Acta Cybernetica **15**(2001) 241–256.

SmallSteps: An Adaptive Distance-based Clustering Algorithm

Gy. Koch* and J. Dombi*

Abstract

In this article we propose a new distance-based clustering algorithm. Distance-based clustering methods operate on data sets that are in similarity space, where the similarities/dissimilarities between the objects are given by a matrix. These algorithms have at least $O(n^2)$ time complexity, where n is the number of objects. One of the latest distance-based method is Chameleon which, according to experiences, works well only on larger data sets and fails on relatively smaller ones. This contraditcs the fact that the $O(n^2)$ time complexity makes the distance-based algorithms unsuitable for huge data sets. Thus we developed a new distance-based method (SmallSteps), which can handle relatively small amount of objects too. In our solution we are looking for connected graphs which have edges with a maximum weight computed on the environments of the objects. The method is capable to detect clusters with different shapes, sizes or densities, it is able to automatically determine the number of clusters and has a special ability to divide clusters into subclusters.

1 Introduction

Clustering is one of the most commonly used statistical methods. It can be seen as an unsupervised machine learning process where the algorithm has to divide a set of objects ($X = \{x_1, \ldots, x_n\}$) into different classes ($C = \{C_1, \ldots, C_k\}$) such a way that similar objects should be in the same class while dissimilars should be in different classes. These classes are called clusters. Clustering algorithms are used to determine the underlying structure of the object set. Often it is useful not to create statistical measures or perform tasks on the whole set of objects, but after a cluster analysing, doing it to the different clusters having similar objects which gives more accurate results. For example one can discover more proper connections between the features of cars if he/she first clusters the data and examines the cars with low top speed and low consumption (city cars) and cars with moderate top speed and very high consumption (luxury cars) separately, rather than handles all sorts of cars together.

*University of Szeged, Hungary e-mail: [koch, dombi]@inf.u-szeged.hu

This aim of clustering is usually described as maximizing the function [10]

$$Q_S^D(X,C) = Q_S(X,C) + Q^D(X,C)$$

where $Q_S(X,C)$ means the similarity between objects in the same cluster and $Q^D(X,C)$ the dissimilarity between the objects in different clusters. There are no exact mathematical formulas for $Q_S(X,C)$ or $Q^D(X,C)$ that could be acceptable for most of the cluster analysing tasks.

Dividing objects into two groups by minimizing the maximum distance in the clusters, can be done by bicoloring a maximum spanning tree in $O(n^2)$ steps. On the other hand dividing the objects into more than two clusters is an NP hard problem [1]. Although segmenting into more than two partitions can be done by sequentially dividing clusters into two, it often does not give optimal solutions and fails on very simple examples, for instance when we want to partition these objects into 3 groups (see Figure 1).

Figure 1: Problems with the multiple bipartition when dividing into two groups.

Detecting the number of clusters is also a very difficult problem, most of the clustering algorithms can only divide the objects into a number of clusters given by the user. Even there are special cases when appropriate clustering does not exist or the only good clustering is to order all of the objects into one cluster (e.g. integer coordinate pairs of the 2-dimensional space).

These are the main reasons why heuristical approaches are so popular among clustering techniques.

1.1 Two Main Types of Clustering Methods

Considering practical use there are two well-separable kinds of cluster analysing methods depending on the type of problems they have to solve.

- In the first group there are the faster algorithms having $O(n)$ complexity. These methods usually take the objects as points in the d-dimensional space (if the objects have d attributes) so we will refer to this group as coordinate-based methods. These methods can handle huge datasets with hundreds of thousands or even millions of records (e.g. calls of a telephone company, web log of an on-line store, shopping transactions of a supermarket, transfers of a bank,...etc.). Due to their quickness (note that clustering with these methods is faster than sorting the objects or finding the two closest/most similar objects) these algorithms give a rough segmentation and mostly recognize only spherical clusters. Using this group of clustering algorithms the user

usually has to give the number of clusters a priori. Most known representatives of this group are the Fuzzy C-Means [2], [11], [12] and the Kohonen Clustering Network [2].

- Methods in the second group have much lower speed, they have at least $O(n^2)$ complexity. These algorithms work on the distances/dissimilarities between the objects, which explains their time complexity. Since they are much slower than the coordinate-based ones, they are only good for smaller tables and most of these algorithms have to store the distance matrix, so their memory consumption can be quite large. The advantage of these methods is that they produce much better results. They may detect clusters with arbitrary shapes or sizes and determine the exact number of clusters. These algorithms are very closely related to shape recognition. For example they can recognize the arcs of a detached double spiral. The first distance-based methods were the agglomerative and divisive methods [1], [4] and one of the latest is HCS [5] which is also a graph theoretic approach but instead of using weighted edges HCS concentrates on edge-connectivity. One of the best algorithms in this group of methods is Chameleon, which was published in 1999 [7]. Chameleon has a very powerful recognizing capabilities, it detects clusters with arbitrary shapes and determines the number of clusters needed, still it has some serious drawbacks in practical use.

To emphasize the gap between the algorithms belonging to the two different groups let us show some calculations. Consider that a clustering algorithm segments 1000 objects in the 10-dimensional space in one second. If this algorithm belongs to the first group of methods, clustering 1 million objects takes approximately 17 minutes and with single precision real number representation it requires approximately 38 megabytes of memory while if the algorithm belongs to the distance-based group clustering 1 million records takes more than 1.5 weeks and the size of the distance matrix is 1.8 terabytes.

In this paper we will propose a new cluster analysing algorithm for the second group of methods. SmallSteps is a distance-based method and overcomes the difficulties of Chameleon while it keeps all the good features of it.

2 The Way Chameleon Works

Chameleon is one of the latest developed distance-based clustering method, which was published in 1999 by Karypis et al. [7]. It takes the objects as vertices of a graph with the weighted edges according to the k nearest neighbour graph on similarities between the objects. The weights of the edges are the similarity values. Chameleon has two main phases. In the first phase it creates small sub-clusters and merges them together into clusters in the second phase.

The sub-cluster creating part is done by a hypergraph-partitioning algorithm. Since partitioning a graph into a large number of equally sized subgraphs is an

Figure 2: Situation when the hypergraph-partitioning phase in Chameleon fails.

NP hard problem, Chameleon uses a heuristic technique called multilevel graph partitioning [8], [9].

To merge these sub-clusters Chameleon calculates special measures. The relative closeness is responsible to merge only clusters that have uniform density among the objects in the same cluster and relative inter-connectivity is for maintaining the similar inter-connectivity in the clusters.

Chameleon can work according to two different schemes.

- In the first scheme Chameleon introduces two technical parameters as thresholds. One for the relative closeness and one for the relative inter-connectivity. Pairs of clusters, whose calculated measures are above these thresholds, will be merged. Chameleon may terminate if there are no pairs of clusters whose relative closeness and relative inter-connectivity is above the thresholds or these parameters may be relaxed during the merging phase allowing Chameleon to create only one big cluster.

- According to the second scheme Chameleon uses a function to combine the relative closeness and relative inter-connectivity. This function is usually has the form

$$f(C_i, C_j) = RI(C_i, C_j) * RC(C_i, C_j)^{\alpha}$$

where $RI(C_i, C_j)$ and $RC(C_i, C_j)$ are the relative inter-connectivity and relative closeness between clusters C_i, C_j and α is a user specified parameter

to increase the importance of one of the two measures. After computing the goodness (f) of merging of all pairs of clusters, Chameleon combines the clusters with the best goodness value. Then the algorithm updates the $RI(C_i, C_j)$ and $RC(C_i, C_j)$ values and continues with the cluster pair selection. The result of this scheme is that we get one big cluster and the order of the mergings.

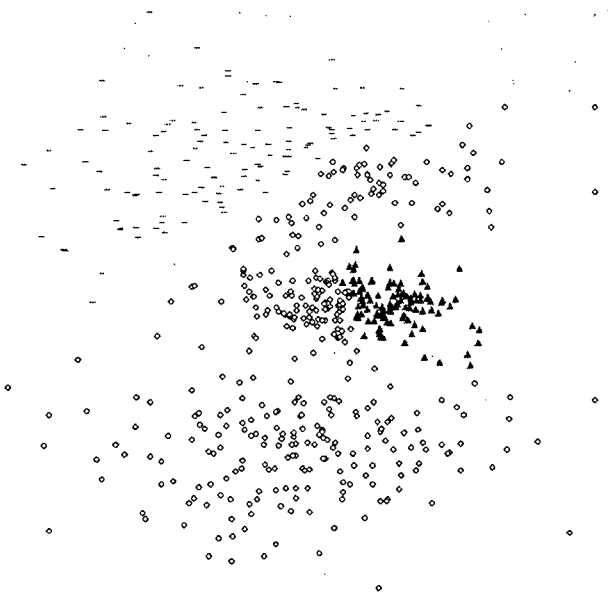

Figure 3: The result of Chameleon starting from wrong division, like on Figure 2.

The Chameleon method can handle nearly arbitrary shape of clusters and can detect quasi-automatically the number of clusters.

Experiments show that if the number of records is low, Chameleon works not well. The problem is that, with the heuristical graph partitioning algorithm, Chameleon at first divides the objects into a large number of relatively small subclusters which still have to be big enough to correctly compute their internal measures (e.g. the internal inter-connectivities which are used to compute the relative inter-connectivity between two clusters). If there are only small numbers of elements (i.e. less than 1000), very often one or more sub-clusters have intersections with more than one real cluster (see Figure 2). Since it has no error correction, this means that these clusters will be connected and the algorithm cannot separate them later (see Figure 3). Remember that methods in the distance-based group are best for relatively small tables.

On Figure 2 two sub-clusters (marked with 1 and 2) have common part with two genuine cluster. Cluster marked with 3 consists of objects of two separate groups

which indicates that the random part of the hypergraph-partitioning phase tried only wrong partitions. This last one occurs rather rarely, but plays major role in getting the wrong clustering of Figure 3.

Another problem with Chameleon is that it needs two technical parameters to detect the number of clusters, which are very hard to interpret by the user and in practical use it is a serious disadvantage.

Chameleon's graph partitioning algorithm is a non-deterministic procedure, which implies that the whole method is also non-deterministic.

Mainly these problems above inspired us to develop a new distance-based clustering method which is able to overcome these difficulties and gives a result of the same high quality as Chameleon does.

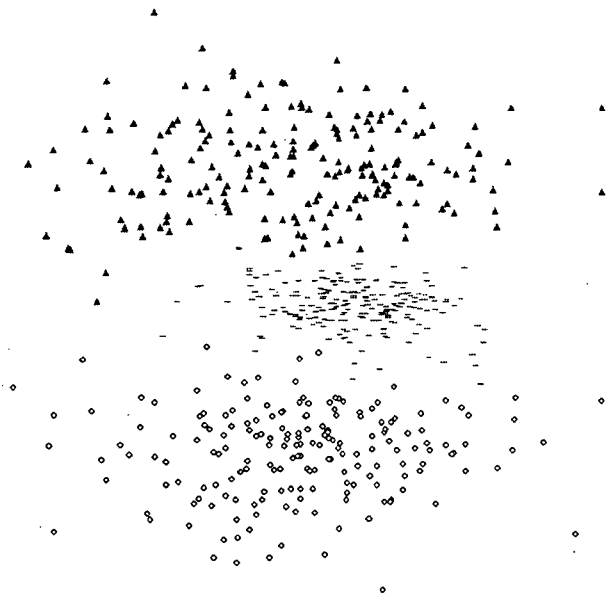

Figure 4: The result of SmallSteps on the object set of Figure 2 and Figure 3.

The developed SmallSteps

1. can be used for datasets of different size: from small to relatively large ones with up to 10000 records;

2. it needs no parameters, still it automatically detects the number of clusters and can recognize clusters with any shape and;

3. always gives the same result for a given set of objects.

3 SmallSteps: The New Procedure

3.1 The Algorithm of SmallSteps

The solution of SmallSteps, similarly to Chameleon's, comes from graph theory. The objects are considered as vertices of a graph and the distances between the objects are the weighted edges of the graph. The clusters are special connected graphs having edges with weights less than a cluster depended threshold called δ. These thresholds are recalculated in every iteration (see below) and unique to every cluster.

first phase	(1) order all elements into one cluster and calculate its δ (2) iterate $t = 1, \ldots, t_{max}$ (3) sort the clusters according to their δ (4) form the new clusters with the δ's (5) calculate the new δ's
second phase	(6) check whether merging clusters is possible
third phase	(7) iteratively process the oulier elements according to the selected strategy (8) check whether a new cluster can be form with the current outlier element (9) if not, order it to one of the existing clusters

Table 1: The algorithm of SmallSteps.

In the following part denote the number of clusters in the t^{th} iteration step by $|C^{(t)}|$. Denote the k^{th} cluster in the t^{th} iteration step by $C_k^{(t)}$ and denote its δ parameter by $\delta_k^{(t)}$ where $1 \leq k \leq |C^{(t)}|$. Finally denote the number of objects in cluster $C_k^{(t)}$ by $|C_k^{(t)}|$.

SmallSteps has three main phases.

- The first phase is the initial cluster-forming phase. At the beginning of this phase ($t = 0$) all the elements belong to one cluster ($C_k^{(t)}, k = 1$) and the initial δ ($\delta_k^{(t)}$) is calculated. Then in an iterative process the algorithm begins to search for special connected graphs with edge weights less than $\delta_k^{(t)}$ and the thresholds of the newly created clusters are recalculated. We repeat this process until a previously given number of iteration steps is reached (see Figure 5, Figure 6 and Figure 7). The δ's are responsible for creating clusters of upwardly bounded, nearly uniform densities, so it is very important to start every iteration with clusters having smaller δ's otherwise neighbouring clusters with smaller densities (higher δ's) would incorporate the denser ones.

The $\delta_k^{(t)}$ for the k^{th} cluster in the t^{th} iteration step is calculated based upon the average distance between the objects of the cluster $C_k^{(t)}$ and their closest neighbours from the same cluster.

$$\delta_k^{(i)} = \frac{\sum_{x_j \in C_k^{(t)}} f\left(\frac{1}{r}\sum_{l=0}^{r} d(x_j, x_{j(l)})\right)}{|C_k^{(t)}|}$$

where $x_{j(l)}$ is the l^{th} closest element to x_j and $d(x_j, x_{j(l)})$ is the distance/dissimilarity between x_j and $x_{j(l)}$. Here r is the degree of δ, i.e. the number of neighbours involved in the calculation and f is a monotone function which determines the way the average distance is taken into consideration. In Figure 4, 5, and 6 f was the same linear transformation.

Figure 5: The clusters found after the 1st iteration step in the first phase.

The degree of the δ's (r) controls the way SmallSteps works. If only one or a few neighbours are considered then the result of SmallSteps is very close to the result of a shape recognition algorithm.

Clusters are not always segmented into sub-clusters during this first phase, but merging is also possible still it is done very rarely.

The time complexity of this phase is $O(n^2 t_{max})$, where n is the number of objects, t_{max} is the number of cluster forming iterations and it is independent of the number of objects.

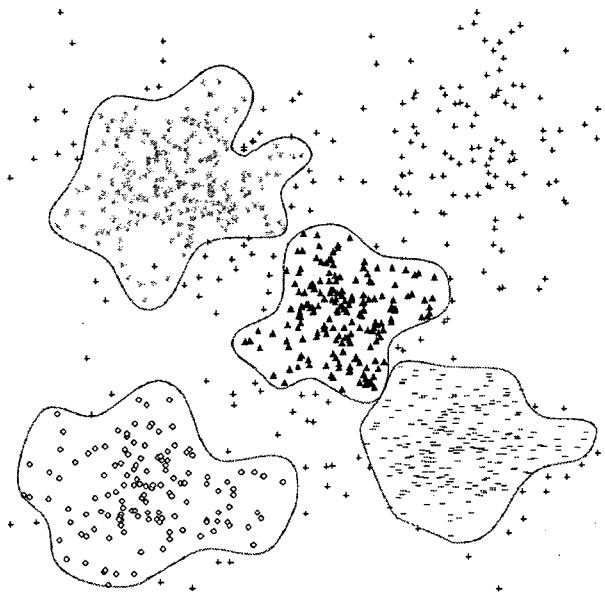

Figure 6: The clusters found after the 2nd iteration step in the first phase.

- The second phase is a merging phase. It makes decisions on merging some of the initially formed clusters based on their δ, size and distance. Although the first phase of SmallSteps is so powerful that usually there is no need to merge clusters in the second phase, our experiments showed that these merging calculations are worth to do.

  Merging objects has $O(|C^{(t_{max}+1)}|^2)$ time complexity, where $|C^{(t_{max}+1)}|$ is the number of clusters created during the first phase.

- The handling of outlier or noisy objects is done in the third phase (see Figure 8). Often datasets have outlier or noisy objects, which does not belong essentially to any clusters and could be left uncategorized, but in most cases the user wants to order all of the objects into clusters. If the user accepts outlier objects then this third phase should be skipped.

  Depending on the structure of outlier elements, in SmallSteps the following two operations can be done with them:

  - Let them form new clusters.
    If there is a group of outlier objects that are far enough from the existing clusters and have enough elements which are close to each other to form a new connected subgraph with edge weights less than their special δ then these objects are allowed to create a new cluster.
  - Order them to existing clusters.

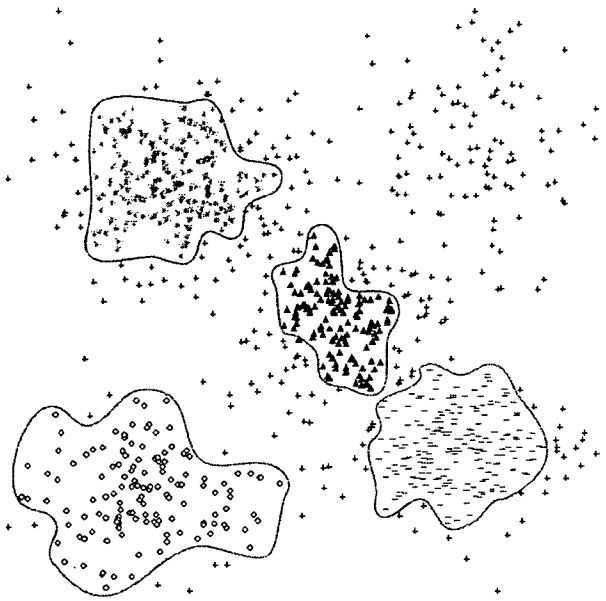

Figure 7: The clusters found after the 5^{th} iteration step in the first phase.

The order of outlier objects influence the result so we have developed different strategies:

- The simplest is to choose always those outlier objects that are the nearest to an existing cluster.
- We may follow a postponing strategy and choose those objects first whose classification are the easiest, i.e. which are close to a cluster but far from the other clusters.
- According to the BestFit method those outlier objects are always classified first whose distance to a cluster best fit to the δ of the cluster. This method tries to maintain the uniform density of the clusters.

The time complexity of the last phase is $O\left(n_{\text{outlier}} * \max\{n_{\text{outlier}}, n_{\text{classified}}\}\right)$, where n_{outlier} and $n_{\text{classified}}$ are the numbers of outlier and classified elements at the beginning of the third phase, respectively.

The overall time complexity of SmallSteps is

$$O\left(n^2 t_{max} + |C^{(t_{max}+1)}|^2 + n_{\text{outlier}} * \max\{n_{\text{outlier}}, n_{\text{classified}}\}\right) = O(n^2).$$

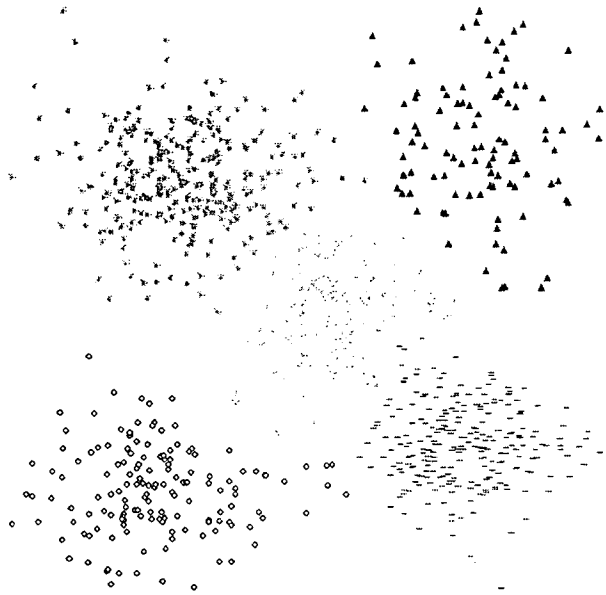

Figure 8: The clusters found after the outlier handling phase.

3.2 Handling Bounding Objects

If two clusters are in touch by only a few objects then by searching for connected graphs the algorithm will find that these clusters could form one cluster since there is a path between the elements of the two clusters with edges less than the δ's of the clusters. To avoid such aggregation it is useful to find the boundary objects of the clusters and if two clusters are connected with only bounding objects then the algorithm should not merge these clusters. The algorithm can recognize the bounding objects by counting their intra-cluster neighbourhood which is the number of objects from the given cluster that are closer to the object than the δ of the cluster. The bounding objects have fewer neighbouring objects than the inner ones.

3.3 Inner Analysis of a Cluster

SmallSteps provides a useful additional feature. If the user wants to analyse a cluster in more detail, he/she can specialize this cluster by segmenting it into sub-clusters. The segmentation is done by iteratively decreasing the original δ of the cluster and searching for connected graphs with edge weights less than its new δ. The specializing procedure terminates when the algorithm is able to segment the cluster into sub-clusters of acceptable size or when it turns out that such a segmentation is not possible (see Figure 9).

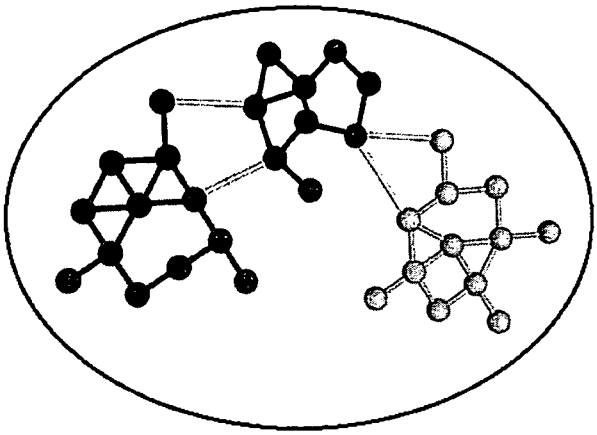

Figure 9: Segmenting a cluster into sub-clusters.

4 Example

On Figure 5, Figure 6, Figure 7 and Figure 8 the different stages of SmallSteps is shown during clustering.

Figure 5 shows the situation after the first iteration step of the first phase. The beginning one big cluster fell into 3 sub-clusters framed with thin lines. The objects marked with "+" signs are considered as outlier objects in this step. Note that one of the sub-clusters, having objects marked with "-" signs, still contains two genuine cluster.

After the second iteration step of the first phase (see Figure 6) the wrongly merged clusters broke up and the final clusters began to take form.

The following iteration steps in the first phase only refine the borders of clusters found in the second step. In this example after the 8^{th} step these refinements stop and no change is made in the subsequent iteration steps. On Figure 7 the division after the 5^{th} iteration step is shown.

The core of four of the five genuine cluster is detected in the first phase and there is no need to merge clusters in the second phase.

The detection of the last cluster is done in the third phase during the outlier handling process (see Figure 8). In this phase the algorithm detected the possibility of forming a new cluster from the outlier objects and the objects not involved in this cluster were incorporated into one of the existing cluster.

5 Results

We tested SmallSteps on numerous sample databases, four of them can be seen on Figure 8, Figure 10, Figure 11 and Figure 4.

The test shown on Figure 8 was performed on 1000 objects from 5 clusters taken from 5 normal distribution. The clusters have different densities with 100 to 300 objects.

The test of Figure 10 contained 1024 objects in 6 clusters of highly different densities. The two clusters on the bottom of Figure 10 are dense clusters connected with a rarer zone, but this zone is still denser than the other clusters.

On Figure 11 the result of a test on 2000 objects is shown. The nearly all of the clusters are only vertical lines with outlier objects on the endings. The objects were taken from the abalone database from the Repository of Machine Learning Databases and Domain Theories maintained by the University of California at Irvine.

The sample database of Figure 4 contained 600 objects taken from 3 normal distribution, all of them having 200 objects. It is a very difficult problem because of the high noise and the elliptical clusters. We also tested 3 coordinate based methods on the sample sets and none of them could solve this problem adequately.

In Table 2 the running times on the different tests are shown. The values are in milliseconds and measured on a 550MHz Intel Pentium III machine with 128MB RAM. Each test was performed 20 times.

In the first column the measured time of the full SmallSteps algorithm are shown while in the second column we skipped the outlier handling phase. We implemented Chameleon and the running times of the implementation is shown in the third column. Testing Chameleon is done by creating 30 sub-clusters with the hypergraph-partitioning algorithm and merging them into the given number of clusters according to the second scheme.

Test	SmallSteps	SmallSteps without 3^{rd} phase	Chameleon
Figure 4 (avg)	152.023	142.467	422.554
Figure 4 (σ)	1.682	5.532	9.530
Figure 8 (avg)	416.872	367.205	984.818
Figure 8 (σ)	0.985	4.772	16.762
Figure 10 (avg)	356.773	338.504	1050.196
Figure 10 (σ)	1.619	0.941	27.605

Table 2: Test results of SmallSteps and Chameleon.

6 Summary

In SmallSteps, the number of clusters evolves automatically during the three phases (mainly during the first phase) hence no user interaction is needed for giving this number. Instead of relative inter-connectivity and relative closeness, SmallSteps calculates δ's to perform cluster forming. These δ's can be computed from only a few objects, which means that SmallSteps works well on smaller tables too.

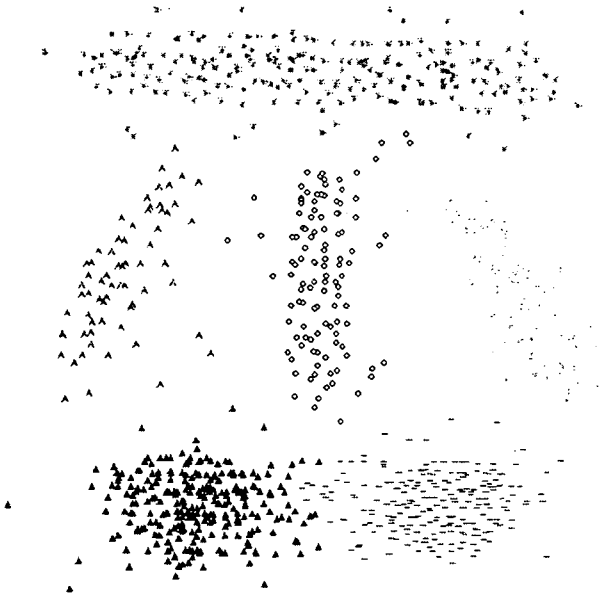

Figure 10: Detecting elliptical clusters with different densities and sizes.

SmallSteps can recognize clusters with different densities (see Figure 10) because every cluster has its own δ and since SmallSteps searches for connected graphs it can recognize clusters with arbitrary shapes or sizes (see Figure 11).

The algorithm of SmallSteps is deterministic so it will give always the same output for a given set of objects.

Since the outlier or noisy objects are handled in the last phase and till then they are eliminated from the processing by the first few iteration steps of the first phase SmallSteps is not too sensible to this kind of objects (see Figure 4).

While Chameleon is a kind of greedy, agglomerative clustering procedure and never corrects the errors made during its merging process, SmallSteps rather resembles to a divisive clustering method, but it has merging steps too and it has some error-correcting feature in all the three phases. Practical experiments showed that, among hierarchical algorithms without error correction, the divisive methods usually outperformed the agglomerative ones because divisive methods needed fewer steps to create the segmentation [4].

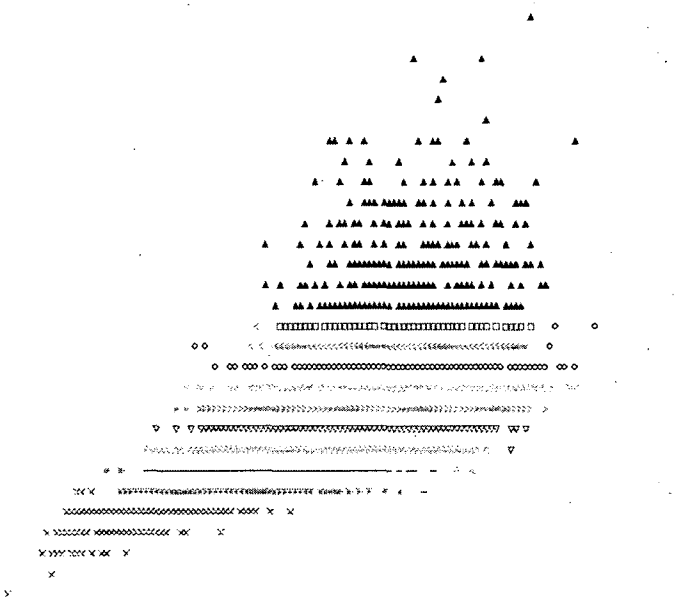

Figure 11: SmallSteps is very close to shape detection.

References

[1] Charles J. Alpert and Andrew B. Kahng: *Splitting An Ordering into a Partition to Minimize Diameter*, Journal of Classification, 14 (1997) 51-74.

[2] James C. Bezdek, Erik Chen-Kuo Tsao and Nikhil R. Pal: *Fuzzy Kohonen Clustering Networks*, Pattern Recognition, 27 (1994) 757-764.

[3] Rajesh N. Davé and Raghu Krishnapuram: *Robust Clustering Methods: A Unified View* IEEE Transactions on Fuzzy Systems, 5 (1997) 270-293.

[4] A. Guénoche, P. Hansen and B. Jaumard: *Efficient Algorithms for Divisive Hierarchical Clustering with the Diameter Criterion*, Journal of Classification, 8 (1991) 5-30.

[5] Erez Hartuv and Ron Shamir: *A clustering algorithm based on graph connectivity* Information Processing Letters, 76 (2000) 175-181.

[6] A. K. Jain, M. N. Murty and P. J. Flynn: *Data Clusterin: A Review* ACM Computing Surveys, 31 (1999) 264-323.

[7] George Karypis, Eui-Hong Han and Vipin Kumar: *Chameleon: Hierarchical Clustering Using Dynamic Modelling*, IEEE Computer, 32 (1999) 68-75.

[8] George Karypis and Vipin Kumar: *Analysis of Multilevel Graph Partitioning*, Technical Report TR 95-037, University of Minnesota, Department of Computer Science, 1995.

[9] George Karypis and Vipin Kumar: *Multilevel k-Way Partitioning Scheme for Irregular Graphs*, Journal of Parallel and Distributed Computing, 48 (1998) 96-129.

[10] Jan W. Owsiński: *Clustering - modelling, capabilities, limits, applications*, Control and Cybernetics, 24 (1995).

[11] Mika Sato, Yoshiharu Sato and Lakhmi C. Jain: *Fuzzy Clustering Models and Applications*, Physica-Verlag Heidelberg, 1997, ISBN 3-7908-1026-6.

[12] Ching-Chang Wong and Chia-Chong Chen: *A Clustering Based Method for Fuzzy Modeling*, IEICE Transactions on Information and Systems, E82-D (1999) 1058-1065.