5. Джерри, А. Дж. Теорема отсчетов Шеннона, ее различные обобщения и приложения [Текст] / Дж. Джерри // Обзор. ТИИЭР. - 1977. - Т. 65, № 11. - С. 53-89.
6. Petre, Stoica. Spectral analysis of signals [Текст] / Petre Stoica, Randolph L. Moses. - Pearson Prentice Hall. 2005. - 452 с.
7. Кравченко, В.Ф. Лекции по теории атомарных функций и некоторым их приложеним [Текст] / В.Ф. Кравченко. — М.: Радиотехника, 2003.- 512 с.
8. Спектральные свойства атомарных функций в задачах цифровой обработки сигналов [Текст] / В. Ф. Кравченко, М. А. Басараб, Х. Перес-Меана // Радиотехника и электроника. - 2001. - Т.46, №5. - С. 534-552.
9. Fractals and the Fractal Dimension [Електронний ресурс] // Vanderbilt University official website. - Режим доступу: \www/ URL: http://www.vanderbilt.edu/AnS/psychology/cogsci/chaos/workshop/Fractals.html/ -22.11.2012.
10. Shezafs, N., Abramov-Segals, H., Sutskovers, I. Adaptive low complexity algorithm for image zooming at fractional scaling ratio [Електронний ресурс]// The Signal and Image processing Lab.-Режим доступу:\www/ URL:http://www-sipl.technion.ac.il/new/Teaching/Projects/Zoom_Article.pdf/ - 10.11.2012.

***Зусилля були спрямовані на пошук методів для ефективної та точної кластеризації великих баз даних. В основному теми дослідження зосереджені на масштабованості кластерних методів, ефективності методів кластеризації для складних форм і типів даних, багатомірних методах кластеризації, а також методах кластеризації змішаних чисельних і категоріальних даних у великих базах даних***

***Ключові слова: база даних, поєднання у кластери, k-NN граф, гіперграф, сусід, багатофункціональний***

***Усилия были направлены на поиск методов для эффективной и точной кластеризации больших баз данных. В основном темы исследования сосредоточены на масштабированности кластерных методов, эффективности методов кластеризации для сложных форм и типов данных, многомерных методах кластеризации, а также методах кластеризации смешанных численных и категориальных данных в больших базах данных***

***Ключевые слова: база данных, объединение в кластеры, k-NN граф, гиперграф, сосед, многофункциональный***

УДК 665.9

# A MULTILEVEL APPROACH TO THE DYNAMIC HIERERCHICAL CLUSTERING FOR COMPLEX TYPES OF SHAPES

**T. Shatovska**
Associate Professor*
E-mail: shatovska@gmail.com
**A. Zaremskaya***
E-mail: NastenkaZar@yandex.ua
*Department of Software Engineering
Kharkiv National University of Radioelectronics
Lenina, 16, Kharkov, Ukraine, 61166

## 1. Introduction

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively as one group in many applications. Data clustering is under vigorous development. Contributing areas of research include data mining, statistics, machine learning, spatial database technology, biology, and marketing. Owing to the huge amounts of data collected in databases, cluster analysis has recently become a highly active topic in data mining research. As a branch of statistics, cluster analysis has been studied extensively for many years, focusing mainly on distance-based cluster analysis. Active themes of research focus on the scalability of clustering methods, the effectiveness of methods for clustering complex shapes and types of data.

Chameleon is a clustering algorithm that explores dynamic modeling in hierarchical clustering. In its clustering process, two clusters are merged if the interconnectivity and closeness between two clusters are highly related to the internal interconnectivity and closeness of objects within the clusters. The merge process based on the dynamic model facilitates the discovery of natural and homogeneous clusters and applies to all types of data as long as a similarity function is specified. Chameleon is derived based on the observation of the weakness of two hierarchical clustering algorithms: CURE and ROCK. CURE and related schemes ignore information about the aggregate interconnectivity of objects in two different clusters, whereas ROCK and related

schemes ignore information about the closeness of two clusters while emphasizing their interconnectivity. In this paper, we present our experiments with hierarchical clustering algorithm CHAMELEON for circles cluster shapes with different densities using hMETIS program that used multilevel k-way partitioning for hypergraphs and a Clustering Toolkit package that merges clusters based on a dynamic model. In CHAMELEON two clusters are merged only if the interconnectivity and closeness between two clusters are comparable to the internal inter-connectivity of the clusters and closeness of items within the clusters.

The methodology of dynamic modeling of clusters is applicable to all types of data as long as a similarity matrix can be constructed. We present a modified hierarchical clustering algorithm that measures the similarity of two clusters based on a new dynamic model with different shapes and densities. The merging process using the dynamic model presented in this paper facilitates discovery of natural and homogeneous not only circles cluster shapes.

## 2. Overview of Chameleon dynamic clustering

Chameleon is a hierarchical clustering algorithm that explores dynamic modelling in hierarchical clustering [KHK, 99a]. Chameleon represents its objects based on the commonly used k-nearest neighbour graph approach. This graph representation of the data set allows CHAMELEON to scale to large data sets. Each vertex of the k-nearest neighbour graph represents a data object, and there exists an edge between two objects if one object is among the k-most similar objects of the other. The k-nearest neighbour graph captures the concept that neighbourhood radius of an object is determined by the density of the region in which this object resides [Mi, 97]. This tends to result in more natural clusters, in comparison with density-based methods such as DBSCAN that instead use a global neighborhood. Moreover, the density of the region is recorded as the weight of the edges.

During the next step a sequence of successively smaller hypergraphs are constructed – Coarsening Phase. Two primary schemes have been developed for selecting what groups of vertices will be merged together to form single vertices in the next level coarse hypergraphs. The first scheme called edge-coarsening (EC) [Al, 97] selects the groups by funding a maximal set of pairs of vertices (i.e., matching) that belong in many hyperedges.

The second scheme that is called hyperedge-coarsening (HEC) [KAK, 97] finds a maximal independent set of hyperedges, and the sets of vertices that belong to each hyperedge becomes a group of vertices to be merged together. At each coarsening level, the coarsening scheme stop as soon as the size of the resulting coarse graph has been reduced by a factor of 1.7 [KHK, 99b].

The third phase of the algorithm is to compute a k-way partitioning of the coarsest hypergraph such that the balancing constraint is satisfied and the partitioning function as mincut is optimized. During the fours phase - uncoarsening phase, a partitioning of the coarser hypergraph is projected to the next level finer hypergraph, and a partitioning refinement algorithm is used to optimize the objective function without violating the partitioning balancing constraints.

At the final iteration of algorithm CHAMELEON determines the similarity between each pair of clusters by taking into account both at their relative inter-connectivity and their relative closeness. It selects to merge clusters that are well inter-connected as well as close together with respect to the internal inter-connectivity and closeness of the clusters. By selecting clusters based on both of these criteria, CHAMELEON overcomes the limitations of existing algorithms that look either at the absolute inter-connectivity or absolute closeness.

## 3. Multilevel approach to the dynamic hierarchical clustering

As we remark above the CHAMELEON operates on a sparse graph in which nodes represent data items and weighted edges represent similarities among the data item (symmetric graph) [KHK, 99a].

In our algorithm during first phase we construct an asymmetric k-NN graph and there exists an edge between two points if for one of it there exist closest neighbour among all existing neighbours according to the value of k. Note that the weight of an edge connecting two objects in the k-NN graph is a similarity measure between them, as usual a simple distance measure (or inversely related to their distance).

In our algorithm the weight of an edge we compute as weighted distance between objects and as more similar two objects as heavier weight of an edge connecting them. It is very useful for outlier detection. The main goal of k-NN graph representation is to define regions of a maximal set of density-connected objects as the initial shapes of clusters and separate noise and non similar points. The Fig. 1 (Fig 1. The k-NN graph “disk in disk” ) represents the k-NN graph for data set “disk in disk” with k=5 number of neighbours (it means that each point will has no less than k neighbours) and Fig.2 with k=15 (Fig 2. The k-NN graph “Men’s face”) represents a general overview of how the data are structured and connected in this case. As we can see the obtain shapes as dense regions of objects that are separated by regions of low density. During our experiments we show that it isn’t important the values k for computing the initial k-nearest neighbour graph.

During next coarsening phase the set of smaller hypergraphs are constructed. In the initial stage we randomly choose a set of vertices and matched it together. In the next stage of coarsening process we choose the set of vertices with maximum degrees and matched it with a random neighbour. On the other stages we visit each vertex in a random order and matched it with adjacent vertex via heaviest edge. Note that usually the weight of an edge connecting two nodes in a coarsened version of the graph is the number of edges in the original graph that connect the two sets of original nodes collapsed into the two coarse nodes [KHK, 99b].

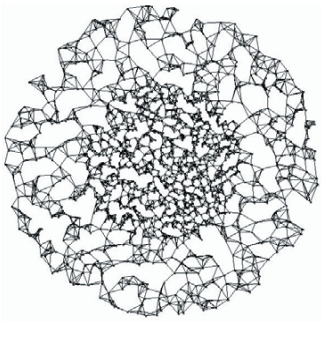


Fig. 1. The k-NN graph “Disk in disk”

Fig. 2. The k-NN graph "Men's face"

In our case we compute the weight of the hyperedge as the sum of the weights of all edges that collapse on each other during coarsening step. We stop the coarsening process at each level as soon as the number of multiservices of the resulting coarse hypergraph has been reduced by a constant less then 2. The main structure of k-NN graph isn't destroyed during coarsening process, but the number of points was reduced. We finally stop this step when the total number of objects is no less then 1200.

On the next level of algorithm we produce a set of small hypergraphs using k-way multilevel paradigm [KHK,1999b]. We start the process of partitioning by choosing k most heavier multiservices, where k can be 8, 16, 32.

After that we gathering one by one all neighbours from each previously chosen most heavier vertex and obtain the initial partitioning w.r.t the balancing constant. Balancing constant is a maximal number of multiservices in each part of the partitioning.

The problem of computing an optimal bisection of a hypergraph is NP-hard.

One of the most commonly used objective function is to minimize the hyperedge-cut of the partitioning; i.e., the total number of hyperedges that span multiple partitions [Karypis, 1999b]. One of the most accuracy algorithm of partitioning the hypergraph is Kernighan-Lin / Fiduccia – Mattheyses algorithm, in which during each pass, the algorithm repeatedly finds a pair of vertices, one from each of the subdomains, and swaps their subdomains [Fiduccia, 1982]. The pairs are selected so as to give the maximum improvement in the quality of the partitioning even if this improvement is negative. Once a pair of vertices has been moved, neither are considered for movement in the rest of the pass. When all of the vertices have been moved, the pass ends. At this point, the state of the bisection at which the minimum edge-cut was achieved is restored.

In our approach we use a main idea of greedy refinement algorithm developed by George Karypis [KK, 99b], but with some extensions. In the first stage of this process we calculate a gain function for each multiservice. But we suggest instead of classic view to compute a gain criteria of each vertex in hypergraph as differences between the sums of the weights of edges incident on vertex that go to the other partition and the of edges weights that stay within the partition. We choose the vertex with maximum positive gain and move it if it result in a positive gain, so we works only with boundary vertices. After this step we obtain k parts of small hypergraphs and implement for each part a simple bisection algorithm to find an optimal min-cut bisection. After that we project the partition to the previous level finer hypergraph and refine partition using an iterative scheme. In both the KL and KL (1) refinement algorithms, we have to insert the gains of all the vertices in the data structures [KKb, 98].

We use the idea of the boundary Kernighan-Lin refinement algorithm, where we initially insert into the data structures the gains for only the boundary vertices. As in the KL refinement algorithm, after we swap a vertex, we update the gains of the adjacent vertices not yet being swapped. If any of these adjacent vertices become a boundary vertex due to the swap of vertex, we insert it into the data structures if they have positive gain. Notice that the boundary refinement algorithm is quite similar to the KL algorithm, with the added advantage that only vertices are inserted into the data structures as needed and no work is wasted.

After the partitioning of hypergraph into the large number of small parts we start to merge the pair of clusters for which both relative inter-connectivity and their relative closeness are high. In our research we use George Karypis formula to compute the similarity between sub-clusters [KHK, 99a].

The relative inter-connectivity between a pair of clusters Ci and Cj is defined as the absolute inter-connectivity between Ci and Cj normalized with respect to the internal inter-connectivity of the two clusters Ci and Cj . The absolute inter-connectivity between a pair of clusters Ci and Cj is defined to be as the sum of the weight of the edges that connect vertices in Ci to vertices in Cj [KHK, 99a]. The relative closeness between a pair of clusters Ci and Cj is defined as the absolute closeness between Ci and Cj normalized with respect to the internal closeness of the two clusters Ci and Cj [KHK,99a].

But mentioned above approach has some limitations – it doesn't evaluate in a full way the similarity between clusters by it's densities and in each iteration it depends on min-cut bisector between two equal parts of hypergraph.

So we introduce some additions as evaluation and comparison of clusters density within merging process. Instead of evaluation the relative inter-connectivity and relative closeness we used formula (1)

$$CS = \frac{|c_{ij}|}{\min(|c_i|,|c_j|)} * \left( \frac{s_{ij}}{\frac{|c_i|}{|c_i|+|c_j|} s_i + \frac{|c_j|}{|c_i|+|c_j|} s_j} \right)^{\alpha} * \left( \frac{\min(s_i, s_j)}{\max(s_i, s_j)} \right)^{\beta}; (1)$$

where $|c_{ij}|$ – the number of edges that connect vertices between subclasses i and j; $|c_i|,|c_j|$ – the number of edges connecting vertices inside classes i and j respectively; $|s_{ij}|$ – the average length of the edges that connect vertices in subclasses i and j, $|s_i|,|s_j|$ – the average length of the edges inside subclasses i and j respectively;

$\alpha,\beta$ – define by user.

The first part of the formula it's a number of the edges that are merging two classes divided on the number of the edges in smaller class, this way allow to compute connectivity between two subgraphs with different densities. The second part of (1) is evaluation of similarity between two subgraphs. At the each step of the merging process we visit each subgraph and checks to see if any one of its adjacent subgraphs satisfy the (1) and then connect two subgraphs with the maximal value of (1).

Such approach allow to classify the clusters with different densities and nonlinear separated.

Experimental results

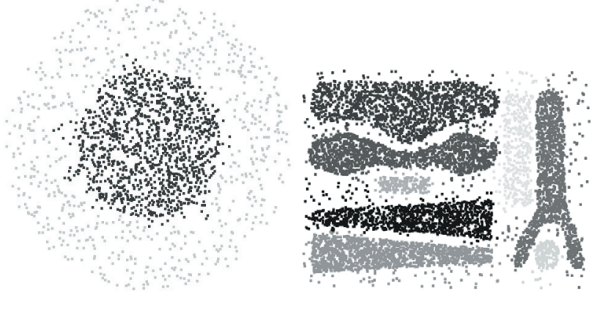
The overall computational complexity of CHAMELEON depends on the amount of time it requires to construct the K – nearest neighbors graph and the amount of time it requires to perform the two phases of the clustering algorithm. In [KHK, 99a] was shown that CHAMELEON is not very sensitive of values k for computing the k-nearest neighbor graph, of the va-

lue of MINSIZE for the phase I of the algorithm, and of scheme for combining relative inter-connectivity and relative closeness and associated parameters, and it was able to discover the correct clusters for all of these combinations of values for k and MINSIZE. In this section, we present experimental evaluation of clustering using hMETIS hypergraph partitioning package for k-way partitioning of hypergraph and for recursive bisection [KK, 98] and CLUTO 2.1.1– A Clustering Toolkit [Ka, 03].

We experimented with five different data sets containing points in two dimensions: "disk in disk", t4.8k, t5.8k, t8.8k, t7.10k [Ka lab.].The first data set, has a particularly challenging feature that two clusters are very close to each other and they have different densities and circles shapes. We choose the number of neighbors k=5, 15, 40, MINSIZE = 5%.

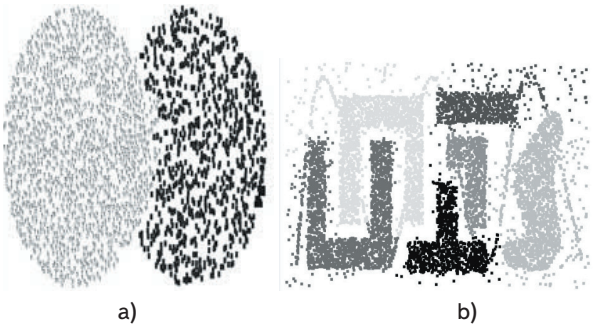
The data set t8.8k has eight clusters of different shapes, size and orientation, some of which are inside the space enclosed by other clusters. Moreover, it also contains random noise such as a collection of points forming vertical streaks. Looking at k=5 nearest neighbors we can see that hMETIS also compute k-way partitioning of hypergraph with mistakes closer to the border of two classes and CLUTO can not effectively merge clusters for such type of dataset using asymmetric k-NN, with k=5. It means that algorithm of the partitioning phase is very sensitive to the value of k for spherical shapes of clusters and to the types of k-NN graph (symmetric and asymmetric). It is very important to choose an optimal value of k, because with k=16 and more, and only for symmetric k-NN with weights of edges equal to the number of common neighbors we obtain final clustering with minimum percentages of errors.

Looking at the Fig. 3, Fig. 4 we can see the correct clustering results for the same data set "disk in disk" using our suggested expression. For another above mentioned data sets we obtain as well accuracy results.



a) b)

Fig 3. Clustering results using a new approach to the sub-clusters merging, k=5: a) Data set "disk in disk"; b) Data set "t8.8k.txt"



a) b)

Fig. 4. Clustering results using a new approach to the sub-clusters merging and k-means method, k=5: a) Data set "t111"; b) Data set "t4.8k", k =5

## 4. Conclusions

In this paper, we present our experiments with hierarchical clustering algorithm CHAMELEON for circles cluster shapes with different densities using hMETIS program that used multilevel k-way partitioning for hypergraphs and a Clustering Toolkit package that merges clusters based on a dynamic model. In CHAMELEON two clusters are merged only if the inter-connectivity and closeness between two clusters are comparable to the internal interconnectivity of the clusters and closeness of items within the clusters.

The methodology of dynamic modeling of clusters is applicable to all types of data as long as a similarity matrix can be constructed.

Experimental results showed that hMETIS compute k-way partitioning of hypergraph with mistakes closer to the border of two classes and CLUTO can not effectively merge clusters using asymmetric k-NN, with k=5.

We present a modified hierarchical clustering algorithm that measures the similarity of two clusters based on a new dynamic model with different shapes and densities. The merging process using the dynamic model presented in this paper facilitates discovery of natural and homogeneous not only circles cluster shapes.

Experimental results showed that this method is not sensitive to the value of k and doesn't need a specific k-nearest neighbor graph creating.

References


1. [Al, 97] Alpert C. J., Huang J. H. and Kahng A. B., Multilevel circuit partitioning. In: Proc. of the 34th ACM/IEEE Design Automation Conference. 1997.
2. [Fi, 82] Fiduccia C. M. and Mattheyses R. M., .A Linear-time Heuristic for Improving.
3. [GRS, 99] Guha S., Rastogi R., Shim K. ROCK: Robust Clustering using linKs, (ICDE'99).
4. [KAK, 97] Karypis G., Aggarwal R., V. Kumar. Multilevel hypergraph partitioning: Application in VLSI domain. In: Proceedings of the Design and Automation Conference. 1997.
5. [KKa, 98] Karypis G.and Kumar V.. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Computing, 1998.
6. [KKb, 98] Karypis G., and Kumar V., hMETIS 1.5.3: A hypergraph partitioning package. Technical report. Department of Computer Science, University of Minnesota, 1998.
7. [KHK, 99a] Karypis G.,. Han E.-H, and Kumar V.. CHAMELEON: A Hierarchical Clustering Algorithms Using Dynamic Modeling. IEEE Computer, 32(8):68–75, 1999.
8. [KHK, 99b] Karypis G., Han E.-H. and Kumar V.. Multilevel k-way hypergraph partitioning. In Proceedings of the Design and Automation Conference, 1999.
9. [Ka, 03] Karypis G., CLUTO 2.1.1. A Clustering Toolkit. Technical report. Department of Computer Science, University of Minnesota, 2003.
10. [Ka lab.] http://www.cs.umn.edu/karypis.
11. [Mi, 97] Mitchell T. M.. Machine Learning. McGraw Hill, 1997.