A Novel Fast Heuristic to Handle Large-Scale Shape Clustering

Hassan Ismkhan
Department of Computer Engineering
University of Bonab
Bonab, East Azerbaijan, Iran
H.Ismkhan@bonabu.ac.ir

ABSTRACT. Clustering algorithms like types of k-means are fast, but they are inefficient for shape clustering. There are some algorithms, which are effective, but their time complexities are too high. This paper proposes a novel heuristic to solve large-scale shape clustering. The proposed method is effective and it solves large-scale clustering problems in fraction of a second.

Keywords: clustering, large-scale shape clustering, automatic cluster number, heuristics

1. Introduction. Let \( U = \{x_1, x_2, \ldots, x_n\} \) be a set of \( n \) objects and each object is characterized by the \( m \) number of attributes: \( x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\} \). The aim of clustering is to group these objects into some subsets in such a way that objects in the same subset (called a cluster) are more similar to each other than to those in other groups (clusters). The similarity between two objects usually is calculated by Euclidian- or Hamming-Distance where each object is implemented by a vector: \( x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m}) \).

There are many clustering algorithms \cite{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Almost all these clustering algorithms try to partition \( U \) into the \( k \) clusters so that it minimizes the below function with unknown \( W \) and \( V \):

\[
F(W, V) = \sum_{i=1}^{k} \sum_{j=1}^{n} W_{ij} ||x_j - v_i||^2
\]  

Subject to

\( W_{ij} \in \{0, 1\}, \sum_{i=1}^{k} W_{ij} = 1, 0 < \sum_{j=1}^{n} W_{ij} < n, 1 \leq i \leq k, 1 \leq j \leq n, \)

where

- \( W = [w_{ij}]_{k \times n} \) matrix of \( \{0, 1\}, w_{ij} \) is a binary variable, and indicates whether object \( x_j \) belongs to the \( i \)th cluster, \( w_{ij} = 1 \) if \( x_j \) belongs to the \( i \)th cluster and 0 otherwise;
- \( ||x_j - v_i||^2 \) is Euclidean distance between the object \( x_j \) and the \( i \)th cluster center \( v_i \).

For \( m = 2 \) where the number of attributes is 2, the problem is named shape clustering. The shape-clustering problem may be simple as Figure 1 and it may be too complicated as Figure 2.
The k-means algorithm is one of the clustering algorithms. The types of k-means have been enhanced in recent years [11] [12]. However k-means algorithm is not suitable for complicated instances like Figure 2. Some heuristic-based clustering algorithms like [6], [3], [13], [14] may be suitable for such problems. Nevertheless, for all these clustering algorithms, the number of clusters should be determined by users in advance. There are some methods to estimate the number of clusters [15], [16], [17]. However, determining the number of clusters is not always straightforward.

This paper proposes a heuristic algorithm which automatically prognosticates the number of clusters and operates fast. Proposed algorithm solves some large-scale instances in fraction of a second. This algorithm is a type of merge-based algorithm which uses simple information of existing clusters and decides which pair of clusters are suitable to be merged. The proposed heuristic is based on statistical information which are simple but effective and unlike other similar algorithms [19] [6], this heuristic operates fast. Therefore, this paper is organized as section 2 reviews some recent clustering algorithms, section 3 describes proposed method, section 4 describes experimental results and section 5 summarizes this paper.
2. **Brief Review of Clustering Algorithms.** The k-means is one of the most famous clustering algorithms. Figure 3 shows the k-means algorithm. In formal view, the k-means tries to minimize F function in equation 1 by repeating the following instructions: (1) it minimizes F by constant V. (2) Updates V (3) minimizes F for obtained W (now w is constant) (4) updates W. The initial points of the start of k-means are important and have many effects on the convergence speed and results of k-means. There are some proposals, which suggest how to select the initial points for k-means [20], [21], [1]. Some types of proposals try to accelerate k-means [22], [23], [24] and [25].

The k-means is fast and its time complexity is linear, but as stated in previous section, it is not suitable for complicated shape clustering. Some clustering algorithms have been based on repeating the merge process [19] [6] [26]. Figure 4 shows the main skeleton of these algorithms. The Chameleon heuristic [6] is one of these algorithms. It follows the algorithm in Figure 4. The Chameleon probes suitability of merging a pair of clusters according to relative interconnectivity and relative closeness between a pair of clusters. The relative interconnectivity is defined based on the sum of the weight of the edges that straddle the two clusters and relative closeness is defined based on the average weights of the edges that belong in the min-cut bisector of a pair of clusters. Like Cure and DBScan, Chameleon needs too many parameters and its time complexity is rather high.

The removing noise points or sampling an efficient prototype is common in many clustering algorithms [25], [27] and [3]. The proposed method by [3] extracts skeleton of an input dataset by two operators including Globbing and Object Movement. After some iterations of an evolutionary process, this algorithm reaches to the main skeleton of an input dataset. However, it needs to compute the nearest neighbors of each point in all iterations.

The evolutionary algorithms and other types of natural-inspired based algorithms include the differential evolutionary (DE) algorithm [4], gen-based or genetic algorithms [28] [2], artificial bee colony (ABC) algorithm [5], ant-based algorithm [29], particle swarm
optimization (PSO) algorithm [30] [31] and gravitational search based algorithm [32] have been applied to the clustering problems. The hybrid of natural-base algorithms with the other types of algorithm like k-means for clustering is common, too [2] [7]. Some heuristics for clustering have been based on minimum spanning tree (MST) [13] [14] [33]. Although these algorithms have been successfully applied to the shape clustering, their time complexities are high.

This paper proposes a heuristic merge-based algorithm which is easy to implement and it can handle complicated large-scale dataset in acceptable runtime. In addition, this algorithm automatically determines the number of clusters. Moreover, unlike Cure, DBScan and Chameleon which need too many parameters, proposed algorithm only needs one parameter which is the number of nearest neighbors.

3.1. The Main Skeleton of Proposed Method. The main skeleton of the proposed algorithm is similar to Figure 4. The main difference between these algorithms is in the stopping criterion. Figure 5 shows that the proposed algorithm goes to the end of the algorithm without considering the number of k. In fact, for the proposed algorithm, the number of k is not an input and the algorithm decides to terminate merging process automatically then it automatically determines the number of clusters. This algorithm has two main phrases. In the first phase, algorithm creates some clusters then in the second phase it looks up for the suitable pair of clusters and merges the pair into the one cluster.

3.2. Initial Phase. Initial phase creates initial clusters by algorithm in Figure 6. In this section knn(i) points to the nearest neighbor list of point i and knn(i, j) points to the j’th of the nearest neighbor of i.
3.3. **Merging Phase.** In fact, this sub-section describes when a pair of cluster is merged or which pair is suitable to be merged in a one cluster. However, before describing more details, we need some definitions.

**Definition 3.1. Critical Point (CP):** The point $p$ is critical point $\iff \exists q \mid q \in \text{knn}(p)$ and $\text{cluster}(p) \neq \text{cluster}(q)$. Here, the cluster $(p)$ and cluster$(q)$ refer to clusters which contain the point $p$ and $q$ respectively.

```plaintext
assign each point to a cluster;
determine k nearest neighbor(knn) of each point; //here k is a parameter
continue = true;
while(continue)
do
  continue = false;
  for each cluster i
do
    for each point j in cluster i
do
      for each nn in knn(j)
do
        if(j and nn are not in same cluster)
          merge j’ cluster with nn’ cluster
          continue = true;
        end if
      end for
    end for
  end for
end while
```

**Figure 6.** First phase that creates initial clusters
**Definition 3.2. Point Density (PD):** The PD\(_i\) is the average distance of the k’th of the nearest neighbor of point i from itself (from the i).

\[ PD_i = \frac{1}{k} \sum_{r \in \text{nn}(i)} \text{distance}(i, n) \]  

(2).

In this equation, k is the number of nearest neighbors.

**Definition 3.3. Black List (BL):** The point p is in black list if the PD\(_p\) > Average-PD where average-PD is computed according to the following equation.

\[ \text{Average - PD} = \frac{1}{n \ (\text{the size of dataset})} \sum_{i \in \text{points in dataset}} PD_i \]  

(3).

Now, with these definitions we can state the details of the algorithm and determine the stopping criterion of the algorithm. The stopping condition is determined by the following equation:

\[ \text{CPs - BL} = \emptyset \]  

(4).

When the stopping criterion is not satisfied then algorithm merges the corresponding clusters of points in the set CPs – BL.

4. **Empirical Results.** All experiments were performed on Pentium 4 laptop with 2.4 GHz CPU and 3.00 GB of random access memory (RAM). All programs were coded by standard C++ language and operating system was Windows 7.

To show the accuracy of the proposed algorithm, it has been applied to the two types of datasets. The datasets of the first type are in small-scale but the second type includes large-scale datasets. While the other types of the famous clustering algorithms like k-means need the number of the clusters, the proposed algorithm automatically estimates the number of the clusters. This is a special advantage for the proposed algorithm. To compete with the proposed algorithm, we selected k-means. In experiments, for the k-means algorithm, the number of the clusters was determined manually before the start of the corresponding program. For the proposed algorithm, the number of computed nearest neighbors for each point is 5 and to speed up the computation, the k-d-tree structure has been used.

The first type of datasets is small. This category includes two dataset that the first is simple and but the second one is complicated. Figure 7 shows the details of these datasets.
Figure 8. Result for dataset 1. (A) Result by k-means. (B) Obtained result by proposed algorithm. The determined number of clusters is 2.

Figure 9. Result for dataset 2. (A) Obtained result by k-means. (B) Obtained result by proposed algorithm. The determined number of the clusters is 5. Comparing these results show that k-means is not suitable algorithm for instances like this.

The results for dataset of the first type have been shown in Figure 8 and 9. It can be seen that the results produced by the proposed algorithm are better than k-means.

Figure 10 shows the second type of datasets with details. The produced results by k-means and proposed algorithm are in Figures 11 and 12.

Figure 10. Large datasets [6]. (A) Dataset 3: number of points: 8000, number of clusters: 6. (B) Dataset 4: number of points: 10000, number of clusters: 9

Figure 11 shows that both k-means and proposed algorithm can solve dataset 3 easily. This dataset is large but it is too simple.
Figure 11. Result for dataset 1 (A) Result by k-means. (B) Result by proposed algorithm.

Figure 12 shows that the result of k-means for dataset 4 is not competitive with proposed algorithm.

For all the tested datasets, the proposed algorithm could gain better results. Its accuracy for complicated datasets also was high. However, the k-means algorithm slightly is quicker than the proposed algorithm. Table 1 shows the average runtime (30 runs) of each algorithm. As shown in this table, for dataset 4, the runtime of proposed algorithm is better than k-means.

<table>
<thead>
<tr>
<th></th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
<th>Dataset 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>0</td>
<td>0</td>
<td>62</td>
<td>437</td>
</tr>
<tr>
<td>Propose algorithm</td>
<td>16</td>
<td>16</td>
<td>297</td>
<td>406</td>
</tr>
</tbody>
</table>

Additional datasets
To show the ability of the proposed algorithm, two other large-scale datasets have been used. The details of these datasets are in Figure 13.
Figure 13. Additional datasets (large scale). (A) Dataset 5: number of Points: 3100, number of clusters: 31 [36]. (B) Dataset 6: number of points: 8000, number of clusters: 6 [6].

Figure 14 shows the result of proposed algorithm for datasets in Figure 13. In this experiment, for the dataset 6, the number of nearest neighbors for each point was 20.

Figure 14. The Results of the proposed algorithm for dataset 5 and 6

5. Conclusions. This paper proposes a novel heuristic for clustering. The proposed algorithm in this paper is based on hierarchical merging process. This algorithm merges a pair of clusters in all iterations and continues this process until no suitable pair is found. This algorithm not only is applicable to the large-scale problems, also solves the complicated large-scale shape clustering. In addition, there is no need to the number of clusters because this algorithm automatically can determine the number of clusters.

Acknowledgment. The author gratefully acknowledge the helpful comments and suggestions of the associated editor and reviewers, which have improved the presentation.

REFERENCES


**Hassan Ismkhan**
Instructor of Data Structures in University of Bonab (at East Azerbaijan)