Improving the Orthogonal Range Search k-windows Algorithm*

P. Alevizos
Department of Mathematics,
University of Patras (UOP)
University of Patras Artificial
Intelligence Research Center (UPAIRC)
GR-26500 Patras, Greece
alevizos@math.upatras.gr

B. Boutsinas
Department of Business Administration,
UOP, UPAIRC
GR-26500 Patras, Greece
vutsinas@bma.upatras.gr

D. Tasoulis
Department of Mathematics, UOP, UPAIRC
GR-26500 Patras, Greece
dtas@math.upatras.gr

M.N. Vrahatis
Department of Mathematics, UOP, UPAIRC
GR-26500 Patras, Greece
vrahatis@math.upatras.gr

Abstract

Clustering, that is the partitioning of a set of patterns into disjoint and homogeneous meaningful groups (clusters), is a fundamental process in the practice of science. k-windows is an efficient clustering algorithm that reduces the number of patterns that need to be examined for similarity, using a windowing technique. It exploits well known spatial data structures, namely the range tree, that allows fast range searches. From a theoretical standpoint, the k-windows algorithm is characterized by lower time complexity compared to other well-known clustering algorithms. Moreover, it achieves high quality clustering results. However, it appears that it cannot be directly applicable in high-dimensional settings due to the superlinear space requirements for the range tree. In this paper, an improvement of the k-windows algorithm, aiming at resolving this deficiency, is presented. The improvement is based on an alternative solution to the orthogonal range search problem.

1. Introduction

Clustering, that is the partitioning of a set of patterns into disjoint and homogeneous meaningful groups (clusters), is a fundamental process in the practice of science. It is applied in various fields including data mining, statistical data analysis, pattern recognition, compression and vector quantization.

Usually, clustering algorithms produce as final output the means of discovered clusters. Providing that these means are the representatives of the clusters, the conjunction of attribute values describing each mean can be considered as a clustering rule for describing data (taking, of course, into consideration and a number of certain properties as density, variance, shape and separation [2]). Recently, the task of extracting knowledge from large databases, in the form of clustering rules, has been attracting increasing interest. Clustering rules can be extracted using unsupervised learning methods.

Algorithms for clustering data have been widely studied in various fields including Machine Learning, Neural Networks, Databases and Statistics. Clustering algorithms can be classified [2] as either hierarchical or iterative (partitional, density search, factor analytic or clumping and graph theoretic). Complete-link, average link and single-link algorithms [8] are the most popular hierarchical clustering algorithms. K-means [12] along with its variants and hill-climbing [4] are among the most popular partitional clustering algorithms.

k-means is a very popular and one of the best algorithms for implementing the clustering process. The time complexity of the algorithm is dominated by the product of the number of patterns, the number of clusters and the number of iterations. k-means often converges to a local minimum. In a former contribution [18] we had presented an improvement of k-means clustering algorithm, the k-windows algorithm, aiming at a better time complexity and partitioning accu-

*The authors acknowledge financial support of the Computer Technology Institute, Greece.

Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02)
1082-3409/02 $17.00 © 2002 IEEE

racy. That approach resulted in a reduction of the number of patterns that need to be examined for similarity, in each iteration, using a windowing technique. The latter is based on well known spatial data structures, namely the range tree, that allows fast range searches.

The k-windows algorithm is characterized by a lower time complexity compared to other well-known clustering algorithms, as shown in [18]. Moreover, it achieves high quality clustering results. However, it appears that the k-windows algorithm cannot be directly applicable in high-dimensional settings due to the superlinear space requirements for the range tree. To address this problem, we present an improvement of the k-windows algorithm, using a different solution to the orthogonal range search problem, namely the multidimensional binary tree method for orthogonal range search in $d \geqslant 2$ dimensions.

The rest of the paper is organized as follows. The k-windows algorithm is briefly described in Section 2, along with its computational complexity. The proposed improvement of the k-windows algorithm is described in Section 3. In Section 4, we present extensive empirical tests that illustrate that the new version outperforms the previous one. The paper ends with concluding remarks.

2 The k-windows Algorithm

The k-windows algorithm [18] is an improvement of the *k-means* algorithm, which in turn is a very popular algorithm particularly suited for implementing the clustering process because of its ability to efficiently partition very large numbers of patterns.

k-means consists of two main phases. During the first phase, a partition of patterns, in k clusters is calculated, while during the second one, the quality of the partition is determined. k-means is implemented by an iterative process that starts from a random initial partition. The latter is repeatedly revised until its quality function reaches an optimum.

In particular, the whole process is built upon four basic steps:

1) selection of the initial k means,

2) assignment of each pattern to the cluster with the nearest mean,

3) revision of k means, and

4) computation of the quality function.

The last three steps are performed iteratively until convergence.

The direct k-means algorithm is computationally very expensive for large sets of patterns. It requires time proportional to the product of the number of patterns, the number of clusters and the number of iterations. More specifically, the computationally most expensive step is that of assigning each pattern to the cluster with the nearest mean. This is imposed not only by its time complexity in relative terms, but, also, by its basic operation which is the calculation of the squared Euclidean distance.

The k-windows algorithm deals with this problem by using a windowing technique, that permits the consideration of only a limited number of patterns in each iteration. Moreover, the basic operation in the first loop, during the assignment of patterns to clusters, is now just the arithmetic comparison between two numbers.

The key idea behind the k-windows algorithm is to use a window in order to determine a cluster. The window is defined as an orthogonal range in the d-dimensional Euclidean space, where d is the number of numerical attributes. Therefore each window is a *d-range* and has a fixed size. Every pattern that lies within a window is considered as belonging to the corresponding cluster. Iteratively, each window is moved in the Euclidean space by centering itself on the mean of the patterns included. This takes place until any further movement does not result in an increase in the number of patterns that lie within it (see solid line squares in Fig. 1). After this step, we are able to determine the means of clusters as the means of the corresponding windows. However, since only a limited number of patterns is considered in each movement, the quality of a partition may not be optimum. Thus, the quality of a partition is calculated in a second phase. At first, windows are enlarged, in order to contain as many patterns from the corresponding cluster as possible (see dotted line squares in Fig. 1). The quality of a partition is determined by the number of patterns contained in any window, with respect to all the patterns.

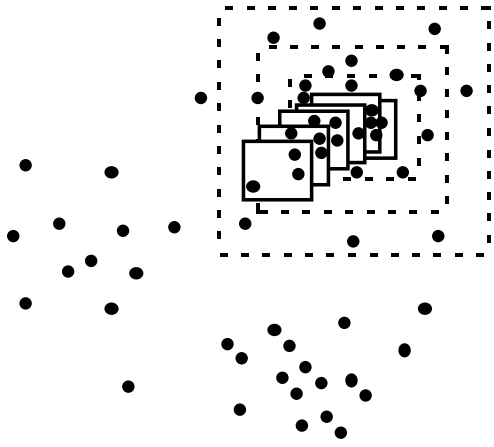

Figure 1. Movements and enlargements of a window.

The k-windows clustering algorithm is as follows:

Algorithm k-windows.

input k, a, v
initialize k means $i_{m1}, \ldots, i_{mk}$ along with their
k d-ranges $w_{m1}, \ldots, w_{mk}$ each of area a
repeat
    for each input pattern $i_l, 1 \leqslant l \leqslant n$
        do
            assign i_l to w_j,
            so that i_l lies within w_j
    for each d-range w_j
        do
            calculate its mean $i_{mj} = \frac{1}{|w_j|}\sum_{i_l \in w_j} i_l$
            and recalculate d-ranges
until no pattern has changed d-ranges
enlarge d-ranges up to no significant
change exists, in their initial mean
compute the ratio $r = \frac{1}{n}\sum_{j=1}^{k}|i_l \in w_j|$
if $r < v$
    do
        re-execute the algorithm

At first, k means are selected (possibly in a random manner). Initial d-ranges (windows) have as centers these initial means and each one is of area a. Then, the patterns that lie within each d-range are found, using the Orthogonal Range Search technique of Computational Geometry [3, 7, 11, 16]. The latter has been shown to be effective in many practical applications and a considerable amount of work has been devoted to this problem [16]. An orthogonal range search is based on a preprocess phase where a *range tree* is constructed. Patterns that lie within a d-range can be found traversing the range tree, in polylogarithmic time. The *orthogonal range search* problem can be stated as follows:

Input:
a) $V = \{p_1, \ldots, p_n\}$ is a set of n points in $\mathbb{R}^d$ the d-dimensional Euclidean space with coordinate axes $(Ox_1, \ldots, Ox_d)$,
b) a query d-range $\mathcal{Q}$= $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$ is specified by two points $(a_1, a_2, \ldots, a_d)$ and $(b_1, b_2, \ldots, b_d)$, with $a_j \leqslant b_j$.

Output:
report all points of V that lie within the d-range $\mathcal{Q}$.

Then, the mean of patterns, that lie within each range, is calculated. Each such mean defines a new d-range, that is considered a movement of the previous d-range. The last two steps are executed repeatedly, until there is no d-range that includes a significant increment of patterns after a movement.

In a second phase, the quality of the partition is calculated. At first, the d-ranges are enlarged in order to include as many patterns as possible from the cluster. This can be achieved by forcing d-ranges to preserve their mean during enlargement. Then, the relative frequency of patterns assigned to a d-range in the whole set of patterns, is calculated. If the relative frequency is small, then, possibly, there may be a missing cluster (or clusters). In that case, the whole process is repeated.

The time complexity of the k-means is $O(ndkt)$ while in the case of k-windows it is reduced to $O(dkqr(\frac{\log^{d-2} n}{d} + s))$, where qr is empirically shown to be proportional to t [18]. This reduction is heavily dependent on the orthogonal range search. Thus, the k-windows algorithm has a lower time complexity than other well-known clustering algorithms such as BIRCH [19], CHAMELEON [13], CLARANS [15], CURE [10] and DBSCAN [9].

3 The proposed improvement.

It seems that the k-windows algorithm cannot be directly applicable in practical settings due to the superlinear space requirements for the range tree. The latter is obvious in multidimensional cases. However, one could follow several approaches for scaling up to very large data sets, as sampling (e.g. [6]), or parallelizing (e.g. [14]), or distributing (e.g. [5]).

The proposed improvement of the k-windows algorithm is based on using the multidimensional binary tree method for orthogonal range search for $d \geqslant 2$ dimensions.

Let us consider a set $V = \{p_1, p_2, \ldots, p_n\}$ of n points in d-dimensional space $\mathcal{R}^d$ with coordinate axes $(Ox_1, Ox_2, \cdots, Ox_d)$. Let $p_i = (x_1^i, x_2^i, \cdots, x_d^i)$ be the representation of any point p_i of V.

Definition: Let V_s be a subset of the set V. The *middle point* p_h of V_s with respect to the coordinate x_i ($1 \leqslant i \leqslant d$) is defined as the point which divides the set V_s-$\{p_h\}$ in two subsets V_{s_1} and V_{s_2}, such that:
i) $\forall p_g \in V_{s_1}$ and $\forall p_r \in V_{s_2}$, $x_i^g \leqslant x_i^h \leqslant x_i^r$.
ii) V_{s_1} and V_{s_2} have approximately equal numbers of elements: If $|V_s| = t$ then $|V_{s_1}| = \lceil \frac{t-1}{2} \rceil$ and $|V_{s_2}| = \lfloor \frac{t-1}{2} \rfloor$.

The **multidimensional binary tree** T which stores the points of the set V is constructed as follows (see Fig. 2).

1) Let p_r be the *middle point* of the given set V, with respect to the first coordinate x_1. Let V_1 and V_2 be the corresponding partition of the set V-$\{p_r\}$. The point p_r is stored in the root of T.
2) Each node p_i of T, obtains a left child $left[p_i]$ and a right child $right[p_i]$ as follows: MBT(p_r,V_1,V_2,1)

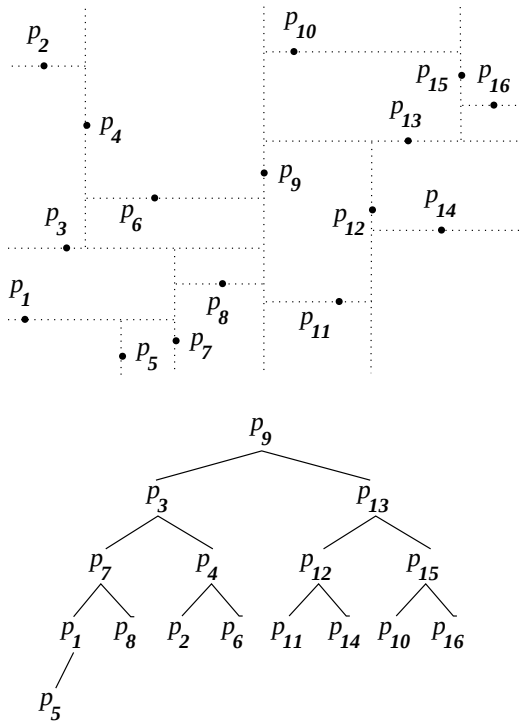

Figure 2. A set $V = \{p_1, p_2, \ldots, p_{16}\}$ of points in 2-dimensional space $\mathcal{R}^2$ and the corresponding 2-dimensional binary tree.

procedure MBT(p,L,M,k)
begin
$k \longleftarrow k+1$
if $k = d+1$ **then** $k \longleftarrow 1$
if $L \neq \emptyset$ **then**
begin
let u be the middle point of the set L with respect to the coordinate x_k. The point u divides the set L-$\{u\}$ in two subsets L_1 and L_2.
$left[p] \longleftarrow u$
MBT(u,L_1,L_2,k)
end
if $M \neq \emptyset$ **then**
begin
let w be the middle point of the set M with respect to the coordinate x_k and let M_1 and M_2 be the corresponding partition of the set M-$\{w\}$.
$right[p] \longleftarrow w$
MBT(w,M_1,M_2,k)
end
end

Let us consider a query d-range $\mathcal{Q} = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$ specified by two points $(a_1, a_2, \ldots, a_d)$ and $(b_1, b_2, \ldots, b_d)$, with $a_j \leqslant b_j$. The search of the tree T is effected by the following algorithm which accumulates the retrieved points in a list $\mathcal{A}$, initialized as empty (see Fig. 3):

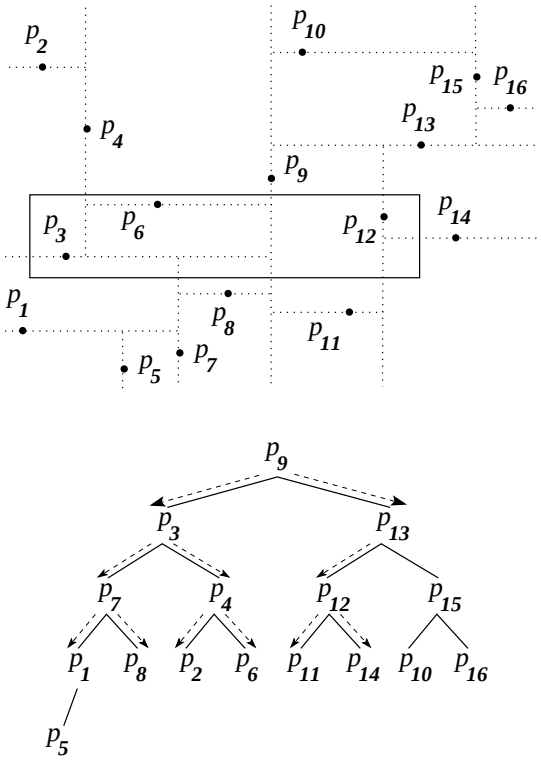

Figure 3. Illustration of a range search in the set V of the previous Figure. In the 2-dimensional binary tree we show the nodes actually visited by the search.

The orthogonal range search algorithm
1) $\mathcal{A} \longleftarrow \emptyset$
2) Let p_r be the root of T: SEARCH(p_r,$\mathcal{Q}$,$\mathcal{A}$,1)
3) return $\mathcal{A}$

procedure SEARCH(p_t,$\mathcal{Q}$,$\mathcal{A}$,i)
begin
if $i = d+1$ then $i \longleftarrow 1$
let $p_t = (x_1^t, x_2^t, \cdots, x_d^t)$
if $a_i \leqslant x_i^t \leqslant b_i$ **then if** $p_t \in \mathcal{Q}$ **then** $\mathcal{A} \longleftarrow \mathcal{A} \cup \{p_t\}$
if $p_t \neq leaf$ **then**
begin
if $a_i < x_i^t$ **then** SEARCH($left[p_t]$,$\mathcal{Q}$,$\mathcal{A}$,$i+1$)
if $x_i^t < b_i$ **then** SEARCH($right[p_t]$,$\mathcal{Q}$,$\mathcal{A}$,$i+1$)
end
end

From the performance viewpoint, the multidimensional binary tree uses $\theta(dn)$ optimal storage and it can also be con-

structed in optimal time $\theta(dn \log n)$. The worst-case behavior of the query time is $O(|\mathcal{A}| + dn^{1-1/d})$ (see [16]).

From a theoretical standpoint, k-windows has a lower time complexity, with respect to other clustering algorithms. Moreover, it achieves high quality clustering results. Since, we could not compare our method to the great number of k-means variations due to space limitations, we chose to simply resort to visual inspection. We applied the k-windows algorithm to various two-dimensional synthetic sample databases (see Fig. 4). Note that these sample databases have already been used as test data sets to evaluate BIRCH, CHAMELEON, CLARANS, CURE and DBSCAN (e.g. in [10, 13, 17]).

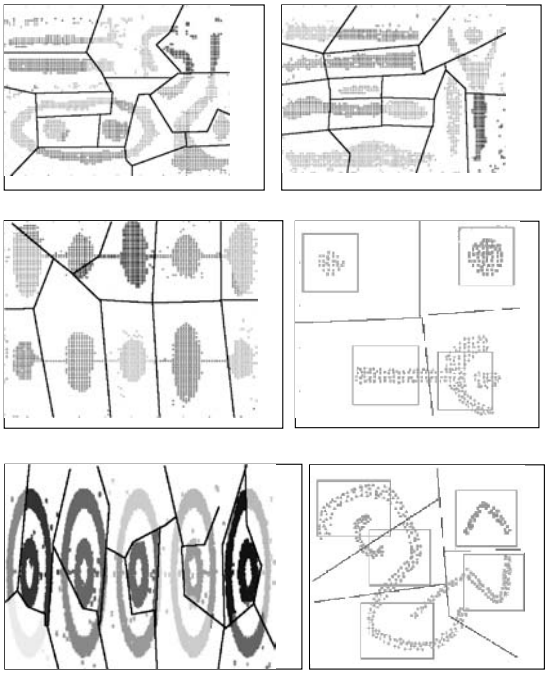

Figure 4. Clusters discovered by k-windows.

4 Empirical Tests

We have implemented the k-windows algorithm in the C++ Language under the Linux operating system with the "g++ ver 2.96" compiler. We have, also, implemented the proposed improved version, in the same developing environment, in order to evaluate its relative performance. For comparative purposes, we have also implemented a brute force version of the k-windows algorithm. In this version, the computationally most expensive step is that of assigning each pattern to the cluster with the nearest mean, which is performed by a brute force search of all patterns. Using those implementations, we have applied the above three versions of k-windows algorithm in six two-dimensional synthetic sample databases. The test sample databases (DSet1-6) are depicted in Fig. 5 and they include clusters with both normal and irregular shape.

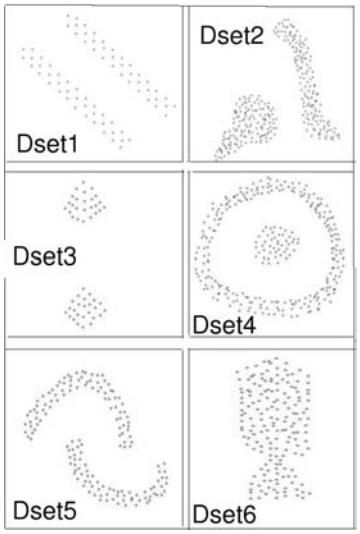

Figure 5. The six synthetic sample databases.

Moreover, we have applied the above three versions of k-windows algorithm in multidimensional MagnetoEncephaloGram (MEG) signals which are generated from the ionic micro-currents of the brain and originated at the cellular level [1]. The MEG analysis can provide information of vital importance for the monitoring of brain dynamics in both normal and pathological conditions of the Central Nervous System. The MEG signals are recorded with the use of specific Superconductive Quantum Interference Devices (SQUIDs). SQUIDs are very sensitive superconductive magneto-meters with the ability to detect and measure very weak magnetic fields, of the order of fT (= 10-15T) and they can be used ideally for the recording of the MEG, since they do not emit any radiation and they are totally non invasive. To derive the multi-dimensional phase space from the one-dimensional MEG signals a well-known technique from the field of non-linear dynamics and chaos was used. This technique is the embedding method for the reconstruction of multi-dimensional dynamics of a system from an one-dimensional observable. According to the embedding theorem the reconstructed multi-dimensional phase space from an one-dimensional observable of a system is topologically equivalent to the original phase space of the system under consideration.

We have applied the above three versions in four different MEG multidimensional signals, namely the MEG1 an

epileptic magnetoencephalogram signal of 500 points in 5-dimensional space, the MEG2 a fetal magnetocardiogram signal of 1500 points in 3-dimensional space, the MEG3 another fetal magnetocardiogram signal of 1500 points in 3-dimensional space and MEG4 a fetal magnetocardiogram signal of 800 points in 5-dimensional space.

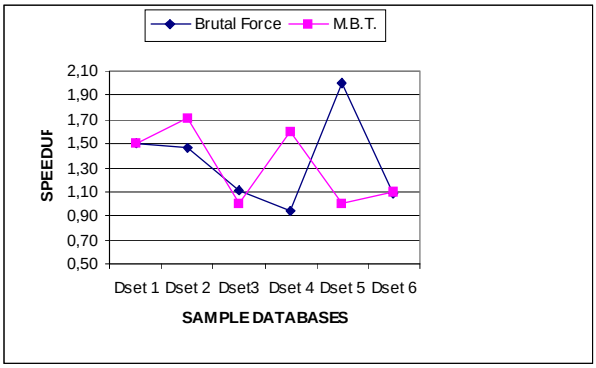

Figure 6. Speedup for 2-dimensional space.

In this paper, empirical tests aim at examining the performance of the three versions of the k-windows algorithm. Notice that, as far as the partitioning accuracy is concerned, all the three versions are identical. Thus, empirical results for the partitioning accuracy with respect to k-means algorithm can be found in [18]. In Fig. 4 clusters discovered by k-windows are shown in the synthetic sample databases.

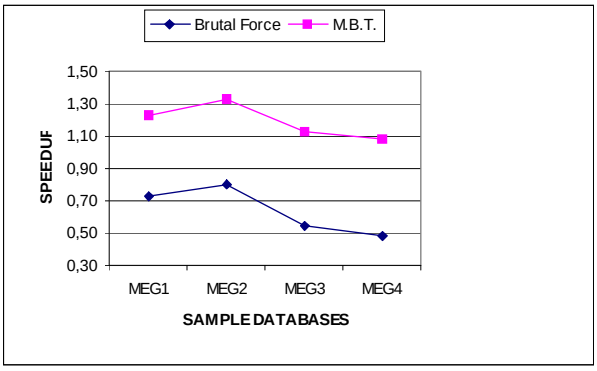

Figure 7. Speedup for multidimensional space.

In Fig. 6 the speedup is depicted for the multidimensional binary tree (M.B.T.) and brute force versions with respect to the first version of the k-windows algorithm applied in the test sample databases. It seems that both M.B.T. and the brute force versions outperform the first version for two dimensions. In Fig. 7 the speedup is depicted for the multidimensional binary tree (M.B.T.) and brute force versions with respect to the first version of the k-windows algorithm applied in the MEG signals. It is obvious that the M.B.T. version outperforms the first version for two dimensions. Moreover, using the M.B.T. version there is a significant speedup in building the range tree during a preprocessing phase, as it is depicted in Fig. 8.

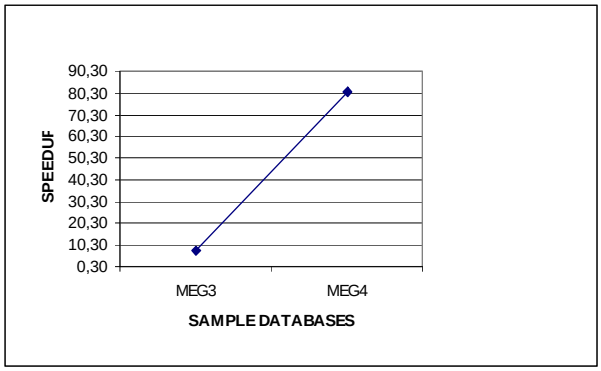

Figure 8. Speedup for building the range tree.

5 Concluding Remarks

The k-windows algorithm [18] constitutes an improvement of the well known and widely used *k-means* algorithm. However, it seems that the version of k-windows algorithm, that uses the orthogonal range search technique of Computational Geometry, cannot be directly applicable in settings due to the superlinear space requirements for the range tree. The latter is obvious in multidimensional practical cases. Thus, we present an improvement of the k-windows algorithm by using a different solution to the orthogonal range search problem. This improvement preserves the partitioning accuracy while at the same time it is characterized by a lower overall time complexity, despite the fact that the time complexity of the range search is higher. The improvement is more pronounced in multidimensional spaces. Finally, the proposed version exhibits a significant improvement in the time complexity of constructing the range tree.

References

[1] A.V. Adamopoulos, B. Boutsinas, M.N. Vrahatis and P. Anninos. *Analysis of Normal and Pathological Fetal Magnetocardiograms Using Clustering Algorithms*, Proceedings of European Symposium on Intelligent

Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (eunite 2001), Special Session - Adaptive Systems and Hybrid CI in Medicine, Tenerife, Spain, December 2001.

[2] M. Aldenderfer and R. Blashf ield. *Cluster Analysis*, in Series: Quantitative Applications in the Social Sciences, Sage Publications Inc., 1984.

[3] P. Alevizos. *An Algorithm for Orthogonal Range Search in* $d \geqslant 3$ *dimensions*, Proceedings of 14th European Workshop on Computational Geometry, Barcelona, 1998.

[4] M. Anderberg. *Cluster Analysis for Applications*, Academic Press, NY, 1973.

[5] B. Boutsinas and T. Gnardellis. *On Distributing the Clustering Process*, Pattern Recognition Letters, Elsevier Science Publishers B.V., 23(8), pp.999–1008, 2002.

[6] P.S. Bradley, U.M. Fayyad and C. Reina. *Scaling Clustering Algorithms to Large Databases*, Proceedings of the 4th Int. Conf. on Knowledge Discovery and Data Mining, pp. 9–15, 1998.

[7] B. Chazelle. *Filtering Search: A new approach to query-answering*, SIAM J. Comput., 15, 3, pp.703–724, 1986.

[8] R.C. Dubes and A.K. Jain. *Clustering methodologies in exploratory data analysis*, Adv. Comput., 19, pp.113–228, 1980.

[9] M. Ester, H.-P. Kriegel, J. Sander and X. Xu. *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the 2nd Int. Conf. on Knowledge Discovery and Data Mining, pp. 226–231, 1996.

[10] S. Guha, R. Rastogi and K. Shim. *CURE: An efficient algorithm for clustering large databases*, Proceedings of ACM-SIGMOD 1998 International Conference on Management of Data, Seattle, pp.73–84, 1998.

[11] B. Chazelle and L.J. Guibas. *Fractional Cascading: II. Applications*, Algorithmica, 1, pp.163–191, 1986.

[12] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*, Prentice-Hall, NJ, 1988.

[13] G. Karyapis, E.H. Han and V. Kumar. *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*, In IEEE Computer, Special Issue on Data Analysis and Mining.

[14] X. Li and Z. Fang. *Parallel clustering algorithms*, Parallel Computing, 11, pp.275–290, 1989.

[15] R.T. Ng and J. Han. *Efficient and Effective Clustering Methods for Spatial Data Mining*, Proc. of 1994 Int'l Conf. on Very Large Data Bases (VLDB'94), pp.144–155, 1994.

[16] F. Preparata and M. Shamos. *Computational Geometry*, Springer Verlag, 1985.

[17] J. Sander, M. Ester, H.-P. Kriegel and X. Xu. *Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications*, Data Mining and Knowledge Discovery, 2(2), pp.169–194, 1998.

[18] M.N. Vrahatis, B. Boutsinas, P. Alevizos and G. Pavlides. *The New* k*-windows Algorithm for Improving the* k*-means Clustering Algorithm*, Journal of Complexity, vol. 18, pp.375–391, 2002.

[19] T. Zhang, R. Ramakrishnan and M. Livny. *BIRCH: An Eff icient Data Clustering Method for Very Large Databases*, Proc. of ACM SIGMOD International Conference on Management of Data, pp.103–114, 1996.