for anomaly detection includes scatter-plots which can help us discover anomalies in 2D datasets, but for large and multi-dimensional datasets we need some enhanced techniques like sparse traversal, etc. [5]. A statistics based definition of an outlier is: "an outlier is any data point more than 1.5 interquartile ranges (IQR) below the first quartile or above the third quartile"; based on this definition box-plots can be used for outlier detection [6]. Other statistical measures like frequencies [7] and entropy [8] have also been used in literature for anomaly detection.

The machine learning based algorithms for anomaly detection can be divided into three categories: supervised, semi-supervised and unsupervised. Supervised anomaly detection is analogous to a binary classification problem, in which a training dataset is provided

Revised Manuscript Received on May 06, 2019

Adeel Shiraz Hashmi, Department of Computer Engineering, Jamia Millia Islamia, Delhi, India.

Tanvir Ahmad, Department of Computer Engineering, Jamia Millia Islamia, Delhi, India.

containing points labeled as normal and outliers. All the classifiers like kNN, SVM, perceptron, Naive Bayes, Random Forest, etc. can be used for anomaly detection. However, supervised learning is not often used for anomaly detection as the ratio of outliers to normal points is quite low, hence training for outlier class is often not sufficient which leads to poor performance of the algorithms. Instead of supervised learning, semi-supervised learning is preferred for detecting anomalies. Semi-supervised learning is analogous to one-class supervised learning i.e. a model of normal behavior is constructed from a training data set. Those points in the test dataset which deviate from the normal are labeled as anomalies. The popular examples of semi-supervised algorithms are One-Class SVMs and Replicator Neural Networks.

Most of the anomaly detection algorithms belong to the category of unsupervised learning. These unsupervised learning based algorithms can further be classified as distance-based and density-based. In distance-based approach, an object O in the dataset D is declared as an outlier if at least fraction *p* of the objects in D are farther than distance *d* from O [9]. Angiulli and Pizzuti [10]–[12] calculated the sum of distances of the point under consideration with rest of the point in the dataset, and identified top *n* points which had highest cumulative sum (marked as outliers). Top-*n* kNN [13] is a simple approach for outlier detection where a distance measure like Euclidean distance can be used to identify *k* (user-defined value) nearest neighbors for each object, and declare *n* objects having highest cumulative/average value from its *k* neighbors as outliers. Distance-based clustering algorithms like k-means, PAM (partition around medoids), etc. have also been used for outlier detection. In outlier detection by k-means algorithm, we initially make *k* clusters using k-means clustering algorithm, then we identify *n* points which have largest distance from their cluster centers to get *n* outliers.

DBSCAN [14] is one of the earliest algorithms to identify outliers, although its main objective is clustering. In DBSCAN, we set two parameters *eps* and *minPts*; we then start by picking a random point in the dataset and considering all the points in its neighborhood which are at a distance less than *eps* away from it to form the cluster. Next we count the number of such point, if this count is greater than *minPts* then this cluster is referred as dense cluster. We expand the cluster by repeating the process for all the new points in the cluster. If we run out of new points and there are still points left to consider in the dataset then we again pick a random point and restart the process till all points in the dataset are considered. A point which has less than *minPts* in its cluster and is also not a part of any other cluster is declared as an outlier.

LOF [15] is an anomaly detection algorithm and is not based on clustering. It calculates a LOF score for every data instance, and instances with high LOF scores are marked as outliers. In LOCI (Local Correlation Integral) [16], a MDEF value is calculated for each point. If the MDEF value deviates three times (or more) from the standard deviation of MDEFs in the neighborhood then the point is marked as an outlier.

Various strategies for anomaly detection in very large and high dimensional datasets have been discussed in literature. Anna et al. [7] implemented AVF (Attribute Value Frequency) in Map-Reduce for fast outlier detection in large categorical datasets. Liu et al. [8] proposed an Entropy-based Fast Detection algorithm for outlier detection in large datasets. Yan et al. [17] proposed a distributed outlier detection algorithm employing compressive sensing for sampling high-dimensional data. DROUT (Dimensionality Reduction for OUTlier detection) can be used for anomaly detection in “very wide” datasets i.e. datasets with large number of features. Distributed Solving Sets (DSS) and Lazy Distributed Solving Sets (LDSS) [18] are distributed strategies for anomaly detection in large datasets. For shorter turnaround time, GPU versions of parallel algorithms like DSS have also been implemented. RAD (Rapid Anomaly Detection) algorithm is used by NetFlix for outlier detection. RAD is able to handle high cardinality dimensions, non-normalized datasets, seasonality and minimizes false positives. RAD is based on Robust Principal Component Analysis (RPCA) which repeatedly calculates SVD (Singular Value Decomposition) and applies thresholds to singular values in each iteration. Recently, Twitter released an R package for anomaly detection in time-series data, which consists of S-H-ESD (Seasonal Hybrid extreme Studentized deviate) test for anomaly detection.

B. Swarm Intelligence

Swarm Intelligence is a class of nature-inspired algorithms based on collective behavior displayed by swarm of insects/organisms/animals in achieving some goal like foraging (act of wandering in search of food and other resources). These algorithms like other nature-inspired algorithms are used for optimization problems like minimizing a convex function, finding an optimal cluster, finding shortest route, etc.

Meta-heuristic term in the field of mathematical optimization refers to a high-level procedure which can be used for providing a good solution to an optimization problem, especially when exploration of whole search space is not possible. There are hundreds of swarm-based meta-heuristics available in the literature, and some of the popular ones are listed in Table I.

Ant colony algorithm [19] is based on the ability of ants to find shortest path to the food source. Initially, the ants wander randomly in search for food. On locating food, they return to colony leaving a pheromone trail. When other ants come across such a trail, they quit wandering and follow the trail. The pheromones tend to evaporate with time, therefore pheromone density remains high on shorter paths compared to longer ones. So, the ants tend to follow the shortest path on which the pheromone density is highest.

Particle Swarm Optimization [20] is based on bird flocking. The initialization in PSO is similar to that of genetic algorithm; a random population is generated to represent the members of the swarm. The swarm is updated throughout the generations in order to search for optimum. The particles fly through the solution space based on their own best value and the swarm’s best value to obtain the global best.

Table I. Swarm Intelligence meta-heuristics

Algorithm	Proposed by	Year
Ant Colony Optimization	Dorigo, Di Caro	1992
Particle Swarm Optimization	Kennedy, Eberhart, Shi	1995
Bacterial Foraging algorithm	Passino	2002
Artificial Fish Swarm	Li et. Al	2003
Glow-worm	Krishnanad and Ghose	2005
Artificial Bee Colony	Karaboga, Basturk	2007
Firefly algorithm	Xin-She Yang	2008
Cuckoo Search	Xin-She Yang, Deb	2009
Bat algorithm	Xin-She Yang	2010

Bacterial Foraging algorithm [21] is based on the behavior exhibited by bacteria while searching for food. The behavior of bacteria depends upon the chemicals in its environment which may be secreted by other bacteria for communication. The bacteria move in swarms towards a location which has the optimal environment (in terms of food, etc.).

The main concept in Artificial Fish Swarm algorithm [22] is the visual scope of each fish. The visual scope of the fish may be empty, crowded, and normal. If the scope is normal, the fish moves towards the center of the scope. The swarming movement is activated if this central point has better fitness value; otherwise a random point within visual scope is selected (which also must have a better fitness value).

The glow-worm algorithm [23] is based on the concept of bioluminescence/glow which is used by glow worms to communicate with each other. The glow worm with less light intensity move towards the glow worm with higher light intensity.

Artificial Bee Colony [24] algorithm is based on foraging behavior of honey bees. Bee colonies send bees to collect nectar from flower patches in proportion to the amount of nectar available in the patch. These bees return and communicate with other bees at the hive via a waggle dance that informs other bees in the hive about the direction, distance, and quality of food sources.

Cuckoo Search [25] algorithm is inspired by the brood parasitism of cuckoos. Cuckoos are known to hide their eggs in nests of birds of other species. The host bird may identify the cuckoo' s egg and destroy it but it can even leave its own nest with cuckoo eggs and build a new nest and take away its egg to the new nest.

III. BAT ALGORITHM

Bat algorithm is a meta-heuristic for global optimization, developed by Xin-She Yang in 2010 [26]. It is based on the concept that all bats use echolocation to sense distance i.e. they emit pulses with a certain loudness, which like a sonar are used to detect the distance of a prey, obstacle or a roosting crevice. In the algorithm, the global optimum is the position of the prey and the positions of the bats are solutions in the search space.

Let a virtual bat fly randomly with velocity v_i at position x_i, while emitting pulses of frequency f_i with pulse emission rate r_i and loudness A_i. As it searches for its prey it changes its frequency and loudness. We assume that the frequency lies in the range $[f_{min}, f_{max}]$ and pulse emission rate lies in the range [0,1] where 0 indicates no pulse and 1 indicates maximum pulse emission rate. The loudness decreases once the bat has found its prey while the rate of pulse emission increases.

Bat meta-heuristic

- Objective function f(x), $x=(x_1,x_2,\ldots,x_d)^T$
- Initialize the bat position x_i and velocity v_i $(i=1,2,\ldots, n)$
- Define pulse frequency f_i at x_i
- Initialize pulse emission rates r_i and the loudness/ amplitude A_i

while (t < Max number of iterations)
  Change bat positions (i.e. generate new solutions) by adjusting frequency, and updating velocities and locations [Eq. (1) to (3)]
  if (rand > r_i)
    Select a solution (x*) among the best solutions
    Generate a local solution around the selected best solution [Eq. (4)]
  end if
  Generate a new solution by flying randomly
  if (rand<A_i and f(x_i)<f(x*))
    Accept the new solution
    Increase r_i and reduce A_i [Eq. (5) and (6)]
  end if
  Rank the bats and find the current best x*
end while
Process results

In the bat algorithm, there are generic parameters like number of bats, amplitude, emission rate, etc. as well as problem specific parameters like fitness function. The algorithm starts with random population of bats i.e. a random set of solution. A problem-specific fitness function has to be defined which is basically a heuristic to evaluate the solution. The solution with good fitness is said to be near to the prey and those with low fitness are said to be far away from the prey. The solutions with low fitness are updated (acc. to Eq. 1-3), and some solutions are moved towards the local best solution whereas some are moved randomly.

We assign frequency of pulse emission at random to a bat acc. to Eq. 1.

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \quad (1)$$

where, β is some random constant in the range 0-1.

The velocity and position of the bat is updated acc. to Eq. 2 and 3.

$$v_i^{t+1} = v_i^t + (x_i^t - x_*) f_i \quad (2)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3)$$

where, x_* is the current best solution.

For some bats (chosen at random), we generate new solutions close to best solution, and for remaining keep the solutions obtained by updating the velocities (to search far away from current best so as to reach global minima):

(i) If pulse emission rate of a bat is lower than a random value then generate a solution around the best solution acc. to the equation,

$$x = x_* + \eta A^t \quad (4)$$

where $\eta \in [-1,+1]$ is a random number, A^t is the average loudness of all bats.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

(ii) If fitness of a new solution obtained by updating velocity (new position of bat) is better (than old position of bat) and loudness is more than a random number (this is done to keep some values from the original solutions) then accept new solutions, and decrease loudness and increase pulse emission rate (indicating that bat has found its prey) acc. to equations,

$$A_i^{t+1} = \alpha A_i^t \quad , \tag{5}$$

$$r_i^{t+1} = r_i^0[1-\exp(-\gamma t)] \tag{6}$$

where, α and γ are constants, such that 0<α<1 and γ>0.

IV. BAAD ALGORITHM

A. Anomaly Detection by Swarm Intelligence

According to Knorr's definition of an outlier, a data point is an outlier if it has a fraction β or less of the total points within distance *r* (Fig. 1). The values of β and *r* are user-defined and very hard to decide. Instead of selecting values of β and *r* randomly, an optimization algorithm can be used to obtain these values. The idea is: for each point *p*, consider a circle of radius *r* with *p* being the center, find the number of points being enclosed by this circle; let the number of points being enclosed is *k*, then the aim is to find the point *p** and radius *r** such that the value of *k/r* is minimized. This value of *r** is then used to compute the *k/r* ratio of other points and rank them. The points with low *k/r* ratio will be outliers. We can also set a value β such that if *k/n*<β (n is total number of points) for a point, then such a point will be an outlier.

Swarm intelligence algorithms are used in many optimization problems and can also be used for this problem. The fitness function is *k/r* and we have to minimize this fitness function. However, if *r* is very small then *k* may be 0 giving the least possible value of *k/r*, and if *r* is very large then also $k/r \to 0$ when $r \to \infty$.Hence, the value of *r* should not be too high or too low. Therefore, we need to set a lower limit as well as an upper limit on the value of *k*.

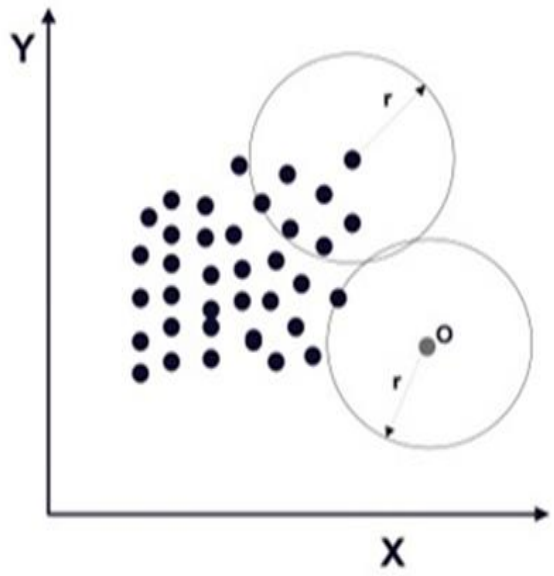

Fig. 1. Knorr's definition of outlier

So, acc. to [27] the fitness function that could be used is:

$$\frac{\alpha}{r*k} + \frac{k}{r} + \frac{k}{n-k} \tag{7}$$

The first term is to limit the lower bound of *r*, and third term is to limit the upper bound of *r*. For a small value of *r*, the first term will be high so fitness function will return a high value (we will consider minimum value of *k* to be 1 i.e. we will consider the point itself in the value of *k*). If we set value of $\alpha=0.05n$, then it will ensure that the value of *r* will be large enough. The third term becomes infinity for $k=n$ (case when *r* is very large), so points for which third term comes out to be infinity will clearly not be outliers. This ensures that *r* is not too small which may lead to missing the outliers or too large to include outliers from the normal set of points.

The swarm intelligence algorithm finds the value of *r* for which the above given fitness function is minimized (let's say the value is *rbest*). To identify the anomalies, the fitness for each point is calculated with radius *rbest*. The points are ranked according to increasing fitness value. The points with lowest fitness value are marked as outliers (as they are deemed highly isolated). The number of points marked as outliers may be user-defined or it may be some small fraction of the total number of points.

B. Bat Algorithm for Anomaly Detection (BAAD)

Bat algorithm is a popular swarm-based intelligent algorithm, and can be used for optimizing the fitness function expressed in Eq. (7) to obtain the optimal value of *r* which can be further utilized to detect the outlier points in the data. The reason behind selection of bat algorithm for optimization is that it has been shown to give better performance than genetic algorithm, PSO, and cuckoo search [26][28].

Bat Algorithm for Anomaly Detection

- Initialize the bat vector *x* i.e. a vector of random values of *r*.
- Initialize velocity vector (all values set as 0).
- Initialize amplitude and pulse emission rate for all bats (all values set as 0.5).

for(each point in the dataset)
  Find Euclidean distance with rest of the points.
  for(each *r* in *x*)
    Find the number of points within threshold *r*.
    Find the fitness acc. to Eq. (7).
  end for
end for
Find *r* with minimum fitness, set it as local best *r**.
  while (t < Max number of iterations)
    Change values of *x*. [Eq. (1) to (3)]
    if (rand > r_i)
      Generate a local solution around the selected best solution *r**. [Eq. (4)]
    end if
    Generate a new solution by flying randomly.
    if (rand<A_i and f(x_i)<f(x*))
      Accept the new solution.
      Increase r_i and reduce A_i [Eq. (5) and (6)].
    end if
    Rank the bats and find the current best *x**.
  end while
The best solution obtained after all the iterations is the global best solution (gbest).
Find the fitness of each point in the dataset taking *r*=*gbest*.
Return *k* points with minimum fitness as the outliers, rest of points are inliers.

The algorithm starts with randomly initializing the bat vector (a bat corresponds to a value of *r*). Next, the distance vector of each data point is calculated with rest of the data points. This distance vector is utilized for calculating the fitness of each bat, by first identifying the neighborhood count for each value of *r* in bat vector i.e. counting the number of points in the distance vector lying below the corresponding value of *r*, and then applying the fitness function as described in Eq. (7). Fitness of each bat is the lowest fitness value returned among all the data points, for the corresponding value of *r*.

The bats are moved towards bat nearest to the prey i.e. the value of *r* is changed so as to be near (by some fraction) to the best value of *r*. Some bats are moved randomly as well to search in unexplored area of the search space. This process is repeated till convergence.

V. EXPERIMENTS & RESULTS

The BAAD algorithm is implemented in PySpark and run on Spark cluster composed of 4-core, 4GB RAM, i5 machines running Ubuntu 16.10. A Spark DataFrame is a distributed dataset, and the operations performed using the DataFrame API are automatically executed in parallel over the distributed dataset. Hence, the parallelization is implicit and isn't needed to be programmed explicitly in the application.

In the experiments, the performance of the proposed BAAD algorithm is compared with standard algorithms of anomaly detection viz. KMeans, DBSCAN, LOF, and Random Forest. Each of these algorithms require some parameters to be set, however the proposed BAAD algorithm doesn't require setting up of any parameters as it is self-optimizing. For k-means algorithm, the number of clusters was set to 30. For DBSCAN, we do not need to set the number of clusters, but we need to set the parameter *eps* i.e. the maximum distance between two samples for them to be considered as in the same neighborhood, and was given the value 0.8. LOF requires the number of neighbors to consider, and this value was set to 15. Random Forest requires the number of trees, which was set to 10. The performance of the BAAD algorithm is also compared with Genetic Algorithm as well as PSO variants of the algorithm.

The experiments were conducted on openly available UCI datasets: lymphography dataset, wisconsin breast cancer dataset, post-operative dataset, pageblocks dataset, credit card fraud detection dataset, forest cover dataset and kddcup99 dataset (discussed in Table II). The performance was evaluated on the basis of accuracy of the algorithms (Table III and Figure 2). The metric used for accuracy is jaccard similarity between actual labels and predicted labels for each instance in the dataset.

The BAAD algorithm doesn't assign labels to the instances, instead it ranks the instances in decreasing order of probability of being an outlier. So, the user needs to provide the count *k* of outliers to be detected, and the algorithm will label top *k* instances in the returned list as outliers, whereas rest of the instances will be labeled as normal.

Table II. Datasets Summary

Dataset	*Dimensions*	*Outlier ratio*	*Description*
Lymphography	148 X 18	6 / 142	Originally 4 classes: normal find (label 1), metastasis (label 2), malign lymph (label 3), and fibrosis (label 4). Classes "normal find" and "fibrosis" have only 2 and 4 instances, hence treated as outliers.
Post-Operative	90 X 9	26 / 64	The data points are classified into 3 classes: A (64 points), S (24 points) and I (2 points). The points labeled as S and I are treated as outliers.
Wisconsin	483 X 10	39 / 444	Following Hawkins [29], only one of every sixth malignant instance is kept, giving us 39 malignant instances, which are treated as outliers among 444 benign instances.
Page-blocks	5053 X 11	140 / 4913	The text blocks are considered as inliers (4913) and non-text blocks (560) are considered as outliers. We will keep only one of every four non-text instances giving 140 outliers.
Credit Card	284807 X 11	492 / 284315	These transactions are done with 172792 credit cards, hence there are multiple transactions for most of the cards.
Forest Cover	286048 X 10	2747 / 283301	The original Forest Cover dataset has 581012 instances (with 54 attributes) which are classified into 7 classes (1-7). We will consider instances of class 2 (inliers) and 4 (outliers) only, and ignore soil type and wilderness attributes.
Kdd-cup99	567497 X 41	2211 / 567497	Kddcup99 is an intrusion-detection dataset. Every type of attack is labeled as an outlier.

Table III. Accuracy of the algorithms on datasets

Dataset	**Km-eans**	**DBS-CAN**	**LO-F**	**RF**	**GA**	**PSO**	**BA-AD**
Lymph-ography	0.79	0.84	0.92	0.90	0.94	0.96	0.97
Post-Operative	0.74	0.79	0.86	0.87	0.88	0.90	0.91
Cancer	0.64	0.72	0.80	0.84	0.95	0.97	0.97
Page-blocks	0.71	0.80	0.87	0.81	0.93	0.96	0.97
Credit Card	0.69	0.82	0.89	0.85	0.96	0.99	0.99
Forest Cover	0.58	0.80	0.89	0.91	0.95	0.99	0.99
KDD-Cup99	0.55	0.61	0.60	0.68	0.96	0.99	0.99

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

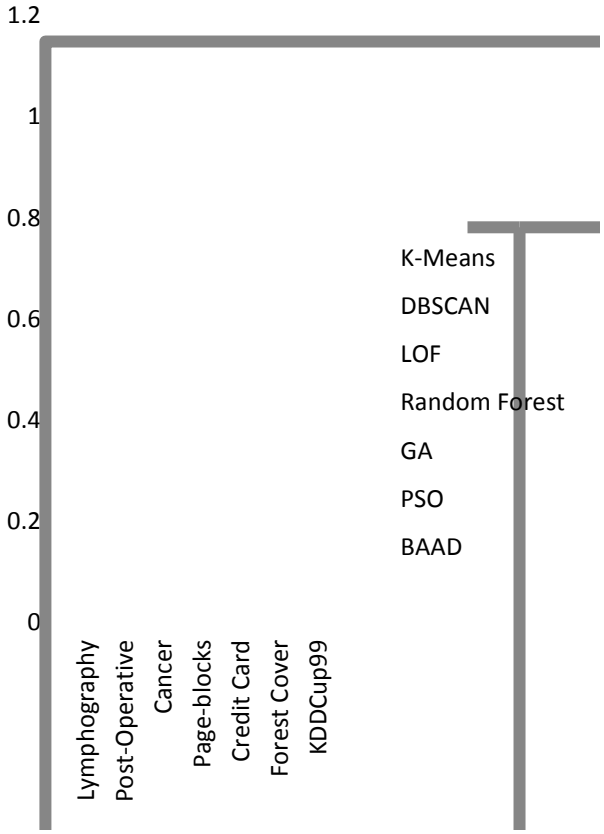

Fig. 2. Comparative Performance of algorithms

The experimental results suggest that the proposed BAAD algorithm is much more accurate than the other state-of-the-art algorithms of anomaly detection. The clustering algorithms like k-means and DBSCAN fare badly, however the random forest algorithm is able to detect anomalies with high accuracy, but it is a supervised learning approach and often the training data is not available for anomaly detection. LOF algorithm with default values also gives good results but the BAAD algorithm clearly outperforms the LOF algorithm.

VI. CONCLUSION

In this paper, the authors proposed a bat meta-heuristic based algorithm for anomaly detection. The major advantage of the proposed solution is that it is self-optimizing and is implemented on a big data tool in a parallel environment. Experiments were performed on multiple datasets to compare the performance of the proposed algorithm with existing algorithms. The results highlighted the drawbacks of existing algorithms which were run with default parameters, and therefore had poor accuracy. The proposed solution being self-optimizing doesn't suffer from such drawback and gives much accurate results.

This work can be extended by replacing bat algorithm with other swarm intelligence algorithms for finding a better alternative. Moreover, optimization techniques other than meta-heuristics can be applied to improve the performance. As far as big data is concerned, tools other than Apache Spark may be utilized and the performance can be compared.

REFERENCES

1. D. M. Hawkins, *Identification of outliers*, Vol. 11, Springer, 1980.
2. G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems", in: *Robots and Biological Systems: Towards a New Bionics?*, Springer, Heidelberg, 1993, pp. 703-712.
3. V. K.Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah and B. Saha, "Apache hadoop yarn: Yet another resource negotiator", in: *Proceedings of the 4th annual Symposium on Cloud Computing* (p. 5). ACM, 2013.
4. M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, and A. Ghodsi, "Apache Spark: A unified engine for big data processing", in: *Communications of the ACM*, 2016, pp.56-65.
5. R. Bryson, S. Kenwright, D. Cox, M. Ellsworth and D. Haimes, "Visually exploring gigabyte data sets in real time", *Commun. ACM*, 1999.
6. J.W. Tukey, *Exploratory data analysis*, 1977.
7. A. Koufakou, J. Secretan, J. Reeder, K. Cardona and M. Georgiopoulos, "Fast parallel outlier detection for categorical datasets using MapReduce", in: *Proceedings of IEEE International Joint Conference on Neural Networks*, 2008, pp. 3298–3304.
8. B. Liu, W. Fan, and T. Xiao, "A Fast Outlier Detection Method for Big Data", in: *Proceedings of Asian Simulation Conference*, 2013, pp. 379–384.
9. E.M. Knox and R.T. Ng, "Algorithms for mining distance based outliers in large datasets", in: *Proceedings of International Conference on Very Large Data Bases*, 1998, pp. 392–403.
10. F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces", in: *Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery*, 2002, pp. 15–27.
11. F. Angiulli and C. Pizzuti, "Outlier mining in large high-dimensional data sets", *IEEE Trans. Knowl. Data Eng.* 17 (2005), 203–215.
12. F. Angiulli, S. Basta, and C. Pizzuti, "Distance-based detection and prediction of outliers", *IEEE Trans. Knowl. Data Eng.* 18 (2006), 145–160.
13. S. Ramaswamy, R. Rastogi and K. Shim, "Efficient algorithms for mining outliers from large data sets", in: *ACM Sigmod Record* 29 (2000), 427–438.
14. M. Ester, H.P. Kriegel, J. Sander, X. Xu, and others, "A density-based algorithm for discovering clusters in large spatial databases with noise", in: *Proceedings of Kdd*, 1996, pp. 226–231.
15. M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander, "LOF: identifying density-based local outliers", in: *Proceedings of ACM sigmod record*, 2000, pp. 93–104.
16. S. Papadimitriou, H. Kitagawa, P.B. Gibbons and C. Faloutsos, "Loci: Fast outlier detection using the local correlation integral", in: *Proceedings of 19th International Conference on Data Engineering*, 2003, pp. 315–326.
17. Y. Yan, et al., "Distributed outlier detection using compressive sensing", in: *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2015, pp. 3–16.
18. F. Angiulli, S. Basta, S. Lodi and C. Sartori, "A distributed approach to detect outliers in very large datasets", *Euro-Par 2010-Parallel Processing* (2010), 329-340.
19. M. Dorigo, *Optimization, learning and natural algorithms*, Ph. D. Thesis, Politec. di Milano, Italy, 1992.
20. R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", in: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.
21. K.M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control", *IEEE Control Syst.* 22 (2002), 52–67.
22. X.L. Li, *A new intelligent optimization-artificial fish swarm algorithm*, Dr. thesis, Zhejiang Univ, Zhejiang, China, 2003.
23. K.N. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications", *Multiagent Grid Syst.* 2 (2006), 209–222.
24. D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", *J. Glob. Optim.* 39 (2007), 459–471.
25. X.S. Yang and S. Deb, "Cuckoo search via Lévy flights", in: *World Congress on Nature & Biologically Inspired Computing*, 2009 , pp. 210–214.
26. X.S. Yang, "A new metaheuristic bat-inspired algorithm", *Nat. inspired Coop. Strateg. Optim.* (2010), 65–74.

27. A.W. Mohemmed, M. Zhang and W.N. Browne, "Particle swarm optimisation for outlier detection", in: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010, pp. 83–84.
28. Y. Gigras, K. Gupta, and K. Choudhury, "A comparison between bat algorithm & cuckoo search for path planning", *J. Innovative Res. Comput. Commun. Eng*, 2015, pp. 4459-4466.
29. S. Hawkins, H. He, G. Williams and R. Baxter, "Outlier detection using replicator neural networks", in: *Proceedings of International Conference on Data Warehousing and Knowledge Discovery*, 2002, pp. 170–180.

AUTHORS PROFILE

Author-1 Photo

Adeel Shiraz Hashmi is a research scholar in Deparment of Computer Engineering of Jamia Millia Islamia, New Delhi, India. He has done his BTech. And M.Tech. from GGSIPU, Delhi. He has over 7 years of teaching experience. His areas of interest are Artificial Intelligence, Soft Computing, Machine Learning, Swarm Intelligence, Data Mining, and Big Data. He has multiple publications in Scopus indexed conferences and journals.

Author-2 Photo

Dr. Tanvir Ahmad is a Professor in Department of Computer Engineering, Jamia Millia Islamia. He has over 15 years of teaching experience and have multiple publications in reputed journals and conferences.