IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 21 October 2013; revised 15 April 2014; accepted 24 April 2014. Date of publication 11 June 2014;
date of current version 30 October 2014.

Digital Object Identifier 10.1109/TETC.2014.2330519

A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis

ADIL FAHAD[1,4], NAJLAA ALSHATRI[1], ZAHIR TARI[1], (Member, IEEE), ABDULLAH ALAMRI[1], IBRAHIM KHALIL[1], ALBERT Y. ZOMAYA[2], (Fellow, IEEE), SEBTI FOUFOU[3], AND ABDELAZIZ BOURAS[3]

[1]School of Computer Science and Information Technology,
Royal Melbourne Institute of Technology, Melbourne, VIC 3000, Australia
[2]Centre for Distributed and High Performance Computing, School of Information Technologies,
University of Sydney, Sydney, NSW 2006, Australia
[3]Department of Computer Science, College of Engineering, Qatar University, Doha 2713, Qatar
[4]Department of Computer Science, Al-Baha University, Al-Baha City 65431, Saudi Arabia
CORRESPONDING AUTHOR: A. FAHAD (alharthi.adil@gmail.com)

ABSTRACT Clustering algorithms have emerged as an alternative powerful meta-learning tool to accurately analyze the massive volume of data generated by modern applications. In particular, their main goal is to categorize data into clusters such that objects are grouped in the same cluster when they are similar according to specific metrics. There is a vast body of knowledge in the area of clustering and there has been attempts to analyze and categorize them for a larger number of applications. However, one of the major issues in using clustering algorithms for big data that causes confusion amongst practitioners is the lack of consensus in the definition of their properties as well as a lack of formal categorization. With the intention of alleviating these problems, this paper introduces concepts and algorithms related to clustering, a concise survey of existing (clustering) algorithms as well as providing a comparison, both from a theoretical and an empirical perspective. From a theoretical perspective, we developed a categorizing framework based on the main properties pointed out in previous studies. Empirically, we conducted extensive experiments where we compared the most representative algorithm from each of the categories using a large number of real (big) data sets. The effectiveness of the candidate clustering algorithms is measured through a number of internal and external validity metrics, stability, runtime, and scalability tests. In addition, we highlighted the set of clustering algorithms that are the best performing for big data.

INDEX TERMS Clustering algorithms, unsupervised learning, big data.

I. INTRODUCTION

IN THE current digital era, according to (as far) massive progress and development of the internet and online world technologies such as big and powerful data servers, we face a huge volume of information and data day by day from many different resources and services which were not available to humankind just a few decades ago. Massive quantities of data are produced by and about people, things, and their interactions. Diverse groups argue about the potential benefits and costs of analyzing information from Twitter, Google, Verizon, 23andMe, Facebook, Wikipedia, and every space where large groups of people leave digital traces and deposit data. This data comes from available different online resources and services which have been established to serve their customers. Services and resources like Sensor Networks, Cloud Storages, Social Networks and etc., produce big volume of data and also need to manage and reuse that data or some analytical aspects of the data. Although this massive volume of data can be really useful for people and corporations, it can be problematic as well. Therefore, a big volume of data or big data has its own deficiencies as well. They need big storages and this volume makes operations such as analytical operations, process operations, retrieval operations, very difficult and hugely time consuming. One way to overcome these difficult problems is to have big data clustered in a compact format that is still an informative

version of the entire data. Such clustering techniques aim to produce a good quality of clusters/summaries. Therefore, they would hugely benefit everyone from ordinary users to researchers and people in the corporate world, as they could provide an efficient tool to deal with large data such as critical systems (to detect cyber attacks).

The main goal of this paper is to provide readers with a proper analysis of the different classes of available clustering techniques for big data by experimentally comparing them on real big data. The paper does not refer to simulation tools. However, it specifically looks at the use and implementation of an efficient algorithm from each class. It also provides experimental results from a variety of big datasets. Some aspects need careful attention when dealing with big data, and this work will therefore help researchers as well as practitioners in selecting techniques and algorithms that are suitable for big data. *V*olume of data is the first and obvious important characteristic to deal with when clustering big data compared to conventional data clustering, as this requires substantial changes in the architecture of storage systems. The other important characteristic of big data is *V*elocity. This requirement leads to a high demand for online processing of data, where processing speed is required to deal with the data flows. *V*ariety is the third characteristic, where different data types, such as text, image, and video, are produced from various sources, such as sensors, mobile phones, etc. These three V (Volume, Velocity, and Variety) are the core characteristics of big data which must be taken into account when selecting appropriate clustering techniques.

Despite a vast number of surveys for clustering algorithms available in the literature [1], [2], [7], and [38] for various domains (such as machine learning, data mining, information retrieval, pattern recognition, bio-informatics and semantic ontology), it is difficult for users to decide a *priori* which algorithm would be the *most* appropriate for a given big dataset. This is because of some of the limitations in existing surveys: (i) the characteristics of the algorithms are not well studied; (ii) the field has produced many new algorithms, which were not considered in these surveys; and (iii) no rigorous empirical analysis has been carried out to ascertain the benefit of one algorithm over another. Motivated by these reasons, this paper attempts to review the field of clustering algorithms and achieve the following objectives:

- To propose a categorizing framework that systematically groups a collection of existing clustering algorithms into categories and compares their advantages and drawbacks from a theoretical point of view.
- To present a complete taxonomy of the clustering evaluation measurements to be used for empirical study.
- To make an empirical study analyzing the most representative algorithm of each category with respect to both theoretical and empirical perspectives.

Therefore, the proposed survey presents a taxonomy of clustering algorithms and proposes a categorizing framework that covers major factors in the selection of a suitable algorithm for big data. It further conducts experiments involving the most representative clustering algorithm of each category, a large number of evaluation metrics and 10 traffic datasets. The rest of this paper is organized as follows. Section II provides a review of clustering algorithms categories. Section III describe the proposed criteria and properties for the categorizing framework. In Section IV, we group and compare different clustering algorithms based on the proposed categorizing framework. Section V introduces the taxonomy of clustering evaluation measurements, describes the experimental framework and summarises the experimental results. In Section VI, we conclude the paper and discuss future research.

II. CLUSTERING ALGORITHM CATEGORIES

As there are so many clustering algorithms, this section introduces a categorizing framework that groups the various clustering algorithms found in the literature into distinct categories. The proposed categorization framework is developed from an algorithm designer's perspective that focuses on the technical details of the general procedures of the clustering process. Accordingly, the processes of different clustering algorithms can be broadly classified follows:

- **Partitioning-based**: In such algorithms, all clusters are determined promptly. Initial groups are specified and reallocated towards a union. In other words, the partitioning algorithms divide data objects into a number of partitions, where each partition represents a cluster. These clusters should fulfil the following requirements: (1) each group must contain at least one object, and (2) each object must belong to exactly one group. In the K-means algorithm, for instance, a center is the average of all points and coordinates representing the arithmetic mean. In the K-medoids algorithm, objects which are near the center represent the clusters. There are many other partitioning algorithms such as K-modes, PAM, CLARA, CLARANS and FCM.
- **Hierarchical-based**: Data are organized in a hierarchical manner depending on the medium of proximity. Proximities are obtained by the intermediate nodes. A dendrogram represents the datasets, where individual data is presented by leaf nodes. The initial cluster gradually divides into several clusters as the hierarchy continues. Hierarchical clustering methods can be agglomerative (bottom-up) or divisive (top-down). An agglomerative clustering starts with one object for each cluster and recursively merges two or more of the most appropriate clusters. A divisive clustering starts with the dataset as one cluster and recursively splits the most appropriate cluster. The process continues until a stopping criterion is reached (frequently, the requested number k of clusters). The hierarchical method has a major drawback though, which relates to the fact that once a step (merge or split) is performed, this cannot be undone. BIRCH, CURE, ROCK and Chameleon are some of the well-known algorithms of this category.
- **Density-based**: Here, data objects are separated based on their regions of density, connectivity and boundary.

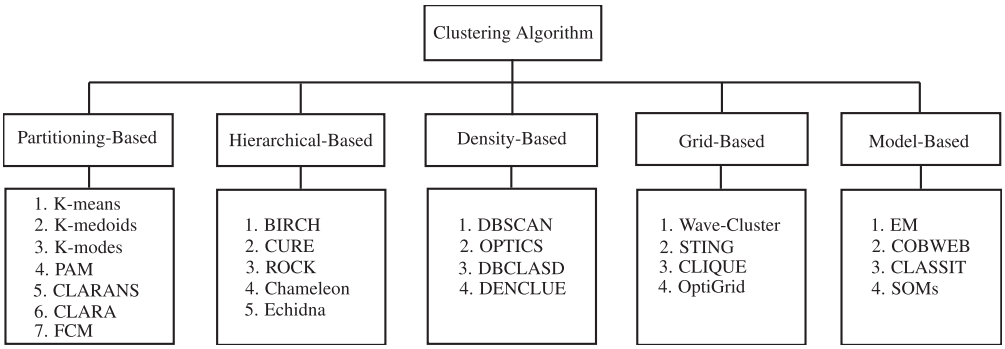

FIGURE 1. An overview of clustering taxonomy.

They are closely related to point-nearest neighbours. A cluster, defined as a connected dense component, grows in any direction that density leads to. Therefore, density-based algorithms are capable of discovering clusters of arbitrary shapes. Also, this provides a natural protection against outliers. Thus the overall density of a point is analyzed to determine the functions of datasets that influence a particular data point. DBSCAN, OPTICS, DBCLASD and DENCLUE are algorithms that use such a method to filter out noise (ouliers) and discover clusters of arbitrary shape.

- **Grid-based**: The space of the data objects is divided into grids. The main advantage of this approach is its fast processing time, because it goes through the dataset once to compute the statistical values for the grids. The accumulated grid-data make grid-based clustering techniques independent of the number of data objects that employ a uniform grid to collect regional statistical data, and then perform the clustering on the grid, instead of the database directly. The performance of a grid-based method depends on the size of the grid, which is usually much less than the size of the database. However, for highly irregular data distributions, using a single uniform grid may not be sufficient to obtain the required clustering quality or fulfill the time requirement. Wave-Cluster and STING are typical examples of this category.
- **Model-based**: Such a method optimizes the fit between the given data and some (predefined) mathematical model. It is based on the assumption that the data is generated by a mixture of underlying probability distributions. Also, it leads to a way of automatically determining the number of clusters based on standard statistics, taking noise (outliers) into account and thus yielding a robust clustering method. There are two major approaches that are based on the model-based method: *statistical* and *neural network* approaches. MCLUST is probably the best-known model-based algorithm, but there are other good algorithms, such as EM (which uses a mixture density model), conceptual clustering (such as COBWEB), and neural network approaches (such as self-organizing feature maps). The *statistical* approach uses probability measures in determining the concepts or clusters. Probabilistic descriptions are typically used to represent each derived concept. The *neural network* approach uses a set of connected input/output units, where each connection has a weight associated with it. Neural networks have several properties that make them popular for clustering. First, neural networks are inherently parallel and distributed processing architectures. Second, neural networks learn by adjusting their interconnection weights so as to best fit the data. This allows them to normalize or prototype. Patterns act as features (or attributes) extractors for the various clusters. Third, neural networks process numerical vectors and require object patterns to be represented by quantitative features only. Many clustering tasks handle only numerical data or can transform their data into quantitative features if needed. The neural network approach to clustering tends to represent each cluster as an exemplar. An exemplar acts as a prototype of the cluster and does not necessarily have to correspond to a particular object. New objects can be assigned to the cluster whose exemplar is the most similar, based on some distance measure.

Figure 1 provides an overview of clustering algorithm taxonomy following the five classes of categorization described above.

III. CRITERION TO BENCHMARK CLUSTERING METHODS

When evaluating clustering methods for big data, specific criteria need to be used to evaluate the relative strengths and weaknesses of every algorithm with respect to the three-dimensional properties of big data, including *Volume*, *Velocity*, and *Variety*. In this section, we define such properties and compiled the key criterion of each property.

- **Volume** refers to the ability of a clustering algorithm to deal with a large amount of data. To guide the selection of a suitable clustering algorithm with respect to the *Volume* property, the following criteria are considered: (i) size of the dataset, (ii) handling high dimensionality and (iii) handling outliers/ noisy data.

TABLE 1. Categorization of clustering algorithms with respect to big data proprieties and other criteria described in Section III.

Categories	Abb. name	Volume			Variety		Velocity	Other criterion
		Size of Dataset	Handling High Dimensionality	Handling Noisy Data	Type of Dataset	Clusters Shape	complexity of Algorithm	Input Parameter
Partitional algorithms	K-Means [25]	Large	No	No	Numerical	Non-convex	O(nkd)	1
	K-modes [19]	Large	Yes	No	Categorical	Non-convex	O(n)	1
	K-medoids [33]	Small	Yes	Yes	Categorical	Non-convex	$O(n^2dt)$	1
	PAM [31]	Small	No	No	Numerical	Non-convex	$O(k(n-k)^2)$	1
	CLARA [23]	Large	No	No	Numerical	Non-convex	$O(k(40+k)^2+k(n-k))$	1
	CLARANS [32]	Large	No	No	Numerical	Non-convex	$O(kn^2)$	2
	FCM [6]	Large	No	No	Numerical	Non-convex	O(n)	1
Hierarchical algorithms	BIRCH [40]	Large	No	No	Numerical	Non-convex	O(n)	2
	CURE [14]	Large	Yes	Yes	Numerical	Arbitrary	$O(n^2 \log n)$	2
	ROCK [15]	Large	No	No	Categorical and Numerical	Arbitrary	$O(n^2+nmmma+n^2logn)$	1
	Chameleon [22]	Large	Yes	No	All type of data	Arbitrary	$O(n^2)$	3
	ECHIDNA [26]	Large	No	No	Multivariate Data	Non-convex	$O(N * B(1 + \log_B m))$	2
Density-based algorithms	DBSCAN [9]	Large	No	No	Numerical	Arbitrary	O(nlogn) If a spatial index is used Otherwise, it is $O(n^2)$.	2
	OPTICS [5]	Large	No	Yes	Numerical	Arbitrary	O(nlogn)	2
	DBCLASD [39]	Large	No	Yes	Numerical	Arbitrary	$O(3n^2)$	No
	DENCLUE [17]	Large	Yes	Yes	Numerical	Arbitrary	$O(\log\|D\|)$	2
Grid- based algorithms	Wave-Cluster [34]	Large	No	Yes	Special data	Arbitrary	O(n)	3
	STING [37]	Large	No	Yes	Special data	Arbitrary	O(k)	1
	CLIQUE [21]	Large	Yes	No	Numerical	Arbitrary	O(Ck + mk)	2
	OptiGrid [18]	Large	Yes	Yes	Special data	Arbitrary	Between O(nd) and O(nd log n)	3
Model- based algorithms	EM [8]	Large	Yes	No	Special data	Non-convex	O(knp)	3
	COBWEB [12]	Small	No	No	Numerical	Non-convex	$O(n^2)$	1
	CLASSIT [13]	Small	No	No	Numerical	Non-convex	$O(n^2)$	1
	SOMs [24]	Small	Yes	No	Multivariate Data	Non-convex	$O(n^2m)$	2

- **Variety** refers to the ability of a clustering algorithm to handle different types of data (numerical, categorical and hierarchical). To guide the selection of a suitable clustering algorithm with respect to the *Variety* property, the following criteria are considered: (i) type of dataset and (ii) clusters shape.
- **Velocity** refers to the speed of a clustering algorithm on big data. To guide the selection of a suitable clustering algorithm with respect to the *Velocity* property, the following criteria are considered: (i) complexity of algorithm and (ii) the run time performance.

In what follows, we explain in detail the corresponding criterion of each property of big data:

1) *Type Of Dataset:* Most of the traditional clustering algorithms are designed to focus either on numeric data or on categorical data. The collected data in the real world often contain both numeric and categorical attributes. It is difficult for applying traditional clustering algorithm directly into these kinds of data. Clustering algorithms work effectively either on purely numeric data or on purely categorical data; most of them perform poorly on mixed categorical and numerical data types.
2) *Size Of Dataset:* The size of the dataset has a major effect on the clustering quality. Some clustering methods are more efficient clustering methods than others when the data size is small, and vice versa.
3) *Input Parameter:* A desirable feature for ''practical'' clustering is the one that has fewer parameters, since a large number of parameters may affect cluster quality because they will depend on the values of the parameters.
4) *Handling Outliers/Noisy Data:* A successful algorithm will often be able to handle outlier/noisy data because of the fact that the data in most of the real applications are not pure. Also, noise makes it difficult for an algorithm to cluster an object into a suitable cluster. This therefore affects the results provided by the algorithm.
5) *Time Complexity:* Most of the clustering methods must be used several times to improve the clustering quality. Therefore if the process takes too long, then it can become impractical for applications that handle big data.
6) *Stability:* One of the important features for any clustering algorithm is the ability to generate the same partition of the data irrespective of the order in which the patterns are presented to the algorithm.
7) *Handling High Dimensionality:* This is particularly important feature in cluster analysis because many applications require the analysis of objects containing a large number of features (dimensions). For example, text documents may contain thousands of terms or keywords as features. It is challenging due to the curse of dimensionality. Many dimensions may not be relevant. As the number of dimensions increases, the data become increasingly sparse, so that the distance measurement between pairs of points becomes meaningless and the average density of points anywhere in the data is likely to be low.
8) *Cluster Shape:* A good clustering algorithm should be able to handle real data and their wide variety of data types, which will produce clusters of arbitrary shape.

IV. CANDIDATE CLUSTERING ALGORITHMS

This section aims to find the *good* candidate clustering algorithms for big data. By *good*, we refer to those algorithms that satisfy most of the criterion listed in Section III. Table 1 provides a summary of the evaluation we performed on the various methods described in Section II based on the described criteria. After this evaluation, the next step is to select the most appropriate clustering algorithm from each category based on the proposed criteria, so to benchmark them for big data. In this way, the best algorithm is selected from each method, and these (selected algorithms) will be properly evaluated. This process produced the following selections:

FCM [6], BIRCH [40], DENCLUE [17], OptiGird [18] and EM [8].

This section discusses each of the selected algorithms in detail, showing how it works, its strengths and weakness, as well as the input parameters it takes.

A. FUZZY-CMEANS (FCM)

FCM [6] is a representative algorithm of fuzzy clustering which is based on K-means concepts to partition dataset into clusters. The FCM algorithm is a ''soft'' clustering method in which the objects are assigned to the clusters with a degree of belief. Hence, an object may belong to more than one cluster with different degrees of belief. It attempts to find the most characteristic point in each cluster, named as the *centre* of one cluster; then it computes the membership degree for each object in the clusters. The fuzzy c-means algorithm minimizes intra-cluster variance as well. However, it inherits the problems of K-means, as the minimum is just a local one and the final clusters depend on the initial choice of weights.

FCM algorithm follows the same principle of K-means algorithm, i.e. it iteratively searches the cluster centers and updates the memberships of objects. The main difference is that, instead of making a hard decision about which cluster the pixel should belong to, it assigns a object a value ranging from 0 to 1 to measure the *likelihood* with which the object belongs to that cluster. A fuzzy rule states that the sum of the membership value of a pixel to all clusters must be 1. The higher the membership value, the more likely a pixel will belong to that cluster. The FCM clustering is obtained by minimizing the objective function shown in Equation 1:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{c} \mu_{ik}^{m} |p_i - v_k|^2 \tag{1}$$

where J is the objective function, n is the number of objects, c is the number of defined clusters, μ_{ik} is the likelihood values by assiging the object i to the cluster k, m is a fuzziness factor (a value 1), and $|p_i - v_k|$ is the Euclidean distance between the i-th object p_i and the k-th cluster centre v_k defined by Equation 2:

$$|p_i - v_k| = \sqrt{\sum_{i=1}^{n} (p_i - v_k)} \tag{2}$$

The centroid of the k^{th} cluster is updated using Equation 3:

$$v_k = \frac{\sum_{i=1}^{n} \mu_{ik}^{m} p_i}{\sum_{i=1}^{n} \mu_{ik}^{m}} \tag{3}$$

The fuzzy membership table is computed using the original Equation 3:

$$\mu_{ik} = \frac{1}{\sum_{l=1}^{c} \left(\frac{|p_i - v_k|}{|p_i - v_l|} \right)^{\frac{2}{m-1}}} \tag{4}$$

This algorithm has been extended for clustering a RGB color images, where the distance computation given in Equation 2 is modified as follows:

$$|p_i - v_k| = \sqrt{\sum_{i=1}^{n} (p_{iR} - v_{kR})^2 + (p_{iG} - v_{kG})^2 + (p_{iB} - v_{kB})^2} \tag{5}$$

As mentioned earlier, this has an iterative process (see FCM pseudo-code).

FCM pseudo-code:

Input: Given the dataset, set the desire number of clusters c, the fuzzy parameter m (a constant > 1), and the stopping condition, initialize the fuzzy partition matrix, and set *stop* = *false*.

Step 1. Do:

Step 2. Calculate the cluster centroids and the objective value J.

Step 3. Compute the membership values stored in the matrix.

Step 4. If the value of J between consecutive iterations is less than the stopping condition, then *stop* = *true*.

Step 5. **While** (!*stop*)

Output: A list of c cluster centres and a partition matrix are produced.

B. BIRCH

BIRCH algorithm [40] builds a dendrogram known as a clustering feature tree (CF tree). The CF tree can be built by scanning the dataset in an incremental and dynamic way. Thus, it does not need the whole dataset in advance. It has two main phases: the database is first scanned to build an in-memory tree, and then the algorithm is applied to cluster the leaf nodes. CF-tree is a height-balanced tree which is based on two parameters: branching factor B and threshold T. The CF tree is built while scanning the data. When a data point is encountered, the CF tree is traversed, starting from the root and choosing the closest node at each level. If the closest leaf cluster for the current data point is finally identified, a test is performed to see whether the data point belongs to the candidate cluster or not. If not, a new cluster is created with a diameter greater than the given T. BIRCH can typically find a good clustering with a single scan of the dataset and improve the quality further with a few additional scans. It can also handle noise effectively. However, BIRCH may not work well when clusters are not spherical because it uses the concept of radius or diameter to control the boundary of a cluster. In addition, it is order-sensitive and may generate different clusters for different orders of the same input data. The details of the algorithm are given below.

C. DENCLUE

The DENCLUE algorithm [17] analytically models the cluster distribution according to the sum of influence functions of all of the data points. The influence function can be seen as a function that describes the impact of a data point within its neighbourhood. Then density attractors can be identified as clusters. Density attractors are local maxima of the overall density function. In this algorithm, clusters of arbitrary shape can be easily described by a simple equation with kernel

BIRCH pseudo-code:

Input: The dataset, threshold T, the maximum diameter (or radius) of a cluster R, and the branching factor B

Step 1. (Load data into memory) An initial in-memory CF-tree is constructed with one scan of the data. Subsequent phases become fast, accurate and less order sensitive.

Step 2. (Condense data) Rebuild the CF-tree with a larger T.

Step 3. (Global clustering) Use the existing clustering algorithm on CF leaves.

Step 4. (Cluster refining) Do additional passes over the dataset and reassign data points to the closest centroid from step #3.

Output:. Compute CF points, where CF = (# of points in a cluster N, linear sum of the points in the cluster LS, the square sum of N data SS).

density functions. Even though DENCLUE requires a careful selection of its input parameters (i.e. σ and ξ), since such parameters may influence the quality of the clustering results, it has several advantages in comparison to other clustering algorithms [16]: (i) it has a solid mathematical foundation and generalized other clustering methods, such as partitional and hierarchical; (ii) it has good clustering properties for datasets with large amount of noise; (iii) it allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional datasets; and (iv) it uses grid cells and only keeps information about the cells that actually contain points. It manages these cells in a tree-based access structure and thus it is significant faster than some influential algorithms, such as DBSCAN. All of these properties make DENCLUE able to produce good clusters in datasets with a large amount of noise.

The details of this algorithm are given below.

DENCLUE pseudo-code:

Input: The dataset, Cluster radius, and Minimum number of objects

Step 1. Take dataset in the grid whose each side is of 2σ.

Step 2. Find highly dense cells, i.e. find out the mean of highly populated cells.

Step 3. If d (mean(c_1), mean(c_2)) $< 4a$, then the two cubes are connected.

Step 4. Now highly populated cells or cubes that are connected to highly populated cells will be considered in determining clusters.

Step 5. Find Density Attractors using a Hill Climbing procedure.

Step 6. Randomly pick point r.

Step 7. Compute the local 4 σ density.

Step 8. Pick another point (r+1) close to the previous computed density.

Step 9. If den(r) $<$ den(r+1) climb, then put points within (σ /2) of the path into the cluster.

Step 10. Connect the density attractor based cluster.

Output: Assignment of data values to clusters.

D. OPTIMAL GRID (OPTIGRID)

OptiGrid algorithm [18] is designed to obtain an optimal grid partitioning. This is achieved by constructing the best cutting hyperplanes through a set of selected projections. These projections are then used to find the optimal cutting planes. Each cutting plane is selected to have minimal point density and to separate the dense region into two half spaces. After each step of a multi-dimensional grid construction defined by the best cutting planes, OptiGrid finds the clusters using the density function. The algorithm is then applied recursively to the clusters. In each round of recursion, OptiGrid only maintains data objects in the dense grids from the previous round of recursion. This method is very efficient for clustering large high-dimensional databases. However, it may perform poorly in locating clusters embedded in a low-dimensional subspace of a very high-dimensional database, because its recursive method only reduces the dimensions by one at every step. In addition, it suffers from sensitivity to parameter choice and does not efficiently handle grid sizes that exceed the available memory [12]. Moreover, OptiGrid requires very careful selection of the projections, density estimate, and determination of what constitutes a best or optimal cutting plane from users. The difficulty of this is only determined on a case-by-case basis on the data being studied.

OptiGrid pseudo-code:

Input: The dataset (x), a set of contracting projections $P = \{P_0, P_1, \cdots, P_k\}$, a list of cutting planes BEST CUT $\Leftarrow \Phi$, and CUT $\Leftarrow \Phi$;

Step 1. For i=0,...,k, do

Step 2. CUT best local cuts $P_i(D)$, CUT SCORE $\Leftarrow$ Score best local cuts $P_i(D)$

Step 3. Insert all the cutting planes with a score $\geq$ min cut score into BEST CUT;

Step 4. Select the q cutting planes of the highest score from BEST CUT and construct a multidimensional grid G using the q cutting planes;

Step 5. Insert all data points in D into G and determine the highly populated grid cells in G; add these cells to the set of clusters C;

Refine C: For all clusters C_i in C, perform the same process with dataset C_i;

Output: Assignment of data values to clusters.

E. EXPECTATION-MAXIMIZATION (EM)

EM algorithm [8] is designed to estimate the maximum likelihood parameters of a statistical model in many situations, such as the one where the equations cannot be solved directly. EM algorithm iteratively approximates the unknown model parameters with two steps: the E step and the M step. In the E step (expectation), the current model parameter values are used to evaluate the posterior distribution of the latent variables. Then the objects are fractionally assigned to each cluster based on this posterior distribution. In the M step (maximization), the fractional assignment is given by re-estimating the model parameters with the maximum likelihood rule. The EM algorithm is guaranteed to find a local maximum for the model parameters estimate. The major disadvantages for EM algorithm are: the requirement of a non-singular covariance matrix, the sensitivity to the selection of initial parameters, the possibility of convergence to a local optimum, and the slow convergence rate. Moreover, there would be a decreased precision of the EM algorithm within a finite number of steps [28].

The details of the EM algorithm are given below.

EM pseudo-code:

Input: The dataset (x), the total number of clusters (M), the accepted error for convergence (e) and the maximum number of iterations

E-step: Compute the expectation of the complete data log-likelihood.

$$Q\left(\theta,\theta^{T}\right)=E\left[\log p\left(x^{g},x^{m}|\theta\right)x^{g},\theta^{T}\right] \quad (6)$$

M-step: Select a new parameter estimate that maximizes the Q-function,

$$\theta^{t+1}=\arg\max_{\theta} Q\left(\theta,\theta^{T}\right) \quad (7)$$

Iteration: increment $t=t+1$; repeat steps 2 and 3 until the convergence condition is satisfied.

Output: A series of parameter estimates $\{\theta^{0},\theta^{1},...,\theta^{T}\}$, which represents the achievement of the convergence criterion.

V. EXPERIMENTAL EVALUATION ON REAL DATA

In some cases, it is not sufficient to decide the most suitable clustering algorithm for big data based only on the theoretical point of view. Thus, the main focus of this section is to investigate the behaviour of the algorithms selected in Section IV from an empirical perspective.

In what follows, we described the traffic datasets used for this experimental study in Section V-A. Section V-B provides details of the experimental set up and Section V-C presents a complete survey for performance metrics to be used to experimentally investigate the relative strength and weakness of each algorithm. Finally, the collected results and a comprehensive analysis study are given in Section V-D.

A. DATA SET

To compare the advantages of the candidate clustering algorithms, eight simulated datasets are used in the experiments including: Multi-hop Outdoor Real Data (MHORD) [36], Multi-hop Indoor Real Data (MHIRD) [36], Single-hop Outdoor Real Data (SHORD) [36], Single-hop Indoor Real Data (SHIRD) [36], simulated *spoofing* attack for SCADA system (detonated as SPFDS) [3], [4], simulated denial of service attack *DOS* for SCADA system (detonated as DOSDS) [3], [4], simulated of both *spoofing* and attacks for SCADA system (detonated as SPDOS) [3], [4], and the operational state *water treatment plant* (WTP). We experimented also with two other publicly available datasets, namely DARPA [35] and *internet traffic data* (ITD) [29]. These two datasets have become a benchmark for many studies since the work of Andrew et al. [30]. Table 2 summarizes the proportion of normal and anomalous flows, the number of attributes and the number of classes for each dataset. This paper does not collect the descriptions of the datasets due to space restrictions. Thus, we recommend that readers consult the original references [3], [4], [10], [11], [36] for more complete details about the characteristics of the datasets.

TABLE 2. Data sets used in the experiments.

dataset	# instances	# attributes	# classes
MHIRD	699	10	2
MHORD	2500	3	2
SPFDS	100500	15	2
DOSDS	400350	15	2
SPDOS	290007	15	3
SHIRD	1800	4	2
SHORD	400	4	2
ITD	377,526	149	12
WTP	512	39	2
DARPA	1000,000	42	5

B. EXPERIMENTAL SET UP

Algorithm 1 shows the experimental procedures used to evaluate the five candidate clustering algorithms. In particular, a cross validation strategy is used to make the best use of the traffic data and to obtain accurate and stable results. For each dataset, all instances are randomised and divided into two subsets as training and testing sets. Consequently, we evaluate the performance of each clustering algorithm by building a model using training set and measuring and using the testing set to evaluate the constructed model. To assure that the five candidate clustering algorithms are not exhibiting an order effect, the result of each clustering is averaged over 10 runs on each datasets. The five candidate clustering algorithms studied here employ different parameters. However, the experimental evaluation does not correspond to exhaustive search for the best parameters settings for each algorithm. Given the datasets at hand, the main objective is to use a default configuration for set the parameters of the clustering algorithms. In general, finding an optimal number of clusters is an ill-posed problem of crucial relevance in clustering analysis [20]. Thus, we have chosen the number of clusters with respect to the number of unique labels in each dataset. However, the true number of clusters may not be the optimal number for which a particular clustering algorithm will determine, to its best potential, the structure in the data.

Following the procedure and the pseudo-code of each clustering algorithm discussed in Section IV, the candidate clustering algorithms were implemented in Matlab 2013a. Our experiments were performed on a 64-bit Windows-based system with an Intel core (i7), 2.80 GHz processor machine with 8 Gbytes of RAM.

C. VALIDITY METRICS

In response to the growing necessity for an objective method of comparing clustering algorithms, we present a complete survey of performance metrics that covers all the properties and issues related to the experimental study of clustering. In particular, this survey of performance metrics will allow researchers to compare different algorithms in an objective way, to characterize their advantages and drawbacks in order to choose a clustering from an empirical point of view.

The survey covers three measurements: *Validity evaluation*, *stability of the results* and *runtime performance*.

Algorithm 1: Experimental Procedure

Input:

Parameter $N := 10$; $M := 100$;
Clustering Algorithms Cls:= $\{cl_1, cl_2, \cdots, cl_m\}$;
DATA= $\{D_1, D_2, \cdots, D_n\}$;
Output:

Validity & Stability;
foreach *Clustering*$_i \in [1, Cls]$ **do**
  foreach $D_i \in DATA$ **do**
    foreach *times* $\in [1, M]$ **do**
      randomise instance-order for D_i;
      generate N *bins from the randomised* D_i;
      foreach *fold* $\in [1, N]$ **do**
        $Test_{Data} = bin[fold]$;
        $Train_{Data} = data - Test_{Data}$;
        $Train'_{Data} =$ *select Subset from* $Train_{Data}$;
        $Test'_{Data} =$ *select Subset from* $Test_{Data}$;
        Cls_{ASGN} =*TestModel*($Test'_{Data}$);
        Validity=`CompuValidaty`(Cls_{ASGN}, $Test_{lbs}$);
        $Assignment_i^{cls} = assignment_i^{cls} \cup Cls_{ASGN}$;
  Stability=`ComputeStability` ($Assignment_i^{cls}$);

1) *Validity evaluation.* Unsupervised learning techniques required different evaluation criteria than supervised learning techniques. In this section, we briefly summarize the criteria used for performance evaluation according to *internal* and *external* validation indices. The former evaluation criteria is to evaluate the goodness of a data partition using quantities and feature inherited from the datasets, this includes *Compactness* (CP) and *Dunn Validity Index* (DVI). The latter evaluation criteria is similar to the process of *cross-validation* that is used in evaluating supervised learning techniques. Such evaluation criteria include *Classification Accuracy* (CA), *Adjusted Rand Index* (ARI) and *Normalized Mutual Information* (NMI). Given a dataset whose class labels are known, it is possible to assess how accurately a clustering technique partitions the data relative to their correct class labels. Note, some of clustering algorithms do not have centroids, and therefore the *internal* indices are not applicable to such an algorithms (e.g. OptiGrid and DENCLUE). To address such as issue we get the centroid of a cluster by using the measure in [26] and [27] and Euclidean distance metric.
The following notation is used: X is the dataset formed by x_i flows; Ω is the set of flows that have been grouped in a cluster; and W is the set of w_j centroids of the clusters in Ω. We will call node to each of the k elements of the clustering method.
   - **Compactness (CP)**. It is one of the commonly measurements used to validate clusters by employing only the information inherent to the dataset. Thus, a good clustering will create clusters with instances that are similar or closest to one another. More precisely, CP measures the average distance between every pair of data points as follows:

$$\overline{CP}_i = \frac{1}{|\Omega_i|} \sum_{x_i \in \Omega_i} \|x_i - w_i\| \tag{8}$$

where Ω is the set of instances (x_i) that have been grouped into a cluster and W is the set of w_i centroids of clusters in Ω. As a global measure of compactness, the average of all clusters is calculated as follows:

$$\overline{CP} = \frac{1}{K} \sum_{k=1}^{K} \overline{CP}_k,$$

where K denotes the number of clusters in the clustering result. Ideally, the members of each cluster should be as close to each other as possible. Therefore, a lower value of CP indicates better and more compact clusters.
   - **Separation (SP)**. This measure quantifies the degree of separation between individual clusters. It measures the mean *Euclidean* distance between cluster centroids as follows:

$$\overline{SP} = \frac{2}{k^2 - k} \sum_{i=1}^{k} \sum_{j=i+1}^{k} \|w_i - w_j\|_2 \tag{9}$$

where an $\overline{SP}$ close to 0 is an indication of closer clusters.
   - **Davies-Bouldin Index (DB)**. This index can identify cluster overlap by measuring the ratio of the sum of within-cluster scatters to between-cluster separations. It is defined as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\overline{C_i} + \overline{C_j}}{\|w_i - w_j\|_2} \right) \tag{10}$$

where a DB close to 0 indicates that the clusters are compact and far from each other.
   - **Dunn Validity Index (DVI)**. The DVI index quantifies not only the degree of compactness of clusters but also the degree of separation between individual clusters. DVI measures intercluster distances (separation) over intracluster distances (compactness). For a given number of clusters K, the definition of such an index is given by the following equation:

$$DVI = \frac{\min\limits_{0<m\neq n<K} \left\{ \min\limits_{\substack{\forall x_i \in \Omega_m \\ \forall x_j \in \Omega_n}} \{\|x_i - x_j\|\} \right\}}{\max\limits_{0<m\leq K} \max\limits_{\forall x_i, x_j \in \Omega_m} \{\|x_i - x_j\|\}} \tag{11}$$

If a dataset containing compact and well-separated clusters, the distance between the clusters are usually large and their diameters are expected to be small. Thus, a larger DVI value indicates compact and well-separated clusters.

- **Cluster Accuracy (CA)**. CA measures the percentage of correctly classified data points in the clustering solution compared to pre-defined class labels. The CA is defined as:

$$CA = \sum_{i=1}^{K} \frac{max(C_i|L_i)}{|\Omega|} \tag{12}$$

where C_i is the set of instances in the ith cluster; L_i is the class labels for all instances in the ith cluster, and $max(C_i|L_i)$ is the number of instances with the majority label in the ith cluster (e.g. if label l appears in the ith cluster more often than any other label, then $max(C_i|L_i)$ is the number of instances in C_i with the label l).

- **Adjusted Rand index (ARI)**. ARI takes into account the number of instances that exist in the same cluster and different clusters. The expected value of such a validation measure is not zero when comparing partitions.

$$ARI = \frac{n_{11} + n_{00}}{n_{00} + n_{01} + n_{10} + n_{11}} = \frac{n_{11} + n_{00}}{\binom{n}{2}} \tag{13}$$

where:

  - n_{11}: Number of pairs of instances that are in the same cluster in both.
  - n_{00}: Number of pairs of instances that are in different clusters.
  - n_{10}: Number of pairs of instances that are in the same cluster in A, but in different clusters in B.
  - n_{01}: Number of pairs of instances that are in different clusters in A, but in the same cluster in B.

  The value of ARI lies between 0 and 1, and the higher value indicates that all data instances are clustered correctly and the cluster contains only pure instances.

- **Normalized Mutual Information (NMI)**. This is one of the common external clustering validation metrics that estimate the quality of the clustering with respect to a given class labeling of the data. More formally, NMI can effectively measure the amount of statistical information shared by random variables representing the cluster assignments and the pre-defined label assignments of the instances. Thus, NMI is calculated as follows:

$$NMI = \frac{\sum d_{h,l} \log\left(\frac{|\Omega|.d_{h,l}}{d_h c_l}\right)}{\sqrt{\left(\sum_h d_h \log\left(\frac{d_h}{d}\right)\right)\left(\sum_l c_l log\left(\frac{c_l}{d}\right)\right)}} \tag{14}$$

where d_h is the number of flows in class h, c_l is the number of flows in cluster l and $d_{h,l}$ is the number of flows in class h as well as in cluster l. The NMI value is 1 when the clustering solution perfectly matches the pre-defined label assignments and close to 0 for a low matching.

2) *Stability of the results*. Since most clustering algorithms rely on a random component, stability of the results across different runs is considered to be an asset for an algorithm. Our experimental study examined the stability of the candidate clustering algorithms. In doing so, we consider a pairwise approach to measuring the stability of the candidate clusterers. In particular, the match between each of the $n(n-1)/2$ runs of a single cluster is calculated and the stability index is obtained as the averaged degree of match across different runs. Let $S_r(R_i, R_j)$ be the degree of match between runs R_i and R_j. The cluster pairwise stability index S_k is:

$$S_k = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} S_r(R_i, R_j). \tag{15}$$

where:

$$S_r(R_i, R_j) = \begin{cases} 1 & \text{if } R_i(x_i) = R_j(x_j) \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Clearly, it can be seen that $S_k(C)$ is the average stability measure over all pairs of clustering across different runs. It takes values from [0, 1], with 0 indicating the results between all pairs of R_i, R_j are totally different and 1 indicating that the results of all pairs across different runs are identical.

3) *Time requirements*. A key motivation for selecting the candidate clustering algorithms is to deal with big data. Therefore, if a clustering algorithm takes too long, it will be impractical for big data.

D. EXPERIMENTAL RESULTS AND COMPARISON

First of all, this section presents a comparison of the clustering outputs with respect to both the external and internal validity measures. After that, the candidate clustering algorithms are analyzed from the perspective of stability, run-time performance and scalability.

E. EVALUATING VALIDITY

The aim of this test is to determine how accurately a clustering algorithm can group traffic records from two different populations. Assessing the validity of clustering algorithms based on a single measure only can lead to misleading conclusions. Thus, we have conducted four types of external tests: Cluster Accuracy (CA), Adjusted Rand index (ARI), Rand index (RI) and Normalized Mutual Information (NMI). Such measurements allow us to exploit prior knowledge of known data partitions and cluster labels of the data. Note the class labels of instances (e.g attack/normal) are used

TABLE 3. External validity results for the candidate clustering algorithms.

Measures	Cls. Algorithms	MHIRD	MHORD	SPFDS	DOSDS	SPDOS	SHIRD	SHORD	ITD	WTP	DARPA
CA	DENCLUE	67.904	69.729	68.042	61.864	66.731	63.149	71.265	51.909	64.350	70.460
	OptiGrid	71.914	71.105	72.045	72.191	70.632	72.234	37.216	40.953	51.953	62.215
	FCM	75.387	73.271	74.682	74.222	72.873	74.723	75.553	59.974	66.435	73.114
	EM	82.512	81.940	82.786	82.919	82.114	79.450	82.023	65.035	72.085	80.685
	BIRCH	71.310	69.763	69.553	69.930	69.716	68.351	70.365	24.510	59.343	77.343
ARI	DENCLUE	57.772	45.248	41.535	39.822	44.510	35.081	46.267	35.663	47.665	60.665
	OptiGrid	35.894	30.297	32.140	29.402	29.970	32.598	29.956	55.824	32.137	52.137
	FCM	58.439	53.418	61.489	64.181	57.038	58.776	59.168	38.567	49.926	58.534
	EM	70.047	69.481	73.914	70.655	79.205	67.864	66.731	44.403	55.343	65.725
	BIRCH	52.424	44.011	52.470	41.662	39.627	40.377	56.462	19.260	51.260	61.483
RI	DENCLUE	74.988	73.527	71.217	68.384	70.043	69.115	75.024	44.164	57.460	75.477
	OptiGrid	76.909	75.404	75.963	75.448	75.631	76.550	75.359	49.252	59.201	66.201
	FCM	64.876	81.210	75.118	77.645	74.855	66.113	88.302	53.160	62.694	78.981
	EM	87.873	83.664	84.858	65.113	88.302	81.210	84.499	68.081	74.808	84.395
	BIRCH	73.099	65.823	77.521	71.422	73.069	70.589	74.156	33.184	62.357	79.890
MI	DENCLUE	59.853	48.916	39.949	49.533	46.986	37.158	47.439	36.561	49.762	65.762
	OptiGrid	34.966	38.308	36.906	39.429	37.328	34.029	47.197	54.081	33.411	53.411
	FCM	64.256	65.680	76.428	69.129	69.708	72.129	73.242	39.242	50.589	59.257
	EM	74.925	85.077	82.405	86.374	85.550	81.742	85.572	64.029	58.871	67.142
	BIRCH	58.450	58.780	56.230	57.930	57.376	55.750	57.979	25.980	52.764	64.994

TABLE 4. Internal validity results for the candidate clustering algorithms.

Measures	Cls. Algorithms	MHIRD	MHORD	SPFDS	DOSDS	SPDOS	SHIRD	SHORD	ITD	WTP	DARPA
CP	DENCLUE	1.986	1.207	1.886	1.104	1.300	1.391	1.357	1.014	1.485	0.832
	OptiGrid	1.629	1.678	1.643	1.232	1.271	2.505	2.330	2.454	1.189	1.973
	FCM	3.243	1.523	3.014	2.540	2.961	3.504	2.548	3.945	2.555	2.727
	EM	3.849	2.163	4.683	2.405	2.255	4.354	3.198	1.537	2.874	1.367
	BIRCH	3.186	3.466	3.310	5.164	1.692	2.793	5.292	5.529	1.834	4.131
SP	DENCLUE	2.973	1.450	2.776	1.247	1.632	1.810	1.742	1.073	1.993	0.716
	OptiGrid	1.914	1.990	1.936	1.311	1.370	3.247	2.981	3.170	1.245	2.437
	FCM	3.972	1.636	3.660	3.017	3.588	4.326	3.028	4.926	3.038	3.271
	EM	4.535	2.389	5.597	2.696	2.505	5.178	3.706	1.592	3.294	1.375
	BIRCH	3.566	3.907	3.717	5.979	1.742	3.086	6.136	6.425	1.916	4.719
DB	DENCLUE	1.788	2.467	4.273	3.524	4.551	0.821	4.990	6.870	7.702	5.987
	OptiGrid	3.798	5.582	3.085	1.703	2.655	5.573	2.128	4.673	5.078	8.502
	FCM	3.972	2.315	4.036	4.104	3.586	6.964	5.760	10.824	10.239	9.238
	EM	8.164	2.065	4.672	4.989	3.198	2.645	4.776	10.882	10.013	2.320
	BIRCH	4.943	6.471	3.234	2.512	2.600	5.002	5.272	11.882	9.336	6.641
DVI	DENCLUE	0.343	0.508	0.354	0.560	0.472	0.444	0.454	0.620	0.420	0.837
	OptiGrid	0.526	0.518	0.524	0.615	0.603	0.446	0.457	0.449	0.630	0.484
	FCM	0.491	0.606	0.498	0.517	0.500	0.485	0.516	0.476	0.516	0.509
	EM	0.524	0.580	0.512	0.567	0.575	0.516	0.538	0.640	0.548	0.669
	BIRCH	0.568	0.562	0.565	0.539	0.646	0.580	0.537	0.536	0.632	0.550

TABLE 5. Stability of the candidate clustering algorithms.

Data sets	Clustering Algorithms				
	EM	OptiGrid	BIRCH	FCM	DENCLUE
MHIRD	0.495	0.532	0.567	0.596	0.415
MHORD	0.408	0.528	0.537	0.589	0.487
SPFDS	0.478	0.518	0.544	0.599	0.451
DOSDS	0.481	0.593	0.561	0.608	0.467
SPDOS	0.479	0.531	0.556	0.591	0.441
SHIRD	0.476	0.513	0.504	0.559	0.492
SHORD	0.486	0.532	0.562	0.519	0.492
ITD	0.473	0.215	0.372	0.272	0.292
WTP	0.436	0.357	0.307	0.278	0.311
DARPA	0.459	0.481	0.397	0.284	0.359

TABLE 6. Runtime of the candidate clustering algorithms.

Data sets	Clustering Algorithms				
	DENCLUE	OptiGrid	BIRCH	FCM	EM
MHIRD	0.336	0.081	1.103	0.109	3.676
MHORD	0.290	0.290	2.253	7.511	60.689
SPFDS	2.5095	8.365	67.401	139.03	830.55
DOSDS	1.73229	5.7743	86.031	126.471	581.59
SPDOS	6.5178	32.6625	208.875	226.55	1543.4
SHIRD	0.011	0.038	0.811	0.603	3.140
SHORD	0.017	0.058	0.780	0.824	4.929
ITD	7.107	23.689	241.074	262.353	1982.790
WTP	0.230	0.388	1.246	1.768	6.429
DARPA	17.347	56.716	364.592	401.795	20429.281
Average	3.610	12.806	97.416	124.701	2544.647

for evaluation purpose only and are not used in the cluster formation process.

Table 3 shows the results of the candidate clustering with respect to the external validity measures. It can be seen from Table 3 that the EM algorithm provides the best clustering output based on all external measures in comparison to the remaining clustering algorithms. The second best clustering algorithm in terms of external validity is the FCM algorithm. The analysis reveals that BIRCH, OptiGrid and DENCLUE respectively yield the lowest quality of clustering output in comparison to EM and FCM.

Table 4 reports the results of clustering algorithms according to the internal validity measures. This is very important, especially when there is no prior knowledge of the correct

TABLE 7. Compliance summary of the clustering algorithms based on empirical evaluation metrics.

Cls. Algorithms	External Validity	Internal Validity	Stability	Efficiency Problem	Scalability
EM	Yes	Partially	Suffer from	Suffer from	Low
FCM	Yes	Partially	Suffer from	Suffer from	Low
DENCLUE	No	Yes	Suffer from	Yes	High
OptiGrid	No	Yes	Suffer from	Yes	High
BIRCH	No	Suffer from	Suffer from	Yes	High

class labels of the datasets. Each of the validation measures evaluates different aspects of a clustering output separately, and based just on the raw data. None of them uses explicit information from the domain of application to evaluate the obtained cluster. Note that the best value for each measure for each of the clustering algorithms is shown in bold. There are several observations from Table 4. First. it can seen that DENCLUE algorithm often produces compact clusters in comparison to other clustering algorithms. The compactness of the DENCLUE is only 37.26 percent of that of OptiGrid, 47.27 percent of that of EM, 47.48 percent of that FCM and 75.74 percent of that of BIRCH. Second, for the separation measure, we observe that the EM algorithm often yields clusters with higher mean separations among the considered clustering algorithms. The separation results of the EM algorithm are 42.27 percent of that of DENCLUE, 50.52 percent of that of OptiGrid, 52.98 percent of that of FCM and 80.60 percent of that of BIRCH. Third, according to the Davies-Bouldin index (DB), it can be seen that EM, DENCLUE and OptiGrid respectively were often able to produce not only compact clusters, but also well-separated clusters.

F. EVALUATING STABILITY

The main focus of this section is to compare the stability of the candidate clustering algorithms output for 10-fold on all datasets. The stability would measure the variation in the outputs of a particular clustering algorithm rather than a general property of the dataset. Thus higher values indicate lower output changes and are always preferable. For comparison, Table 5 displays the stability results obtained for each clustering algorithm on all datasets. Note that the sequence roughly orders the candidate clustering algorithms according to growing stability values. Let us point out some of the most notable phenomena that can be observed regarding the presented stability results. First, the overall stability level in most cases only rarely approach 0.599, indicating that clustering algorithms often suffer from stability issues and frequently fail to produce stable output. Second, it can be seen that in most cases the EM algorithm achieves the highest stability value on all datasets except for ITD and WTR and DARPA datasets. Third, it can be seen that the OptiGrid and DENCLUE algorithms often yield the highest stability values for ITD and WTR and DARPA datasets among all considered clustering algorithms. This confirms their suitability for dealing with high-dimensional datasets. Finally, Table 5 shows that FCM scores the lowest stability values on datasets with high problem dimensionality. Future work would investigate the stability of clustering algorithms using different parameter settings.

G. EVALUATING RUNTIME AND SCALABILITY

A key motivation for this section is to evaluate the ability of the candidate clustering algorithms to group similar objects efficiently (i.e. with faster runtimes). This is particularly important when of the collected data is very large. In order to compare the effectiveness of the candidate clustering algorithms, we applied each clustering algorithm to the ten datasets. We then measure the execution time required by each algorithm on an Intel core i7 2.80 GHz processor machine with 8 Gbytes of RAM. Table 6 records the runtime of the five candidate clustering algorithms. First, we observe that the DENCLUE is significantly faster than all other clustering algorithms. The runtime of DENCLUE 0.1 percent of that of EM, 2.89 percent of that of FCM, 3.71 percent of that of BIRCH and 28.19 percent of that of OptiGrid. This indicates that DENCLUE is more efficient than others when choosing clustering to deal with big data. Second, the EM algorithm had the slowest runtime of all, and was slower than FCM, BIRCH, OptiGrid and DENCLUE by 20.40, 26.12, 198.70 and 704.94 percent, respectively. This indicates that EM is less efficient, and thus it is not recommended for big data.

VI. CONCLUSION

This survey provided a comprehensive study of the clustering algorithms proposed in the literature. In order to reveal future directions for developing new algorithms and to guide the selection of algorithms for big data, we proposed a categorizing framework to classify a number of clustering algorithms. The categorizing framework is developed from a theoretical viewpoint that would automatically recommend the most suitable algorithm(s) to network experts while hiding all technical details irrelevant to an application. Thus, even future clustering algorithms could be incorporated into the framework according to the proposed criteria and properties. Furthermore, the most representative clustering algorithms of each category have been empirically analyzed over a vast number of evaluation metrics and traffic datasets. In order to support the conclusion drawn, we have added Table 7, which provides a summary of the evaluation. In general, the empirical study allows us to draw the following conclusions for big data:

- No clustering algorithm performs well for all the evaluation criteria, and future work should be dedicated to accordingly address the drawbacks of each clustering algorithm for handling big data.
- EM and FCM clustering algorithms show excellent performance with respect to the quality of the clustering

outputs, except for high-dimensional data. However, these algorithms suffer from high computational time requirements. Hence, a possible solution is to rely on programming language and advances hardware technology which may allow such algorithms to be executed more efficiently.
- All clustering algorithms suffer from stability problem. To mitigate such an issue, ensemble clustering should be considered.
- DENCLUE, OptiGrid and BIRCH are suitable clustering algorithms for dealing with large datasets, especially DENCLUE and OptiGrid, which can also deal with high dimensional data.

As future work, we would investigate the following questions:

- Are ensembles of single clustering algorithm more stable and accurate than individual clustering?
- Are ensembles of multi-clustering algorithms more stable and accurate than ensemble of a single clustering?
- How to incorporate the concept of distributed system to improve the performance and efficiency of existing algorithms on big data?
- How can the most suitable parameter settings be found for each clustering algorithm?

REFERENCES

[1] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Comput. Commun.*, vol. 30, nos. 14–15, pp. 2826–2841, Oct. 2007.

[2] C. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Mining Text Data*. New York, NY, USA: Springer-Verlag, 2012, pp. 77–128.

[3] A. Almalawi, Z. Tari, A. Fahad, and I. Khalil, "A framework for improving the accuracy of unsupervised intrusion detection for SCADA systems," in *Proc. 12th IEEE Int. Conf. Trust, Security Privacy Comput. Commun. (TrustCom)*, Jul. 2013, pp. 292–301.

[4] A. Almalawi, Z. Tari, I. Khalil, and A. Fahad, "SCADAVT-A framework for SCADA security testbed based on virtualization technology," in *Proc. IEEE 38th Conf. Local Comput. Netw. (LCN)*, Oct. 2013, pp. 639–646.

[5] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Rec.*, 1999, vol. 28, no. 2, pp. 49–60.

[6] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, nos. 2–3, pp. 191–203, 1984.

[7] J. Brank, M. Grobelnik, and D. Mladenić, "A survey of ontology evaluation techniques," in *Proc. Conf. Data Mining Data Warehouses (SiKDD)*, 2005.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *J. Roy. Statist. Soc., Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. ACM SIGKDD Conf. Knowl. Discovery Ad Data Mining (KDD)*, 1996, pp. 226–231.

[10] A. Fahad, Z. Tari, A. Almalawi, A. Goscinski, I. Khalil, and A. Mahmood, "PPFSCADA: Privacy preserving framework for SCADA data publishing," *Future Generat. Comput. Syst.*, vol. 37, pp. 496–511, Jul. 2014.

[11] A. Fahad, Z. Tari, I. Khalil, I. Habib, and H. Alnuweiri, "Toward an efficient and scalable feature selection approach for internet traffic classification," *Comput. Netw.*, vol. 57, no. 9, pp. 2040–2057, Jun. 2013.

[12] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Mach. Learn.*, vol. 2, no. 2, pp. 139–172, Sep. 1987.

[13] J. H. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," *Artif. Intell.*, vol. 40, nos. 1–3, pp. 11–61, Sep. 1989.

[14] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," in *Proc. ACM SIGMOD Rec.*, Jun. 1998, vol. 27, no. 2, pp. 73–84.

[15] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," *Inform. Syst.*, vol. 25, no. 5, pp. 345–366, 2000.

[16] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. San Mateo, CA, USA: Morgan Kaufmann, 2006.

[17] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. ACM SIGKDD Conf. Knowl. Discovery Ad Data Mining (KDD)*, 1998, pp. 58–65.

[18] A. Hinneburg and D. A. Keim, "Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering," in *Proc. 25th Int. Conf. Very Large Data Bases (VLDB)*, 1999, pp. 506–517.

[19] Z. Huang, "A fast clustering algorithm to cluster very large categorical data sets in data mining," in *Proc. SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, 1997, pp. 1–8.

[20] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[21] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.

[22] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modelling," *IEEE Comput.*, vol. 32, no. 8, pp. 68–75, Aug. 1999.

[23] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. New York, NY, USA: Wiley, 2009.

[24] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1, pp. 1–6, 1998.

[25] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, Berkeley, CA, USA, 1967, pp. 281–297.

[26] A. Mahmood, C. Leckie, and P. Udaya, "An efficient clustering scheme to exploit hierarchical data in network traffic analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 752–767, Jun. 2007.

[27] A. N. Mahmood, C. Leckie, and P. Udaya, "ECHIDNA: Efficient clustering of hierarchical data for network traffic analysis," in *Proc. 5th Int. IFIP-TC6 Conf. Netw. Technol., Services, Protocols Perform. Comput. Commun. Netw. Mobile Wireless Commun. Syst. (NETWORKING)*, 2006, pp. 1092–1098.

[28] M. Meilă and D. Heckerman, "An experimental comparison of several clustering and initialization methods," in *Proc. 14th Conf. Uncertainty Artif. Intell. (UAI)*, 1998, pp. 386–395.

[29] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt, "Architecture of a network monitor," in *Proc. Passive Active Meas. Workshop (PAM)*, LaJolla, CA, USA, Apr. 2003.

[30] A. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proc. ACM Int. Conf. Meas. Model. Comput. Syst. (SIGMETRICS)*, 2005, pp. 50–60.

[31] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 1994, pp. 144–155.

[32] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng. (TKDE)*, vol. 14, no. 5, pp. 1003–1016, Sep./Oct. 2002.

[33] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, Mar. 2009.

[34] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 1998, pp. 428–439.

[35] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," in *Proc. DARPA Inform. Survivability Conf. Exposit. (DISCEX)*, 2000, pp. 130–144.

[36] S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie, and M. Palaniswami, "Labelled data collection for anomaly detection in wireless sensor networks," in *Proc. 6th Int. Conf. Intell. Sensors, Sensor Netw. Inform. Process. (ISSNIP)*, Dec. 2010, pp. 269–274.

[37] W. Wang, J. Yang, and R. R. Muntz, "Sting: A statistical information grid approach to spatial data mining," in *Proc. 23rd Int. Conf. Very Large Data Bases (VLDB)*, 1997, pp. 186–195.

[38] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[39] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proc. 14th IEEE Int. Conf. Data Eng. (ICDE)*, Feb. 1998, pp. 324–331.

[40] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Rec.*, Jun. 1996, vol. 25, no. 2, pp. 103–114.

ADIL FAHAD received the B.S. degree in computer science from King Abdul Aziz University, Jeddah, Saudi Arabia, in 2003, and the M.S. (Hons.) degree from the Royal Melbourne Institute of Technology, Melbourne, VIC, Australia, in 2008, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Technology.

His research interests are in the areas of wireless sensor networks, mobile networks, and ad hoc networks with emphasis on data mining, statistical analysis/modeling, and machine learning.

NAJLAA ALSHATRI received the B.S. degree in computer science from King Abdul Aziz University, Jeddah, Saudi Arabia, in 2008. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Technology, Royal Melbourne Institute of Technology, Melbourne, VIC, Australia.

Her research interests are in the areas of machine learning, big data, and data stream.

ZAHIR TARI is a Full Professor of Distributed Systems with the Royal Melbourne Institute of Technology (RMIT), Melbourne, VIC, Australia. He received the bachelor's degree in mathematics from the University of Algiers, Algiers, Algeria, in 1984, and the M.Sc. degree in operational research and the Ph.D. degree in computer science from the University of Grenoble, Grenoble, France, in 1985 and 1989, respectively. From 1990 to 1992, he joined the Database Laboratory at the Swiss Federal Institute of Technology, Zurich, Switzerland, as a Researcher, where he was involved in various aspects of distributed database systems. In 1996, he joined RMIT as a Senior Lecturer and is currently a Professor, where he led the Distributed Systems and Networking Discipline at the School of Computer Science and IT. He is an expert in the areas of system performance (e.g., Web servers, P2P, an cloud) and system security (e.g., SCADA systems and cloud). He has co-authored six books (John Wiley and Springer) and has edited over 25 conference proceedings. He was a recipient of over 5M$ in funding from the Australian Research Council and lately part of a successful 7th Framework Australia to European bid onĆ Authorization and Authentication for Entrusted Unions. He is an Associate Editor of the *IEEE Transactions on Computers*, the *IEEE Transactions on Parallel and Distributed Systems*, and the *IEEE Magazine on Cloud Computing*.

ABDULLAH ALAMRI received the bachelor's degree in computer science from King Khalid University, Abha, Saudi Arabia, in 2007, and the master's degree in information technology from the School of Engineering and Mathematical Sciences, La Trobe University, Melbourne, VIC, Australia, in 2009. He is currently pursuing the Ph.D. degree with the School of Computer Science and Information Technology, Royal Melbourne Institute of Technology, Melbourne . His research interests include semantic web, linked data and relational databases, and data integration.

IBRAHIM KHALIL is a Senior Lecturer with the School of Computer Science and IT, Royal Melbourne Institute of Technology (RMIT), Melbourne, VIC, Australia. He has several years of experience in silicon valley-based companies, where he is involved in large network provisioning. He was also an Academic in several research universities. Before joining RMIT, he was with the Swiss Federal Institute of Technology, Lausanne, Switzerland, the University of Berne, Bern, Switzerland, and Osaka University, Suita, Japan. His research interests are quality of service, wireless sensor networks, and remote healthcare.

ALBERT Y. ZOMAYA is currently the Chair Professor of High Performance Computing Networking and an Australian Research Council Professorial Fellow with the School of Information Technologies, University of Sydney, Sydney, NSW, Australia. He is also the Director of the Centre for Distributed and High Performance Computing at the University of Sydney, which was established in 2009. He has authored and co-authored seven books, more than 400 papers, and the Editor of nine books and 11 conference proceedings. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTERS, and serves as an Associate Editor for 19 leading journals, such as the *IEEE Transactions on Parallel and Distributed Systems* and *Journal of Parallel and Distributed Computing*. He was a recipient of the Meritorious Service Award (2000) and the Golden Core Recognition (2006) both from the IEEE Computer Society. He is also a recipient of the IEEE Technical Committee on Parallel Processing Outstanding Service Award and the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing, both in 2011. He is a Chartered Engineer, a fellow of the American Association for the Advancement of Science and the Institution of Engineering and Technology, U.K., and a Distinguished Engineer of the Association for Computing Machinery.

SEBTI FOUFOU received the Ph.D. degree in computer science from the University of Claude Bernard Lyon I, Villeurbanne, France, in 1997, for a dissertation on parametric surfaces intersections. He was with the Department of Computer Science, University of Burgundy, Dijon, France, from 1998 to 2009, as an Associate Professor and then as a Full Professor. His research interests concern geometric modeling and CAD-CAM topics and include surfaces blending using Dupin cyclides, subdivision surfaces, and geometric constraints solving. He was with the National Institute of Standards and Technology, Gaithersburg, MD, USA, from 2005 to 2006 as a temporary Guest Researcher, where he was involved in product engineering-related researches such as smart machining systems, tolerances, assembly modeling, and PLM. He joined the Department of Computer Science and Engineering at Qatar University, Doha, Qatar, in 2009.

ABDELAZIZ BOURAS is the ictQATAR Supreme Council of ICT Scientific Chair with the Department of Computer Science and Engineering, Qatar University, Doha, Qatar. His research interests focus on distributed information modeling and ontology construction. He is currently involved in several international projects, such as the EU FP7 EASY-IMP project on Intelligent Products in the Cloud and the FP7 FITMAN project on Future Internet Technologies, and coordinates a large ICT Erasmus-Mundus Consortia on ICT for Sustainable eTourism. He is currently the Chair of the IFIP WG5.1 Group on ICT for product Life cycle.