Expert Systems with Applications 42 (2015) 2785–2797

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Efficient agglomerative hierarchical clustering

Athman Bouguettaya [a,*], Qi Yu [b], Xumin Liu [b], Xiangmin Zhou [c], Andy Song [a]

[a] RMIT University, Australia
[b] Rochester Institute of Technology, USA
[c] Victoria University, Australia

ARTICLE INFO

Article history:
Available online 22 October 2014

Keywords:
Clustering analysis
Hybrid clustering
Data mining
Data distribution
Coefficient of correlation

ABSTRACT

Hierarchical clustering is of great importance in data analytics especially because of the exponential growth of real-world data. Often these data are unlabelled and there is little prior domain knowledge available. One challenge in handling these huge data collections is the computational cost. In this paper, we aim to improve the efficiency by introducing a set of methods of agglomerative hierarchical clustering. Instead of building cluster hierarchies based on raw data points, our approach builds a hierarchy based on a group of centroids. These centroids represent a group of adjacent points in the data space. By this approach, feature extraction or dimensionality reduction is not required. To evaluate our approach, we have conducted a comprehensive experimental study. We tested the approach with different clustering methods (i.e., UPGMA and SLINK), data distributions, (i.e., normal and uniform), and distance measures (i.e., Euclidean and Canberra). The experimental results indicate that, using the centroid based approach, computational cost can be significantly reduced without compromising the clustering performance. The performance of this approach is relatively consistent regardless the variation of the settings, i.e., clustering methods, data distributions, and distance measures.

1. Introduction

Clustering is an important means of data analytics in real-world scenarios because manual tagging of the data is usually expensive. Furthermore prior knowledge required to facilitate manual tagging is often unavailable or insufficient. Under such circumstances clustering is a more suitable option over supervised learning approaches, such as classification and regression.

Efficient techniques for data clustering has been studied for decades due to its significant implication in real-world applications where the amount of data are often very large and the accumulation of data is often accelerating (Jain, Murty, & Flynn, 1999; Romesburg, 1990). A clustering method which requires less computational cost can be beneficial in general data mining and knowledge discovery, as well as in specific domains e.g. bio-informatics, web usage monitoring and social network analysis. Due to the widespread of web applications, mobile devices and network of sensors, the volume of data to be analyzed grows much faster than computational power, especially in recent years. This flood of data makes efficiency a high priority in developing clustering methods.

In this study we address the efficiency issue of hierarchical clustering which is one of the main stream clustering methods as it is generally applicable to most types of data. In comparison with partitional clustering algorithms such as K-means, hierarchical approaches have higher cost, with a complexity of $O(N^2 logN)$, but they do not require any predefined parameter hence are more suitable for handling real-world data where finding a suitable set of parameters can be tricky.

Hierarchical clustering can go both ways, aggregating from individual points to the most high-level cluster or dividing from a top cluster to atomic data objects. Our focus is the bottom-up approach which is known as the agglomerative approach, because computational cost can be reduced if the bottom-up process starts from somewhere in the middle of the hierarchy and the lower part of the hierarchy is built by a less expensive method such as partitional clustering. This idea would not work well on the top-down approach which is known as divisive hierarchical clustering because it is notorious for its high cost, $O(2^N)$, and verifying middle level sub-clusters by individual data points would still be expensive.

It is possible to use a hierarchical approach to generate middle-level sub-clusters then apply partitional algorithms on these sub-clusters. However predefined parameters like K still need to be determined. Another possible way to improve efficiency in hierarchical clustering is to perform feature extraction or selection,

* Corresponding author. Tel.: +61 399252169; fax: +61 396621617.
E-mail addresses: Athman.Bouguettaya@rmit.edu.au (A. Bouguettaya), qi.yu@rit.edu (Q. Yu), xumin.liu@rit.edu (X. Liu), Xiangmin.zhou@vu.edu.au (X. Zhou), andy.song@rmit.edu.au (A. Song).

http://dx.doi.org/10.1016/j.eswa.2014.09.054
0957-4174/

which may reduce data dimensionality. However that process often requires domain knowledge of the data. It also makes the clustering outcomes dependent on the performance of the feature extraction or selection algorithms.

In this paper we present an efficient agglomerative hierarchical method which does not require feature extraction or selection. The main goals of this study are:

1. Presenting a methodology of combining agglomerative hierarchical clustering and partitional clustering to reduce the overall computational cost. By this method the number of output clusters needs not to be determined beforehand.
2. Studying the behaviors of our methods with different distributions.
3. Evaluating the performance of our methods based on the coefficients of correlation.

The paper is organized as follows. In Section 2, we briefly discuss the related works. In Section 3, we describe the proposed methodology with associated approaches. In Section 4 the datasets used in this study are described. Section 5 shows the experimental settings and the results. The further discussion on the results is presented in Section 6. Section 7 concludes this study with a brief outlook for further studies.

2. Related work

We review some representative clustering analysis techniques in this section and highlight their difference with the proposed approach. We categorize existing approaches into three major categories: partitional, hierarchical, and hybrid clustering, to achieve a more focused discussion and comparison. We also review some important applications of clustering to demonstrate its importance in data-intensive processing environments (Altingövde, Demir, Can, & Ulusoy, 2008; Hruschka, Campello, Freitas, & de Leon F. de Carvalho, 2009; Jacinth Salome & Suresh, 2012; Jain et al., 1999; Lee, Han, & Whang, 2007; Lin, Liu, & Chen, 2005; Liu & Yu, 2005; Liu, Li, Sim, & Wong, 2007; Lee, Han, Li, & Gonzalez, 2008; Ordonez & Omiecinski, 2004; Pan, Zhang, & Wang, 2008; Rokach, 2010; Xu & Wunsch, 2010; Zhou et al., 2009).

2.1. Hierarchical clustering

A hierarchical algorithm yields a dendrogram representing the nested grouping of patterns and similarity levels at which groupings change (Jain et al., 1999). The clustering process is performed by merging the most similar patterns in the cluster set to form a bigger one. In Bouguettaya (1996) and Bouguettaya, Qi, Park, and Delis (2002), Bouguettaya et al. investigated the different hierarchical clustering algorithms, including UPGMA, WARDS, SLINK, CLINK, etc, and studied the behavior and stability of these algorithms on low-dimensional and high-dimensional data respectively. Hierarchical clustering approaches produce clusters of higher quality. However, these approaches suffer from high time cost. The efficiency of hierarchical algorithms can be improved with the support of index structures (Zhang, Ramakrishnan, & Livny, 1996). In Zhang et al. (1996), "Balancing Iterative Reducing and Clustering using Hierarchies" (BIRCH) has been proposed for minimizing the running time of clustering. BIRCH incrementally clusters very large datasets, whose sizes are much greater than the amount of available memory. Given a large dataset, the clustering process is performed by constructing a height-balanced tree, called CF tree. The algorithm continuously parses the dataset and updates the CF tree until the final result is achieved. BIRCH algorithm adopts the notion of clustering features to capture the information of a cluster. The clusters that are built so far by the algorithm are organized into the CF tree. The leaf node of the CF tree is a sub-cluster instead of a single data point. Therefore, CF tree is a concise representation of the original dataset and can fit into the memory. However, different from the proposed approach, BIRCH adopts the centroid method with fixed order of the points, which may affect the behavior of clustering results.

In recent years, evolutionary computation has been introduced into clustering. As a kind of stochastic search methods, it can often be quite effective in finding optimal solutions. However the efficient aspect is rather an issue as an evolutionary process is time-consuming (Wu, Hu, Maybank, Zhu, & Li, 2012). To speed up a hierarchical agglomerative clustering process, GPU can certainly be utilized (Shalom & Dash, 2013). This study does not involve GPU although the proposed method can include GPU to further enhance the execution time.

2.2. Partitional clustering

In contrast to the hierarchical clustering, a partitional clustering algorithm obtains a flat partition of the dataset which optimizes a predefined criterion function. The most widely used partitional clustering algorithm is K-means clustering, which repeatedly assigns each object to its closest cluster center and computes the new cluster centers accordingly until the predefined criterion is met. Based on how the distance between data points is computed, various partitional clustering algorithms have been developed and representative ones include spectral clustering (Luxburg, 2007; Ng, Jordan, & Weiss, 2001), graph-partitioning based (Dhillon, Guan, & Kulis, 2004), and non-negative matrix factorization based approaches (Li & Ding, 2006). Comparing with K-means clustering, these algorithms usually generate clusters of better quality. However, these algorithms are more computationally involved, requiring performing eigendecomposition or repetitive matrix multiplication, making them not scalable to very large datasets. Mixture model or other density based clustering algorithms output soft cluster memberships, allowing each data point to be associated with multiple clusters with different probabilities. Compared with the proposed approach and hierarchical clustering in general, partitional clustering algorithms suffer two major limitations. First, their performance heavily relies on pre-defined parameters, especially the number of clusters, so the quality of data clusters can not be guaranteed. Second, the resultant clusters have a flat structure instead of hierarchical structure that captured much richer relationship among data points. A hierarchical structure offer a more natural way to organize many real-world objects (e.g., documents and webpages) and facilitate human users to browse the data.

2.3. Hybrid clustering

Hybrid data clustering combines the hierarchical and partitional methods to obtain the good quality of the former and the efficiency of the latter. Different hybrid data clustering algorithms have been proposed (Guha, Rastogi, & Shim, 1998; Lin & Chen, 2005; Wattanachon, Suksawatchon, & Lursinsap, 2009). In Guha et al. (1998), a hybrid clustering algorithm called CURE was proposed to effectively identify the arbitrarily shaped clusters. Given a large dataset, CURE draws a set of data samples from the whole dataset by random sampling. The data samples are grouped as several partitions and those in each partition are partially clustered. The outliers are then removed from the dataset. The final clusters are obtained by further clustering over the partial clusters produced in the previous step. CURE is scalable to large datasets with a linear time complexity. However, different from the proposed approach, it

still requires the user-specified parameter values including the number of clusters and the shrinking factor, which may affect the quality of clusters. In Lin and Chen (2005) a Cohesion-based Self-Merging (CSM) clustering algorithm is proposed. CSM adopts a new similarity measure, referred to as cohesion, to calculate the distance between clusters. Cohesion is defined based on the merging inclination of two clusters according to the existence of a shared data point. Since the merging inclination should not be determined by only a few points, cohesion collectively considers all the data points in the two clusters to be merged. This makes the cohesion measure robust to the existence of outliers. By using cohesion, CSM effectively combines the features of partitional and hierarchical clustering methods. In the first phase, it partitions the original data space into small clusters using k-means. Then the obtained small clusters are merged together using the cohesion similarity measure in a hierarchical manner. Experimental studies demonstrate that CSM excels at both clustering accuracy and execution time. However, since CSM requires users to specify the number of sub-clusters expected in the data partition stage and the number of final clusters, it implies that suitable parameter values need to be supplied according to the domain knowledge of a particular database. In contrast, the proposed approach is domain independent. The Self-Partition and Self-Merging (SPSM) algorithm also tried to reduce the effect of user-specified parameters by employing a recursive data partition processing (Wattanachon et al., 2009). The sub-clusters are produced by recursively dividing the dataset into four partitions. However, it still suffers from the negative effect of dividing a single cluster into different parts or grouping data pointers of different categories into one cluster. This is caused by the pre-defined number (i.e., four) of sub-clusters for each partitioning step. In Cheng, Kannan, Vempala, and Wang (2005), authors proposed to combine a divisive strategy and an agglomerative strategy. Both the divisive and agglomerative components use a kind of hierarchical algorithms, thus this method does not take advantage of the benefits from partitional clustering as in the proposed approach.

2.4. Important applications

The main purpose of using data clustering techniques is to improve the performance of data access by summarizing the data objects into groups. Often a group of clustering methods or a combination of clustering with other methods works well. In Ordonez and Omiecinski (2004), Ordonez et al. proposed to integrate clustering with a relational DBMS for allowing K-means algorithm to cluster large data sets inside a relational database management system. Unlike the standard K-means approach that manages the input data and clustering results in memory, all the data are stored in disk. The performance of clustering is improved by statistics based initialization of centroids and achieved fast convergence. In Lee et al. (2007), a *partition-and-group* framework was proposed to discover common sub-trajectories from a trajectory database using trajectory-based clustering which is combined with a proposed region-based clustering in Lee et al. (2008). This region-based clustering discovers the regions having trajectories of one major type. The trajectory-based clustering exploits the move patterns of trajectories based on their low-level features, while the region-based clustering utilizes more general features without considering particular move patterns. Both the efficiency and accuracy could be improved due to the collaboration between these two different clustering methods. In Altingövde et al. (2008), a cluster-skipping inverted file is proposed based on the partitional clustering for efficient retrieval of documents. Besides the general document information, the cluster membership and centroid information are stored in the inverted file as well. Clustering techniques have been applied to analysis of microarray datasets (Pan et al., 2008; Zhao et al., 2008). For example, Pan et al. proposed to use sampling-based matrix decomposition for fast co-clustering of microarray data (Pan et al., 2008). Zhao et al. proposed to identify co-regulated gene clusters by a new tree-based clustering algorithm (Zhao et al., 2008). Subspace based clustering has been proposed to overcome the dimensionality curse of high dimensional data (Parsons, Haque, & Liu, 2004). In Liu et al. (2007), a distance-based clustering model called nCluster was proposed to identify the significant clusters by a flexible dimension partition approach which allows the overlapping between different bins of an attribute. In Zhou et al. (2009), an optimized visual dictionary is proposed which is established on the subspace-based clustering for effective and efficient video retrieval. The high-quality clusters are obtained by first finding an optimal subspace combination which produces the maximal discrimination power, and then performing the recursive K-means algorithm over each dominant subspace. Consequently the high accuracy of video retrieval is preserved. This work is based on the above studies as we investigate the behavior and quality of hybrid clustering technique for large databases based on hierarchical clustering and partitional clustering from a broad view of point. We propose a set of hybrid data clustering solutions which can be a basis of clustering based applications.

3. Methodology

In this section, we will describe the key elements for designing the clustering techniques and conducting the experiments. The parameters that will be used in this analysis include the *measure of resemblance*, *clustering methods*, *statistical distribution*, and *coefficients of correlation*. Other related concepts are types of data (*quantitative* vs. *qualitative*) and *normalization*. In what follows, we overview these criteria as related to the proposed study.

3.1. Overview

Our approach, referred to as *KnA*, integrates *K-means* and Agglomerative approaches, when generating the clustering hierarchy. The process first applies K-means to the individual data objects and generates k clusters, called as *middle-level* clusters. Each cluster is then represented by its centroid. The clustering process then applies agglomerative clustering approaches, such as Single Linkage method (SLINK) and Group Average Linkage (UPGMA), on those centroids, to build the final clustering hierarchy. The distance between two middle-level clusters are measured as the distances between their centroids.

We will evaluate the performance of our approach and compare it to the traditional agglomerative clustering approaches where the step of applying K-means clustering is skipped, i.e., the hierarchy is directly build on the original data objects. We will compare these two methods in terms of effectiveness, i.e., the accuracy of the clustering result, and efficiency, i.e., the CPU time of the entire clustering process. We believe that, if the centroids-based method produces a hierarchy highly correlated to the hierarchy produced using individual points, then it is more preferred as it always has the better efficiency than the traditional ones.

3.2. Distance measure

Measuring distance between data objects is the foundation of clustering algorithms since all of these algorithms are built on proximity or similarity between data objects. Two common distance measures are used here, Euclidean *Distance* and Canberra *Distance*.

- Euclidean *Distance*, e_{jk},

$$e_{jk} = \sqrt{\sum_{i=1}^{n}(X_{ij} - X_{ik})^2}, \quad 0 < e_{jk} < \infty$$

- Canberra *Distance*, c_{jk},

$$c_{jk} = (1/n)\sum_{i=1}^{n}\left(\frac{|X_{ij} - X_{ik}|}{(X_{ij} + X_{ik})}\right), \quad 0 \leqslant c_{jk} \leqslant 1$$

where j and k are two objects, n is the number of attributes available in the data objects, X_{ij} and X_{ik} are the values of the i*th* attribute of objects j and k, respectively.

Both distance measures are sensitive to *additive*, *mirror image*, and *proportional* translations. We should note that for *Canberra* coefficient to be defined, all denominators should be non-zero. This means that the coefficient is not defined where two objects both have zero values for the same attribute. Similar to measures like *Bray-Curtis*, *Average Euclidean*, and *Cosine Similarity*, these two types of distance are mainly used for quantitative attributes. For qualitative attributes, there are more suitable methods defined, among which are the *Jaccard*, *Sorenson*, and *Hamann* coefficients (Romesburg, 1990).

3.3. Partitional clustering: K-means

K-means is the most well-known partitional clustering algorithm. The output of this algorithm is a *flat structure* of clusters. The main steps are:

1. Selecting a few centroid points randomly.
2. Assigning each data point to the closest centroid.
3. Updating centroids by calculating central points of these newly formed clusters.
4. Repeating the previous two steps until no object is resigned to another cluster.

The *K-means* clustering is advantageous in efficiency as its computational complexity is only linear to sample size N and number of clusters $K, O(NK)$. How does the performance of K-means depend on the initial choice of K? For many data it is difficult to determine the optimal value especially when domain knowledge is absent or inadequate.

3.4. Agglomerative hierarchical clustering

There are a few possible ways to perform agglomerative hierarchical clustering here. However they generally follow the following major steps:

1. Calculating the proximity matrix for the initial clusters which are the output from the above K-means process.
2. Searching for the minimal distance in the matrix.
3. Combining the two clusters with the minimal distance.
4. Updating the proximity matrix by calculating the distances between the new cluster with the other clusters
5. Repeating the previous three steps if more than one cluster remains.

The complexity of such a process is at least $O(N^2)$, and may go up to $O(N^2 logN)$. There are mainly two approaches of agglomerative clustering used in our methodology, Unweighted Pair-Group Method of Average (UPGMA) and Single Linkage (SLINK). In additional Unweighted Pair-Group Method of Centroids (UPGMC) is also used in our extended experiments.

UPGMA: The average distances between *all* pairs of objects in two clusters are calculated. If cluster X has n_X objects, and cluster Y has n_Y objects, the distance between clusters X and Y will be calculated as the average of all distances (i.e., $n_X \cdot n_Y$ distances) from x_i to y_j, where x_i is an object in X and y_j is an object in Y. The distances between all pairs of clusters are calculated. The two clusters with minimum distance are then merged. This can be repeated until all clusters merge into one. The distance between two clusters is calculated as:

$$\mathcal{D}_{X,Y} = \frac{\sum \mathcal{D}_{x,y}}{n_X \cdot n_Y}$$

where X and Y are the two clusters, n_X and n_Y are the number of objects in clusters X and Y, respectively, x and y are objects in clusters X and Y, and $\mathcal{D}_{x,y}$ is the distance between objects x and y, and $\mathcal{D}_{X,Y}$ is the distance between clusters X and Y.

SLINK: The distance between clusters is calculated as the minimum distance among each pair of objects in two clusters X and Y. If cluster X has n_X objects, and cluster Y has n_Y objects, the distance between clusters X and Y is calculated as the minimum of all distances from x to y, where x is an object in X and y is an object in Y.

$$\mathcal{D}_{X,Y} = min(D_{x,y})$$

where $\mathcal{D}_{x,y}$ is the distance between objects x and y, and $\mathcal{D}_{X,Y}$ is the distance between clusters X and Y. When the distances between all pairs of clusters have been calculated, then the two clusters with *minimum* distance will be merged. This can be repeated until all clusters merge into one. This algorithm has a tendency to *chaining* which makes it suitable for with chain-like and concentric clusters (Romesburg, 1990).

UPGMC: The distance between two clusters X and Y, is calculated as the distance between their centroids instead of their averages.

$$\mathcal{D}_{X,Y} = \mathcal{D}_{\bar{x},\bar{y}}$$

where $\bar{x}$ and $\bar{y}$ are the centroids of cluster X and Y respectively, which are calculated by arithmetic mean of the corresponding objects in the cluster.

3.5. Coefficients of correlation

The coefficient of correlation calculates the relationship between two variables X and Y. In this study we use the Pearson coefficient which can be calculated as:

$$r_{X,Y} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\left(\sum_{i=1}^{n} x_i\right) \times \left(\sum_{i=1}^{n} y_i\right)}{n}}{\sqrt{\left(\left(\sum_{i=1}^{n} x_i^2\right) - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}\right) \times \left(\left(\sum_{i=1}^{n} y_i^2\right) - \frac{\left(\sum_{i=1}^{n} y_i\right)^2}{n}\right)}} \quad (1)$$

where $X = (x_1, x_2, x_3, \ldots, x_n)$ and $Y = (y_1, y_2, y_3, \ldots, y_n)$. The coefficients of correlation draw their values from the range $[-1, 1]$ with the value 1 representing a *perfect* correlation, 0 representing *no* correlation, -1 representing *negative* correlation.

3.6. Normalization

As the nominal value does not usually correspond to the importance of the attribute, there is a need to give all attributes appropriate and comparable weights. This is achieved by normalizing the attributes' values There are different functions that can be used to normalize the attributes. One of the most widely used is

$$Z_{ij} = (X_{ij} - Avg_i)/\sigma_i,$$

where X_{ij} is the ith attribute of jth object, Avg_i is the average for attribute i across the objects, and σ_i is the standard deviation for the ith attribute.

Another function, falling in the group which uses proportions, is:

$$Z_{ij} = (X_{ij} - X_{i_{min}})/(X_{i_{max}} - X_{i_{min}}),$$

where $X_{i_{min}}$ and $X_{i_{max}}$ are the minimum and the maximum value, respectively, of the ith attribute over all objects. X_{ij} is the value of attribute i of object j, and Z_{ij} is the normalized value of X_{ij}. When considering a selection of a normalizing function, one could take into account the fact that the second one, produces positive normalized values within the range [0,1], while the first one produces positive, as well as negative normalized values. In this study the second function is chosen as the normalization method.

3.7. Handling qualitative and quantitative data

In general data can be classified as either *qualitative* or *quantitative*. Quantitative data are continuous or discrete values, e.g. *height*, *weight*, and *speed*. In contrast qualitative data contain nominal states, e.g. *color*, *education*, *gender*, and *disease symptoms*. For distance measure numerical encoding is needed so data can be stored numerically. Qualitative data are encoded as multi-state or binary. Multi-state can be represented by ordinal encoding. For example data objects can be *teacher* = 1, *doctor* = 2, *student* = 3) or *teacher* = [1,0,0], *doctor* = [0,1,0], *student*=[0, 0, 1]. The difference between two qualitative attributes x_i and y_j (which are both represented as multi-state vectors) can be calculated as: $1 - x_i \cdot y_j$, where $x_i \cdot y_j$ denotes the dot product of x_i and y_j.

4. Experimental data sets

In this study we used two types of data: *synthetic* data and *real* data for evaluation. They are described below.

4.1. Synthetic data

To evaluate whether the KnA clustering method presented here works well on different types of data, we introduced a set of synthetic data in which the distribution can be easily controlled. Data sets with *Uniform distribution* and *Normal distribution* were used in the main experiments and data sets with *Exponential distribution* and *Zipf distribution* are used in the extended experiments. These four distributions are described below:

Uniform distribution: This distribution is described by $\mathcal{F}(x) = x$, where the density function is $f(x) = \mathcal{F}'(x) = 1$, for $0 \leqslant x \leqslant 1$, or in more general case, for $a \leqslant x \leqslant b$, $f(x) = 1/(b-a)$. $f(x) = 0$ for x outside this interval. This distribution means that the probability of x falling within an interval is proportional to the size of the interval and is the same for all values of x.

Normal distribution: Normal distribution (also called *Gaussian* distribution) is described by its density function $f_X(x_1,\ldots,x_N) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$, where $\mu = [\mu_1,\ldots,\mu_N]^T$ and Σ is a positive-definite covariance matrix. This distribution is the one most often found in nature. Its shape is like a bell, hence its known as bell distribution.

Exponential distribution: The exponential distribution is frequently used to model the time interval between successive random events. Examples of variables distributed in this manner would be the gap length between cars crossing an interaction or arrivals of customers at the checkout counter in a store. The exponential distribution function is defined as: $f(x) = \lambda \times e^{-\lambda x} (0 \leqslant x < \infty)$, where λ is an exponential function parameter.

Zipf distribution: A Zipf distribution is a set of values (samples) that follow the Zipf's law. The density function of Zipf distribution is defined as: $f(x) = C/x^{\alpha}$ for x in $1 \leqslant x \leqslant N$ where C is the normalization constant (i.e., $\sum f(x) = 1$).

We generated multidimensional sets of data with 50 attributes, using *uniform* and *normal* distributions. The sizes of the datasets start from 100 objects with increments of 100 up to a size of 600. We first generate 50 seed values (as many as the number of attributes) and use each one of them as initial input for a generator of random numbers. We successively use the uniform distribution and normal distribution. For example, for the 100 objects set, we generate 100 values for each attribute. Each step is repeated for both distributions. Data is normalized using ratio standardizing method.

4.2. Real-world movie data

In this set of experiments, we use real-world data: a large dataset on *movie ratings* (Project, n.d.). We used the publicly available dataset that consists of approximately one million ratings for 3900 movies by 6040 users.

In what follows, we describe this movie ratings data. The master file contains *user id*, *movie id*, *ranking*, and *timestamp*. Another file contains the movie genre classification and user information. For faster processing, we replaced the *user id* with *gender*, *occupation* and *age* data from the user file. The gender takes *F* (Female) or *M* (Male) values. We replaced these values with 0 or 1, respectively. The *occupation* is encoded as ordinal values from 1 to 20. The *age* is in groups indicating upper age limit. For example, the value 1 is for users up to 17 years old, 18 is for users from 18 to 24 years old, 25 is for users 25 to 35, etc. We replaced the *movie id* with 18 fields of genre information from the movie file. Each field represents a particular genre. A movie has a value of 1 if it falls in this genre or 0 otherwise. Understandably, most of the fields values have zeros for each of the records.

It is important to note that we use sets of data with mixed attribute types. For example, the first attribute, *gender*, takes a binary value. The second and third attributes, *occupation* and *age*, are qualitative, and are encoded using ordinal numbers. The fourth to the twentieth attributes are multi-state binary, where 1 means presence of the attribute value and 0 means absence of the attribute value. The last one, *ranks*, belongs to the qualitative group. As previously mentioned, there are different coefficients of resemblance that are suitable for this type of data. We use a popular approach to selecting the resemblance coefficient in case of mixed types of attributes: they are *ignored*. We then use *Euclidean* coefficient as a resemblance measure. The reason for not using the *Canberra* coefficient is the nature of the set of data: 19 of the total of 22 attributes can only have 0 or 1 values. Furthermore, because 18 of these 19 describe genre, i.e., a movie falls into one, two, three categories but rarely in more than three categories, about 15 attributes with zero values can be present in a single data object. The probability that an attribute value is zero simultaneously for two objects is very high. This makes the *Canberra* distance unfit for this type of data because it is undefined when two objects have a zero for the same attribute value.

There are essentially three main parts in the experiments using real data. Once we have the file in this form, it is sorted by timestamp. A generator of random numbers with uniform distribution is used to draw random sets of data from the 1,000,000-records sorted master dataset. As a result, six datasets with sizes 100 to 600 at a step of 100 are obtained. As was the case with synthetic data, we proceed in a similar manner except that the *Canberra* distance is not used. In the second part of the experiment, the same number of records is drawn from the master file for each group

of users, where a group of users is formed by age and occupation. Gender is left out as we want to be sex-blind for the purpose of the study. We use 120 age-occupation groups. Sets with sizes 120, 240, 360, 480, 600, 720 are drawn. The third part of the real data experiments is based on a third type of samples of the 1,000,000 record dataset. From the 20 occupations, an equal number of records are randomly selected and drawn from the original dataset. The resulting sets are grouped by occupation. For example, for the smallest dataset of size 100 we have 5 objects with attribute occupation 1, followed by 5 with occupation 2, etc.

5. Experiments and results

5.1. Experimental settings

As mentioned previously, our hierarchical method is based on *K-means* where the *K* value is to be defined. In our method, this parameter is determined as the ratios of the number of clusters to total number of original data-objects, ranged from 0.1 to 1.0 at a step of 0.1. For example, each experiment will carry out 10 times for a dataset of 600 objects with the number of clusters varying from 60 to 600 at a step of 60. Distances between data objects are *Euclidean* distance and *Canberra* distance.

Based on the output from the above *K-means*, hierarchies were generated by *UPGMA* and *SLINK* in the main experiments and by *UPGMC* in the extended experiments. Hierarchies were also generated directly on those original data objects for comparison purpose. *Pearson* correlation coefficient was performed to compare each pair of hierarchies on a same dataset, one from centroids generated by *K-means* and one from individual objects. This correlation is calculated between the corresponding members of the cophenetic matrices.

A summary of the experiment settings is shown in Table 1. Considering all setting combinations including, statistical distribution, dataset size, cluster-to-data ratio, distance measure, agglomerative mechanism, and cluster representation, there are total of 960 combinations. Each experiment combination is repeated 50 times for result validation. The set of extended experiments is to further validate the findings from the main experiments. These include enlarging the datasets (up to 10 k), investigating less usual statistical distributions (Exponential and Zipf), and employing different agglomerative approach (UPGMC).

5.2. Results

Since each experiment is repeated 50 times, the least-square method is used for accessing the "acceptability" of the results (Bouguettaya et al., 2002) to ensure that all of the results are credible. This method is not applicable on movie data because the data is pre-existing and therefore the standard variation would be 0.

The results from our experiments are presented in a series of tables. Their key purpose is to show mainly two aspects of our investigation: firstly whether the cluster hierarchy generated by the standard agglomerative approach and that from the 'KnA' method are similar or not by showing the correlation between them, secondly whether the computational cost is reduced by the 'KnA' method. Along that line these tables also show the impact of varies facts on the above aspects, including the ratio of cluster-to-data, distance measures, and agglomerative approaches.

Table 1
Experiment settings.

Parameter	Main experiments	Extended experiments
Distribution	Uniform, Normal	Exponential, Zipf
Dataset size	100 to 600	Up to 10 k
Distance Measure	Euclidean, Canberra	
Algorithm	SLINK, UPGMA	UPGMC
Ratio	0.1 to 1.0	
Cluster Representation	Centroid, Individual Objects	

Fig. 1 shows an example of results which are from the experiments on the data of uniform distribution using Slink as the hierarchical method and Euclidean distance measure. The left sub-figure on the top row is the results from 'KnA' and the right one is for the standard agglomerative method without K-means. The bottom sub-figure shows the time spent on K-means calculation alone during the 'KnA' processes. These figures show the results of 10 data ratio from 0.1 to 1. For each data ratio, the data size gradually increase from 100 to 600. For the sake of comparison, the right sub-figure 'Standard' is in reverse. The *y*-axes on these graphs are time in seconds. One can clearly observe that the K-means component takes much less time compared to the agglomerative component as all measures of K-means in Fig. 1 are less than 1 s while most of the measure for the agglomerative part are more than 20 s. So the computational cost of the first stage K-means component can be ignored.

Furthermore we can see from Fig. 1 that the required time for K-means increases in a linear fashion. However the growth on the top row figures appears quadratically as the complexity of hierarchical algorithms is $O(n^2 lgn)$.

One can clearly observe in comparing the figures that depict the first stage and those that depict the second stage that the time for the first stage is negligible compared to the time needed for the second stage. The time to build clusters seems to grow linearly with a low value for the slope. In contrast, the time to build a tree seems to grow quadratically with the size of the dataset. This is expected as it is known that the computational complexity of this type of hierarchical algorithms is $O(n^2 lgn)$.

A very important observation is that there is a sizeable difference between the execution times using centroids ("KnA", on the left), and all objects ("Standard, on the right), especially when the ratios are smaller. The execution times using centroids are consistently lower than those for all objects method. This gap widens quickly between the two approaches' execution times as the ratio decreases. The only exception is when the ratio is 1. Using either centroids or all objects has the same execution times because all objects approach mutates to the centroid approach where each object is a centroid.

Table 2 gives more detailed results obtained from the data of uniform distribution at 600 data points. The four combinations of UPGMA or SLINK and Eculidean Distance or Canberra Distance are shown in the table for data ratios from 0.2 to 0.9 as the two extremes 0.1 and 1.0 are not that informative in comparison. For each row the time spent by "KnA" and by the standard agglomerative approach are listed. The correlation between the two cluster groups generated by these two methods is also presented in the table. Similar to what is shown in Fig. 1, this table also shows that lower data ratio require much less time compared to the standard counterpart. At ratio 0.2, the difference is way more than 10 times, while at ratio 0.9, KnA is less advantageous (see Table 3).

Another important observation is that the clustering outcome exhibits high similarities between both approaches, especially at high data ratios. At ratio 0.9, the correlations between two cluster groups are around 0.8. Even at ration 0.2, the correlations are around 0.4. That gives a big advantages to the KnA approach because its clustering results are not too far different from that of Standard method yet the cost could be much lower. A user can potential tune this ratio to achieve the idea balance between fast execution and good cluster quality. One thing we need to point out is that specifying a data ratio is different from specifying the *k* value in K-means because the ratio is independent from the actual data itself.

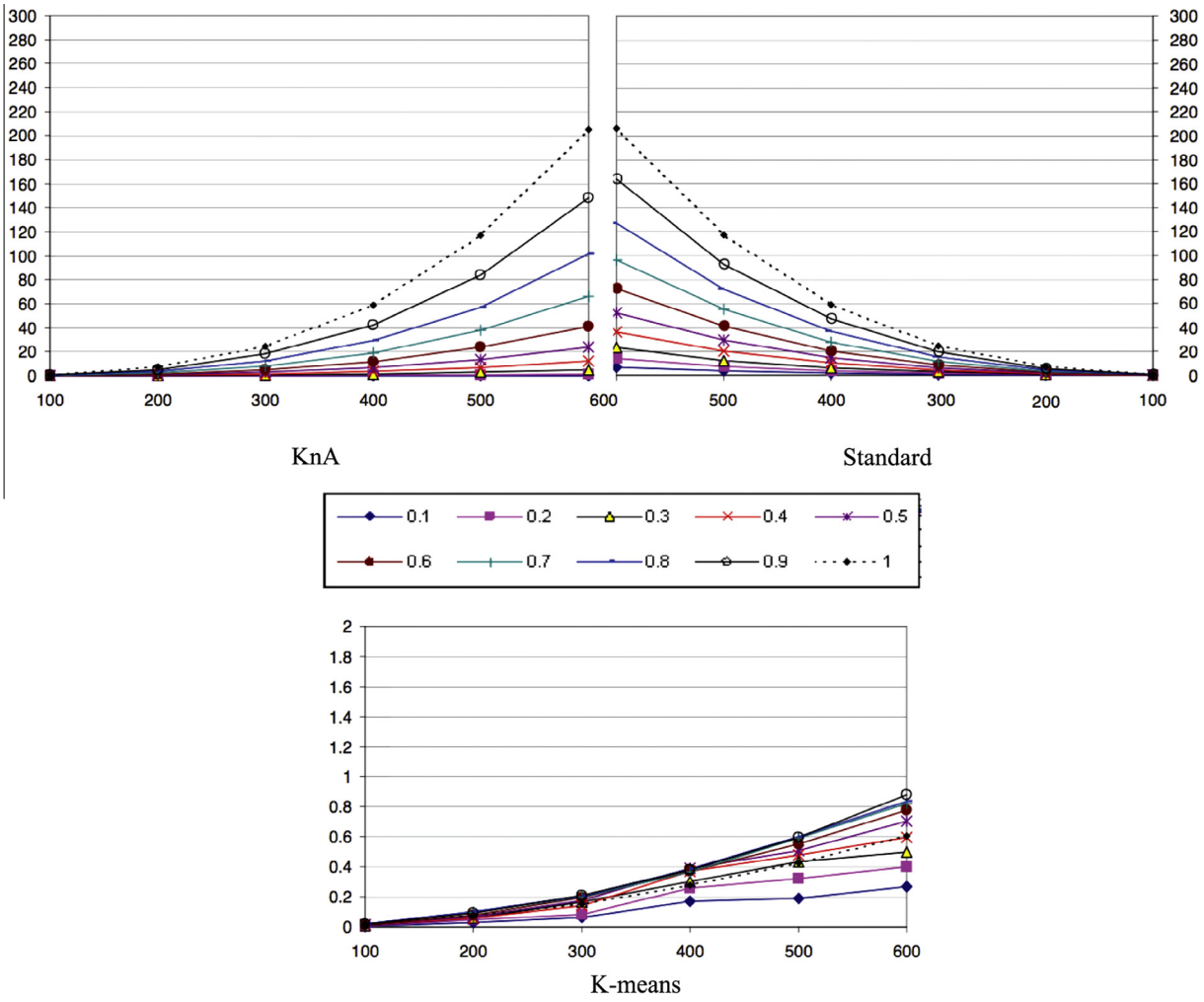

Fig. 1. Example results: K-means followed by Slink using Euclidean distance on Uniform distribution data.

In terms of the effect of UPGMA and SLINK approaches for representing cluster groups, the difference between them are not obvious. UPGMA requires a little more time than SLINK as shown in Table 2. That is understandable because UPGMA needs to compute the averages of all points in the cluster while SLINK only take one. When comparing distance measure functions, Canberra Distance seems better than the Eculidean counterpart. With no noticeable difference in computational cost, correlation achieved by the Canberra method is consistently higher than the correlation obtained by the Eculidean method on a same data set with either UPGMA or SLINK.

The above observations mostly hold true on the results from Normal Distribution Data: lower data ratios cost less computation power and do not negatively affect the clustering qualities, Canberra Distance is in general better than Eculidean Distance. However due to the nature of this data, UPGMA gives worse result than SLINK. Averaging non-centroid points, which is used in UPGMA, could result bias, while that is less likely by SLINK's single point approach. So the SLINK/Canberra combination seems the ideal choice.

When applying the KnA method on real-world data, the outcome is similar to that on synthetic data. Figs. 2–4 illustrates the execution time by KnA and by standard agglomerative method, on various size of data, with different data ratio for the K-mean partition process. The increase of actual execution times with data size is not as smooth as that for synthetic data. This is due to the nature of these data. Because there are identical values for the attributes of different records in the data set, the algorithm must continue searching for centroids if it happens to select a centroid that is identical to one that has already been selected. This may, as a result, increase the execution time. This is especially true if the sample dataset contains a sizeable number of identical values of the considered attributes. Note also that the maximum number of clusters is determined by the number of *different* records. For example, if we have a set of 750 objects and 700 *different* records (considering only those attributes that we are clustering on), the clustering would have to consider 750 objects, although. the maximum number of clusters *K-means* would be 700 clusters (ratio = 1).

When the data ratio for K-means is 1 on these real-world data, the KnA algorithm is also almost equivalent to using a pure hierarchical method. The increase of execution time with data size is also quadratic. That means the standard agglomerative clustering on large data sets can quickly become computation prohibitive, while the running cost for KnA grows much slower especially at low data ratios. In the mean time, as we can see from Figs. 2–4, the performance of KnA is not much different with the standard approach as the correlations between the clusters generated by these two methods are mostly above 0.8. So the advantage of KnA method is evident as it offers a good solution for handling clustering tasks where cost is a concern.

Table 2
Experiments on data of uniform distribution.

	Data ratio	KnA(ms)	Standard(ms)	Correlation
UPGMA/Eculidean	0.2	1	20	0.39
	0.3	3	32	0.41
	0.4	12	42	0.46
	0.5	30	61	0.53
	0.6	52	88	0.62
	0.7	81	121	0.71
	0.8	122	154	0.80
	0.9	178	198	0.87
SLINK/Eculidean	0.2	1	17	0.42
	0.3	2	22	0.44
	0.4	9	38	0.49
	0.5	22	52	0.51
	0.6	41	74	0.55
	0.7	63	96	0.60
	0.8	102	126	0.67
	0.9	149	162	0.78
UPGMA/Canberra	0.2	1	21	0.44
	0.3	4	40	0.47
	0.4	9	60	0.50
	0.5	22	82	0.57
	0.6	41	102	0.66
	0.7	76	132	0.72
	0.8	118	166	0.80
	0.9	151	197	0.88
SLINK/Canberra	0.2	2	18	0.52
	0.3	3	26	0.58
	0.4	10	40	0.61
	0.5	21	53	0.64
	0.6	40	76	0.69
	0.7	61	97	0.74
	0.8	98	128	0.80
	0.9	138	156	0.89

Table 3
Experiments on data of normal distribution.

	Data size	KnA	Standard	Correlation
UPGMA/Eculidean	0.2	1	13	0.47
	0.3	2	25	0.48
	0.4	9	40	0.50
	0.5	23	62	0.51
	0.6	42	82	0.54
	0.7	74	115	0.61
	0.8	114	143	0.65
	0.9	160	181	0.70
SLINK/Eculidean	0.2	1	10	0.78
	0.3	3	18	0.79
	0.4	8	29	0.80
	0.5	20	42	0.82
	0.6	41	61	0.83
	0.7	61	82	0.84
	0.8	94	115	0.85
	0.9	136	144	0.87
UPGMA/Canberra	0.2	2	21	0.62
	0.3	4	40	0.63
	0.4	11	57	0.66
	0.5	23	68	0.68
	0.6	41	100	0.70
	0.7	68	121	0.72
	0.8	113	157	0.78
	0.9	178	182	0.82
SLINK/Canberra	0.2	1	19	0.82
	0.3	3	24	0.83
	0.4	9	37	0.84
	0.5	21	58	0.85
	0.6	40	76	0.86
	0.7	62	98	0.87
	0.8	99	124	0.88
	0.9	141	158	0.91

6. Discussions

6.1. Impact of data distribution

The synthetic data used in this experiment was generated using two types of distribution: *uniform* and *normal* distributions. The results show that the our method is not very sensitive to the distribution of data although distribution does influence the outcome. This is especially the case when SLINK is used in the second phase of the clustering process. To further evaluate how the method react to data distribution, we introduced exponential distribution and Zipf distribution in the extended experiments. We use both UPGMA and SLINK with Euclidean distance. As shown in Figs. 5 and 6, the result is similar as obtained on normal distribution. The correlations are above 0.5 for all cluster-objects ratios after 600 data points. SLINK can better cope with these contributions. In the case of Exponential distribution, SLINK maintains a high correlation across all data ratio even when the ratio is as low as 0.1. The impact of Zipf distribution is similar to that of Normal distribution on our method.

6.2. On movie data

For real-world data, the movie data, the results of the experiments are presented Figs. 2–4, which are the results obtained by using SLINK approach. The results by UPGMA are quite similar in terms of computational cost over different data size with different data ratio. In terms of correlation with the agglomerative methods, those of UPGMA are slightly lower than those of SLINK. Nevertheless from these figures we can see a very strong correlation in almost all cases of cluster-objects ratios. We conjecture that there are two main reasons for this strong correlations. First, a large amount of data in the real world tend to have a normal distribution. Therefore, we expect the results to be similar to those of the synthetic data generated using a normal distribution. Second, taking into account that the gender multi-state attributes have many "0"s that represent the value "absent" and that we use the *Euclidean* distance, we expect the correlation to be high. As we use *Euclidean* distance, all "0"s in the same attributes contribute to the similarity of two objects. Combining these two factors, these centroids would be good representatives of the individual data points in the cluster and therefore for the two resulting cluster groups to have high correlation.

6.3. Data size

When considering the effect of data sizes on the clustering performance, the previous figures clearly show that size itself does not affect the correlation that much especially at a slightly higher data ratio. We observe that the lines on correlation figures are almost flat over the range of data sizes [100–600]. The correlations are consistent across different data size, while the computational cost grow at a quadratic pace but much slower than that of the agglomerative approach.

In our extended experiment, we increase the data size to 1600. As shown in Fig. 7, the results confirm the previous observation since we do not see significant change in results based on dataset size. This also largely confirms the previous results clustering multi-dimensional objects (Bouguettaya et al., 2002).

6.4. Distance type

As mentioned before, we used two types of distance (resemblance measures) for the synthetic data: *Euclidean Distance* and

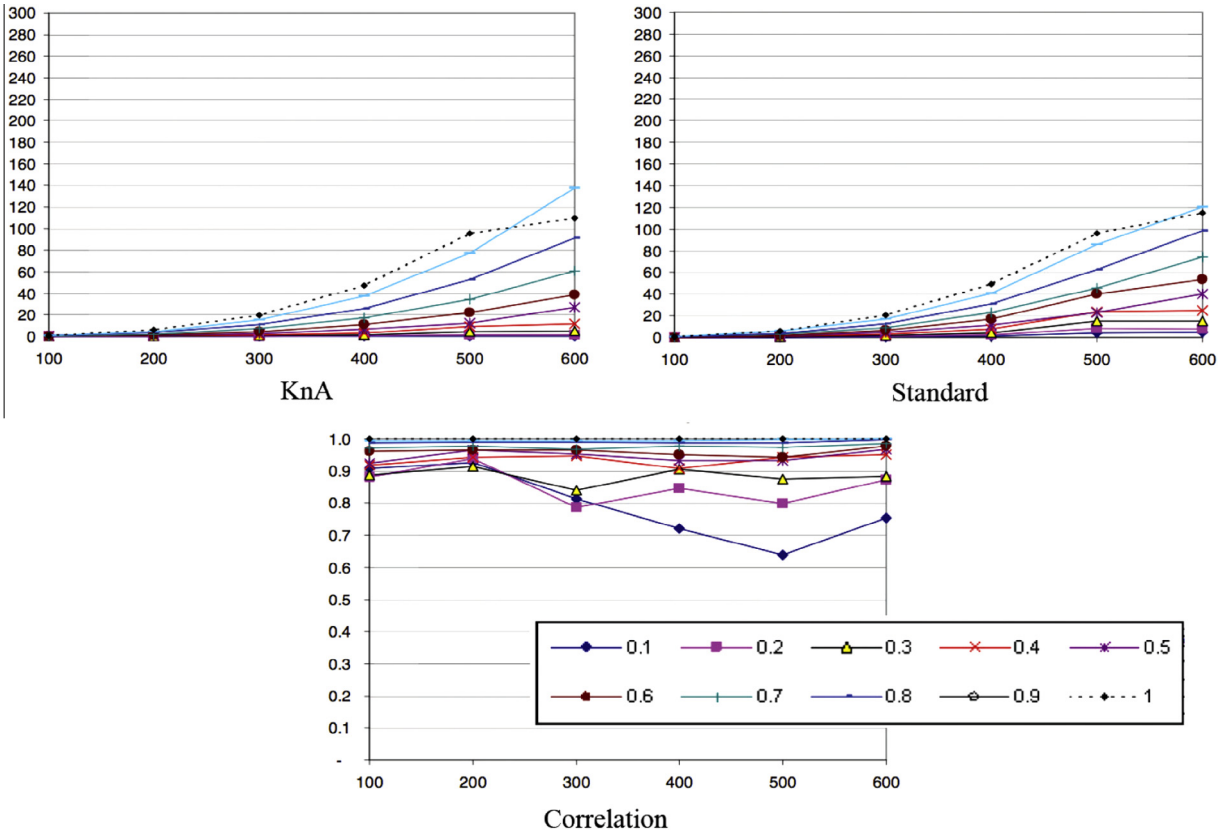

Fig. 2. Results: K-means followed by Slink using Euclidean distance on movie data – random grouping.

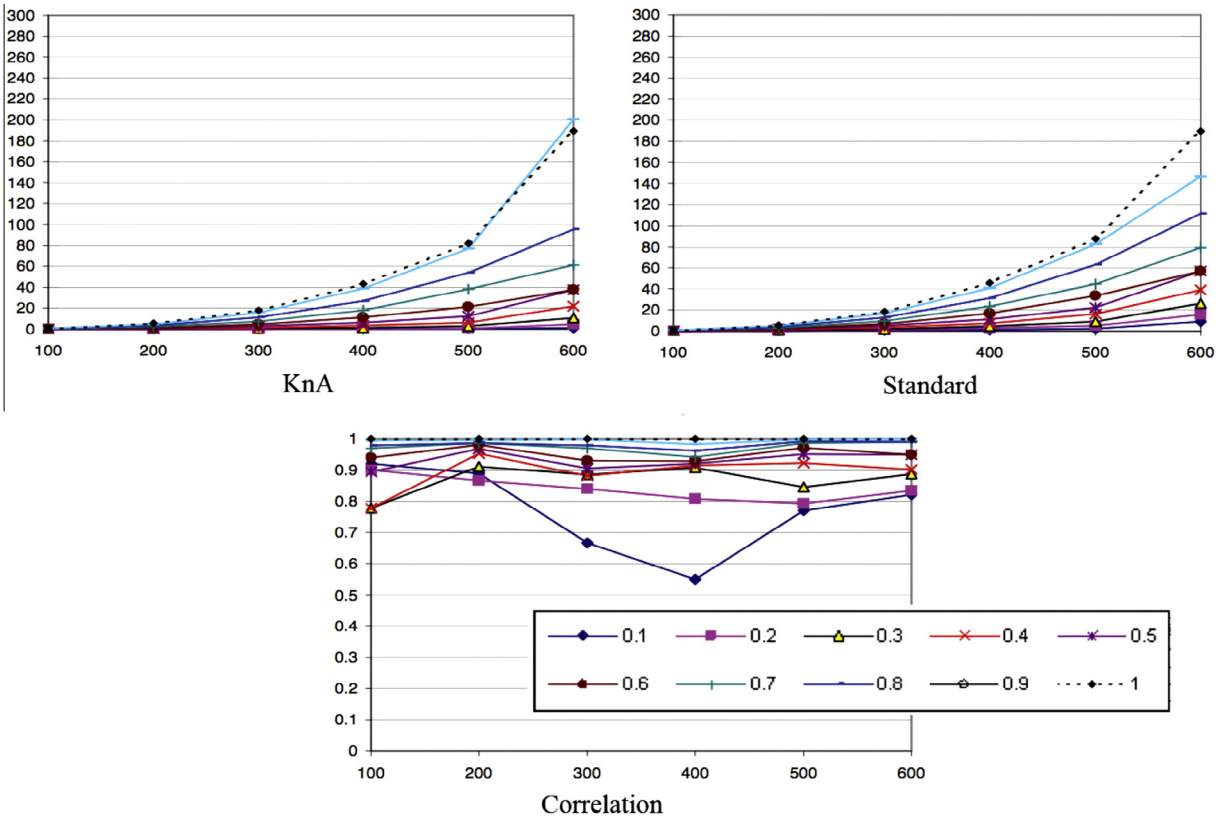

Fig. 3. Results: K-means followed by Slink using Euclidean distance on movie data – occupation.

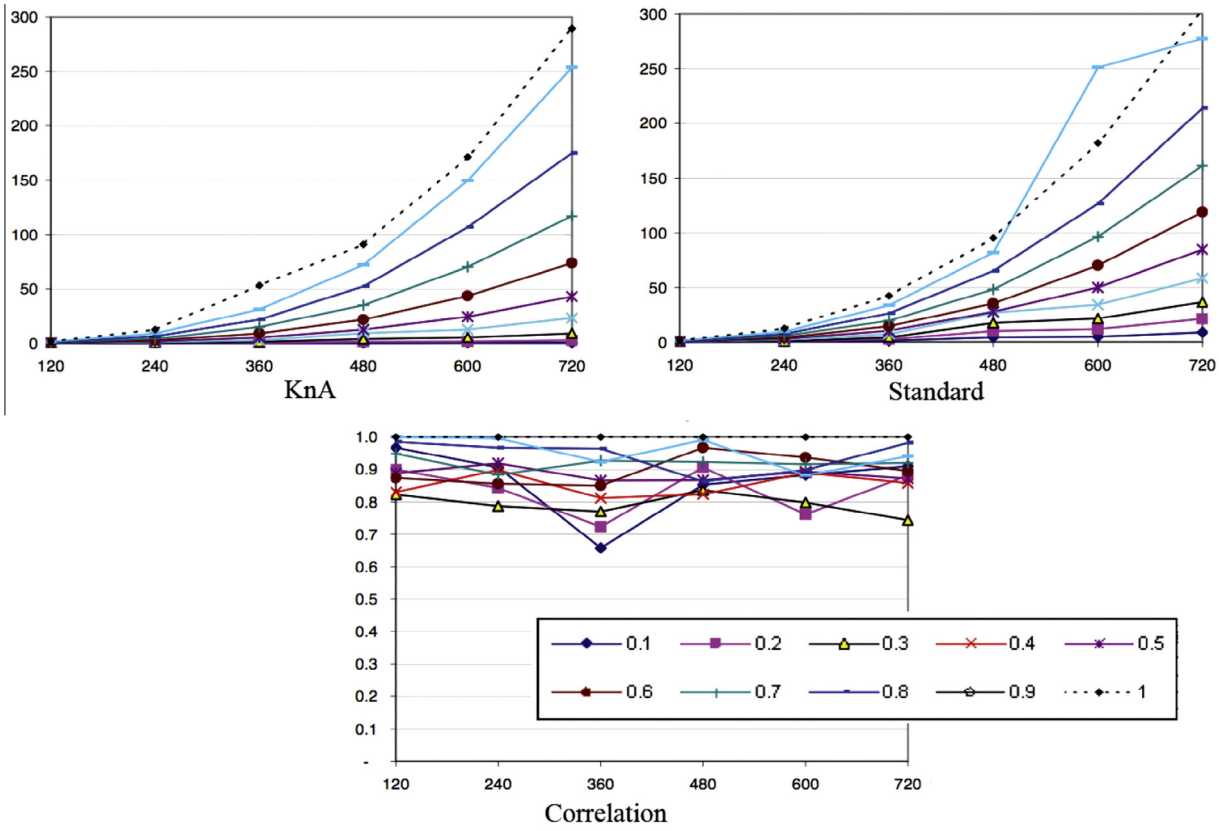

Fig. 4. Results: K-means followed by Slink using Euclidean Distance on movie data – age and occupation.

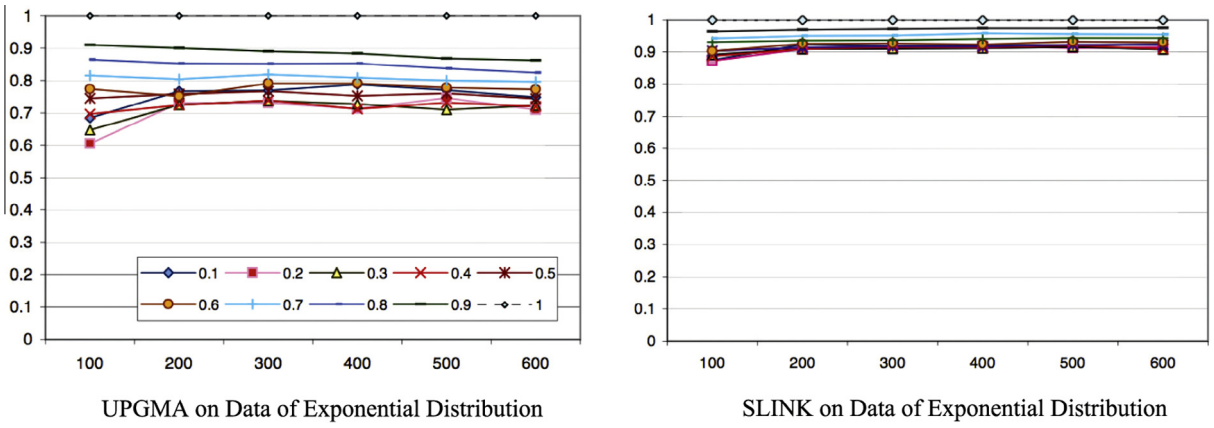

Fig. 5. Experiments on data of exponential distribution.

Canberra Distance. In general they do not have significant difference. The only slight difference is in the results on data of Uniform Distribution. This means that the distance type had little influence on the clustering when the distribution is uniform. The Euclidean and Canberra distances for *Slink* give higher values for the coefficients of correlation than those for *UPGMA* suggesting that the clustering method has a small influence on the type of distance used.

6.5. Agglomerative approaches

While analysing the results from the two methods (i.e., *SLINK* and *UPGMA*), it can be noticed that with normal distribution there is some difference in the correlation levels obtained with SLINK and UPGMA (Table 2, Figs. 5 and 6). The correlation obtained using *SLINK* is slightly higher and more stable than the correlation obtained with *UPGMA*. *SLINK* also seems less sensitive to the initial number of clusters in *K-means*. It shows very little spread between the lines with different ratios. In our extended experiments on the synthetic data, we apply Centroid Linkage (i.e., UPGMC) method with both Canberra and Euclidean distances on uniform data. As shown in Fig. 8, the results give a very strong correlation. This is because UPGMC also uses the centroids of clusters to calculate distances between clusters. Therefore, in the initial several steps of the second phase, using centroids and using all objects will have similar behavior. The difference may come from the later part of the clustering.

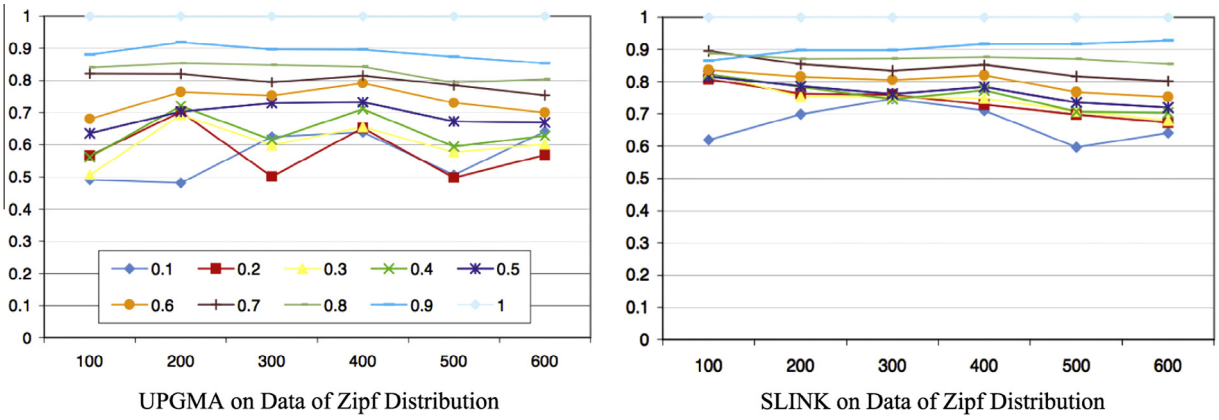

Fig. 6. Experiments on data of Zipf distribution.

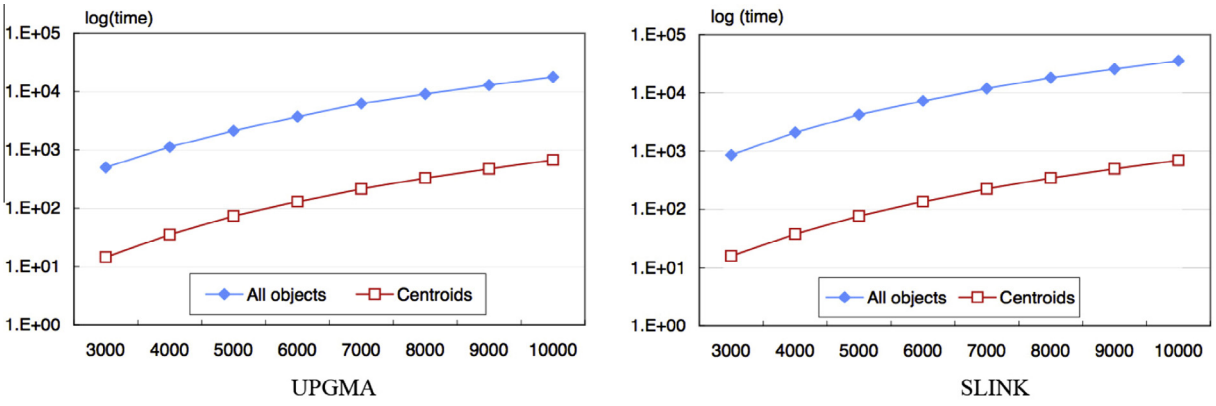

Fig. 7. Runtime on large data sets.

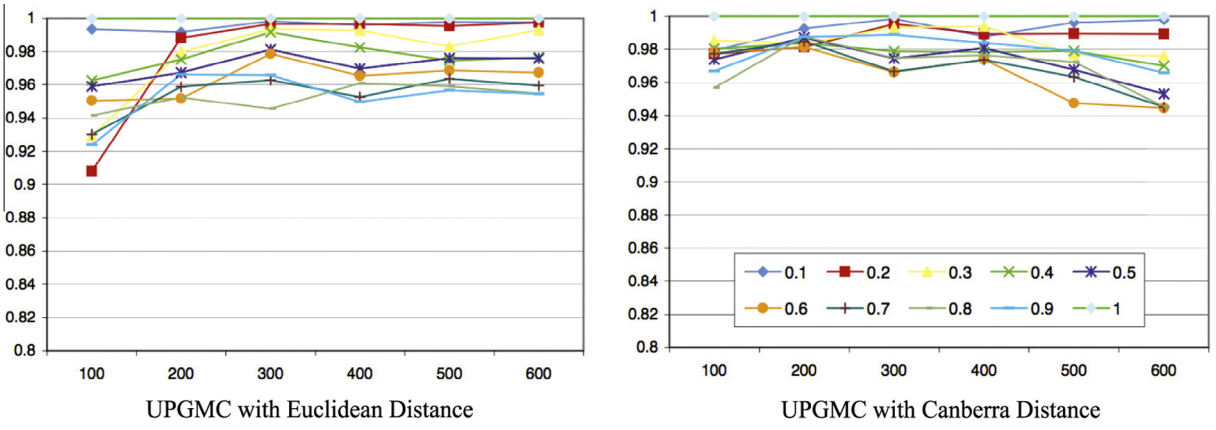

Fig. 8. UPGMC on Data of Uniform distribution using Euclidean distance and Canberra distance.

Fig. 9 illustrates the impact of centroids in UPGMC. Assume that there are four clusters generated from K-means, K_1 to K_4. By using UPGMC, the distances between these clusters are the same when using both centroid and all objects. Therefore, cluster K_1 and K_2 will always be merged initially to generate C_{12}. After that, the distances between clusters may become different between using centroids and all objects. For example, when using centroids, the new centroid of cluster C_{12} is CC whereas the new centroid of C_{12} is AC when all objects are used. In this regard, C_{12} will be merged with K_4 when centroids are used. In contrast, C_{12} will be merged with K_3 when all objects are used.[1]

As for the real-world Movie data, we also observe a slight difference in the behaviors of UPGMA and SLINK. However, the

[1] Maybe need rewritten. SLINK seems a bit better while UPGMA and UPGMC are not far away from it. It further illustrates that the KnA approach is not particularly dependent on a certain approach.

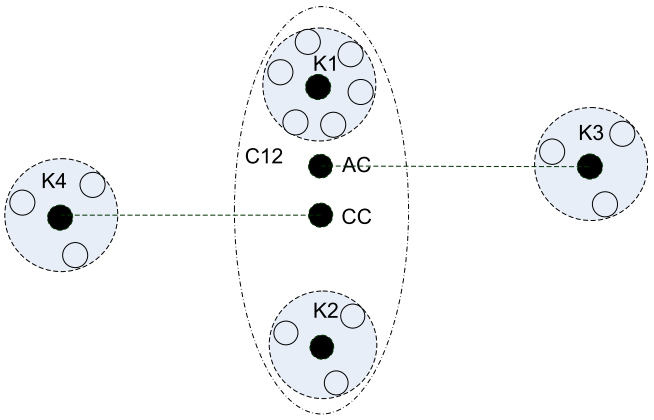

Fig. 9. Behavior of UPGMC.

difference is not as obvious as on the above mentioned synthetic data including normal, exponential, or Zipf distribution.

7. Conclusion

In this paper, we present a methodology, named as KnA method, to improve the efficiency of hierarchical clustering. Instead of clustering on individual data objects, the hierarchy generated by this KnA method is based on a group of sub-clusters created by a partitional method *K-means*. The computational cost reduction of KnA method has been investigated on synthetic data with different distributions as well as on real-world data. The results clearly demonstrate the cost saving introduced by our proposed method. On the other hand the clustering hierarchy created by KnA method does not differ much with that from the standard agglomerative hierarchical clustering method. The set of conducted experiments clearly show that, for the most part, there is high correlation between the trees produced by the proposed method using centroids and the tree produced by using all data points. Therefore the proposed KnA method can be a cost-effective clustering method without losing clustering performance.

To study the applicability of this efficient clustering method, we investigated the impact of different settings such as the type of data, the distribution of the data, the distance measure, the choice of agglomerative methods, the ratio of cluster-to-data. Our experiments show that there is a strong correlation between the use of centroids and the use of all objects with little influence from the cluster-objects ratio when the data distribution follows normal, exponential, or Zipf distribution. That is still the case on data of uniform distribution when the cluster-objects ratios are high. Our experiments also show that the distance type has little influence on the performance of our method (e.g., UPGMA, SLINK, and Centroid Linkage all have different effects on the correlation). Because the correlations between the hierarchies from the KnA method and the standard hierarchical clustering method are consistently high with only a few exceptions, we conclude that the choice of distance measure and the agglomerative methods does not affect the overall performance of our method that much.

The contribution of this work is the proposed KnA method which has computational advantages over hierarchical clustering approaches as it uses centroids rather than raw data points. It reduces the sample space for building the hierarchy hence requires less resources. Therefore the KnA method can potentially analyze larger data sets or operate in a resource limited environment. This performance of this method is not sensitive to the distribution of data set and the distance measure. These strengths make this KnA method widely applicable especially to real-world applications as specific tuning and adjustment are not necessary. In comparison with other methods, the KnA does have weaknesses in certain situations. For example, it is difficult to control the the clusters in KnA when the number of clusters is unknown by domain experts.

This work opens many possibilities for further improvement and investigation. We plan to generalize the findings from this study. For example our future work will cover the impact of other various facts on this KnA approach, including different hierarchical methods (e.g., *Ward*, *Clink*), distributions, and high-dimensional data. By these studies the applicability of the KnA method can be further extended. More real-world data will be tested by this method. Moreover we will compare other hybrid clustering algorithms similar to Cheng et al. (2005).

Acknowledgemet

Funding for this 1 was partly supported by ARC grant for Xiangmin Zhou (DP140100841).

References

Altingövde, I. S., Demir, E., Can, F., & Ulusoy, Ö. (2008). Incremental cluster-based retrieval using compressed cluster-skipping inverted files. *ACM Transactions on Information Systems, 26*(3).

Bouguettaya, A. (1996). On-line clustering. *IEEE Transactions on Knowledge and Data Engineering, 8*(2), 333–339.

Bouguettaya, A., Qi, H., Park, J.-H., & Delis, A. (2002). Wards and upgma clustering of data with very high dimensionality. *Encyclopedia of computer science and technology, 45*.

Cheng, D., Kannan, R., Vempala, S., & Wang, G. (2005). A divide-and-merge methodology for clustering. In *PODS* (pp. 196–205).

Dhillon, I. S., Guan, Y., & Kulis, B. (2004). Kernel k-means: Spectral clustering and normalized cuts. In *KDD* (pp. 551–556).

Guha, S., Rastogi, R., & Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. In *SIGMOD* (pp. 73–84).

Hruschka, E. R., Campello, R. J. G. B., Freitas, A. A., & de Leon F. de Carvalho, A. C. P. (2009). A survey of evolutionary algorithms for clustering. *IEEE Transactions on System man and Cybernetics Part C – Applications and Reviews, 39*(2), 133–155.

Jacinth, Salome J., & Suresh, R. M. (2012). Efficient clustering for gene expression data. *International Journal of Computer Applications, 47*(5), 30–35.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys, 31*(3), 264–323.

Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: A review. *SIGKDD Explorations Newsletter, 6*(1), 90–105.

Lee, J., Han, J., Li, X., & Gonzalez, H. (2008). raClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *PVLDB, 1*(1), 1081–1094.

Lee, J., Han, J., & Whang, K. (2007). Trajectory clustering: A partition-and-group framework. In *SIGMOD* (pp. 593–604).

Li, T., & Ding, C. H. Q. (2006). The relationships among various nonnegative matrix factorization methods for clustering. In *ICDM* (pp. 362–371).

Lin, C., & Chen, M. (2005). Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. *IEEE Transactions on Knowledge and Data Engineering, 17*(2), 145–159.

Lin, C.-R., Liu, K.-H., & Chen, M.-S. (2005). Dual clustering: Integrating data clustering over optimization and constraint domains. *IEEE Transactions on Knowledge and Data Engineering, 17*(5), 628–637.

Liu, G., Li, J., Sim, K., & Wong, L. (2007). Distance based subspace clustering with flexible dimension partitioning. In *ICDE* (pp. 1250–1254).

Liu, H., & Yu, L. (2005). Toward integating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering, 17*(4), 491–502.

Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing, 17*(4), 395–416.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849–856). MIT Press.

Ordonez, C., & Omiecinski, E. (2004). Efficient disk-based k-means clustering for relational databases. *IEEE Transactions on Knowledge and Data Engineering, 16*(8), 909–921.

Pan, F., Zhang, X., & Wang, W. (2008). CRD: fast co-clustering on large datasets utilizing sampling-based matrix decomposition. In *SIGMOD* (pp. 173–184).

Project, G. R. (n.d.). <http://www.cs.umn.edu/Research/GroupLens/index.html>.

Rokach, L. (2010). A survey of clustering algorithms. In *Data Mining and Knowledge Discovery Handbook* (pp. 269–298).

Romesburg, H. (1990). *Cluster analysis for researchers*. Malabar, FL: Krieger Publishing Company.

Shalom, A., & Dash, M. (2013). Efficient partitioning based hierarchical agglomerative clustering using graphics accelerators with cuda. *International Journal of Artificial Intelligence and Applications, 4*(2), 13–33.

Wattanachon, U., Suksawatchon, J. & Lursinsap, C. (2009). Nonlinear data analysis using a new hybrid data clustering algorithm. In *PAKDD* (pp. 160–171).

Wu, O., Hu, W., Maybank, S. J., Zhu, M., & Li, B. (2012). Efficient clustering aggregation based on data fragments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 42*(3), 913–926.

Xu, R., & Wunsch, D. (2010). Clustering algorithms in biomedical research: A review. *IEEE Reviews in Biomedical Engineering, 3*, 120–154.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). Birch: an efficient data clustering method for very large databases. *SIGMOD Record, 25*(2), 103–114.

Zhao, Y., Yu, J., Wang, G., Chen, L., Wang, B., & Yu, G. (2008). Maximal subspace coregulated gene clustering. *IEEE Transactions on Knowledge and Data Engineering, 20*(1), 83–98.

Zhou, X., Zhou, X., Chen, L., Shu, Y., Bouguettaya, A., & Taylor, J. A. (2009). Adaptive subspace symbolization for content-based video detection. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1372–1387.