Unsupervised Distributed Clustering

D. K. Tasoulis, M. N. Vrahatis,
Department of Mathematics,
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece.
{dtas,vrahatis}@math.upatras.gr

ABSTRACT
Clustering can be defined as the process of partitioning a set of patterns into disjoint and homogeneous meaningful groups, called clusters. The growing need for distributed clustering algorithms is attributed to the huge size of databases that is common nowadays. In this paper we propose a modification of a recently proposed algorithm, namely k-windows, that is able to achieve high quality results in distributed computing environments.

KEY WORDS
Distributed Knowledge-based Systems, Data Mining, Clustering, High Performance Computing

1 Introduction

Clustering, that is "grouping a collection of objects into subsets or clusters, such that those within one cluster are more closely related to one another than objects assigned to different clusters" [8], is a fundamental process of Data Mining. In particular, clustering is fundamental in knowledge acquisition. It is applied in various fields including data mining [5], statistical data analysis [1], compression and vector quantization [11]. Clustering is, also, extensively applied in social sciences.

The task of extracting knowledge from large databases, in the form of clustering rules, has attracted considerable attention. The ability of various organizations to collect, store and retrieve huge amounts of data has rendered the development of algorithms that can extract knowledge in the form of clustering rules, a necessity. Distributed clustering algorithms embrace this trend of merging computations with communication and explore all the facets of the distributed computing environments. Thus a distributed algorithm must take under consideration that the data may be inherently distributed to different loosely coupled sites connected through a network.

While there are many approaches to parallel and distributed Data Mining [3, 9, 10], the field of parallel and distributed clustering is not extensively developed. In [14] a parallel version of DBSCAN [12] and in [4] a parallel version of k-means [7] were introduced. Both algorithms start with the complete data set residing in one central server and then distribute the data among the different clients. For instance, in the case of parallel DBSCAN, the data are organized at the server site within an R*-tree [2]. This preprocessed data is then distributed among the clients which communicate with each other via message passing.

In this paper we present a modification of a recently proposed algorithm [13], namely k-windows, that best fits distributed computing environments. The k-windows algorithm has the ability to endogenously determine the number of clusters. This is a fundamental issue in cluster analysis, independent of the particular technique applied. The proposed distributed version of the algorithm manages to distinguish the number of clusters present in a dataset with satisfactory accuracy.

The paper is organized as follows; Section 2 briefly describes the workings of the k-windows algorithm; Section 3 discusses the distributed implementation of the algorithm; while Section 4, reports the results of the experiments conducted. The paper closes with concluding remarks and a short discussion about future research directions.

2 The Unsupervised k-windows Algorithm

For completeness purposes we briefly describe the unsupervised k-windows algorithm. Intuitively, the k-windows algorithm proposed in [13] tries to place a d-dimensional window (frame, box) containing all patterns that belong to a single cluster; for all clusters present in the dataset. At a first stage, the windows are moved in the Euclidean space without altering their size. Each window is moved by setting its center to the mean of the patterns currently included (sea solid line squares in Fig. 1). This process continues iteratively until further movement does not increase the number of patterns included. At the second stage, the size of each window is enlarged in order to capture as many patterns of the cluster as possible. The process of enlargement terminates when the number of patterns included in the window no longer increases. The two processes are exhibited in Fig 1, where the initial 2-dimensional window $M1$, is successively moved, and then subjected to two phases of enlargement that result to the final window $E2$. The unsupervised k-windows algorithm generalizes the original algorithm [13], by considering a number of initial windows greater than the expected number of clusters. After the clustering procedure is terminated, the windows that are suspected to capture patterns that belong to a single cluster

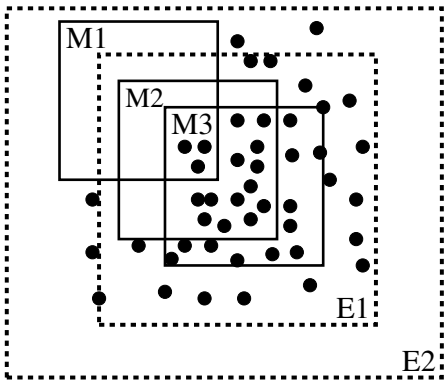

Figure 1. Sequential Movements (solid lines) and subsequent, enlargements (dashed lines) of the initial window $M1$ that result to the final window $E2$.

are merged. The merging procedure is illustrated in Fig. 2. Windows $W1$ and $W2$ share a sufficiently large number of patterns between them, and thus, are merged to form a single cluster. On the other hand $W5$ and $W6$ are not merged, because although they overlap, the patterns in their intersection are insufficient to consider them as part of one cluster. In more detail; at first, k points are selected (possibly

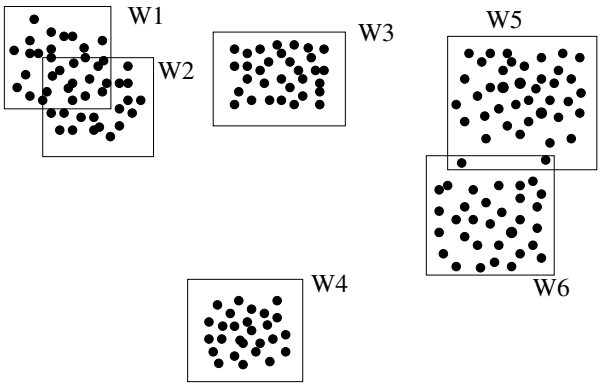

Figure 2. The merging procedure. w_1 and w_2 have many points in common thus they are considered to belong to the same cluster.

in a random manner). The k initial d–ranges (windows), of size a, have as centers these points. Subsequently, the patterns that lie within each d-range are identified. Next, the mean of the patterns that lie within each d–range (i.e. the mean value of the d–dimensional points) is calculated. The new position of the d–range is such that its center coincides with the previously computed mean value. The last two steps are repeatedly executed as long as the increase in the number of patterns included in the d–range that results from this motion satisfies a stopping criterion. The stopping criterion is determined by a variability threshold θ_v that corresponds to the least change in the center of a d–range that is acceptable to re-center the d–range.

Once movement is terminated, the d–ranges are enlarged in order to capture as many patterns as possible from the cluster. Enlargement takes place at each coordinate separately. The d–ranges are enlarged by θ_e/l percent at each coordinate, where θ_e is user defined, and l stands for the number of previous successful enlargements. After the enlargement in one coordinate is performed, the window is moved, as described above. Once movement terminates, the proportional increase in the number of patterns included in the window is calculated. If this proportion does not exceed the user–defined coverage threshold, θ_c, the enlargement and movement steps are rejected and the position and size of the d–range are reverted to their prior to enlargement values. If, on the other hand, the proportional increase in the patterns included in the window exceeds θ_c, then the new size and position are accepted. In the case that enlargement is accepted for coordinate $d' \geqslant 2$, then for all coordinates d'', such that $d'' < d'$, the enlargement process is performed again assuming as initial position the current position of the window. This process terminates if enlargement in any coordinate does not result in a proportional increase in the number of patterns included in the window beyond the threshold θ_c. An example of this process is provided in Fig. 3. In the figure the window is initially enlarged horizontally ($E1$). This enlargement is rejected since it does not produce an increase in the number of patterns included. Next the window is enlarged vertically, this enlargement is accepted, and the result of the subsequent movements and enlargements is the initial window to become $E2$. The key

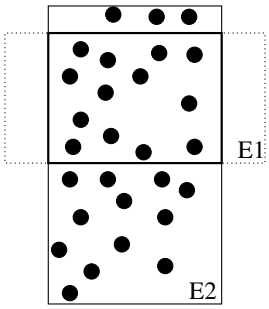

Figure 3. The enlargement process. Enlargement in the horizontal coordinate (E1) is rejected, while in the vertical coordinate it is accepted. After subsequent movements and enlargements the window becomes E2.

idea to automatically determine the number of clusters, is to apply the k-windows algorithm using a sufficiently large number of initial windows. The windowing technique of the k-windows algorithm allows for a large number of initial windows to be examined, without any significant overhead in time complexity. Once all the processes of movement and enlargement for all windows terminate, all overlapping windows are considered for merging. The merge

operation is guided by a merge threshold θ_m. Having identified two overlapping windows, the number of patterns that lie in their intersection is calculated. Next the proportion of this number to the total patterns included in each window is calculated. If the mean of these two proportions exceeds θ_m, then the windows are considered to belong to a single cluster and are merged, otherwise not. This operation is illustrated in Fig. 2; the extent of overlapping between windows $W1$ and $W2$ exceeds the threshold criterion and the algorithm considers both to belong to a single cluster, unlike windows $W5$ and $W6$, which capture two different clusters.

The remaining windows, after the quality of the partition criterion is met, define the final set of clusters. If the quality of a partition, determined by the number of patterns contained in any window, with respect to all patterns is not the desired one, the algorithm is re-executed. The user defined parameter u serves this purpose. Thus the algorithm takes as input seven user defined parameters

1. a: the initial window size,
2. u: the quality parameter,
3. θ_e: the enlargement threshold,
4. θ_m: the merging threshold,
5. θ_c: the coverage threshold,
6. θ_v: the variability threshold,
7. k: the number of initial windows.

The output of the algorithm is a number of sets that define the final clusters discovered in the original dataset. In brief, the algorithm works as follows:

1. input$\{a, u, \theta_e, \theta_m, \theta_c, \theta_v\}$
2. Determine the number, k, and
the centers of the initial d–ranges.
3. Perform sequential movements and enlargements
of the d–ranges.
4. Perform the merging of the resulting d–ranges.
5. Report the groups of d–ranges
that comprise the final clusters.

3 Distributing the Clustering Process

This section describes the distributed implemetnation of the unsupervised k-windows algorithm.

In a distributed computing environment the dataset is spread over a number of different sites. Thus, let us assume that the entire dataset X is distributed among m sites each one storing X_i for $i = 1, \ldots, m$, so $X = \bigcup_{i=1,\ldots,m} X_i$. Furthermore let us assume that there is a central site O that will hold the final clustering results.

The proposed implementation takes into consideration the fact that the data is distributed across m sites, and distributes the whole clustering procedure locally. In more detail, at each site i, the k-windows algorithm is executed over the X_i dataset. This step results in a set of d–ranges (windows) W_i for each site. To obtain the final clustering result over the whole dataset X, all the final windows from each site are collected to the central node O. The central node is responsible for the final merging of the windows and the construction of the final results. As it has already been mentioned in Section 2, all overlapping windows are considered for merging. The merge operation is based on the number of patterns that lie in the window intersection. In the distributed environment the determination of the the number of patterns at each intersection between two windows may be impossible. (For example each site might not want to disclose this kind of information about its data. Alternatively, the exchange of data might be over a very slow network that restrains the continuous exchange of information.) Under this constraint, the proposed implementation always considers two overlapping windows to belong to the same cluster, irrespective of the number of overlapping points. The θ_m parameter becomes irrelevant. A high level description of the proposed algorithmic scheme follows:

1. for each site i, with $i = 1, \ldots, m$
execute the k-windows algorithm over X_i
send W_i to the central node O.
2. At the central node O
for each site i
get the resulting set of d–ranges W_i
set $W \leftarrow W \bigcup W_i$
{**comment**: d–range merging}
for each d–range w_j not marked
do
mark w_j with label w_j
if $\exists w_i \neq w_j$, that overlaps with w_j
then mark w_i with label w_j

4 Numerical Experiments

Numerical experiments were performed having developed a C++ Interface, for the proposed algorithm, built under the Red Hat Linux 8.0 operating system using the GNU compiler collection (gcc) version 3.2. To test the efficiency of the algorithm, we resolve to experiments that produce results that can be readily visualised. Thus a two dimensional dataset $dataset_1$ consisting of 10241 points was constructed. This dataset contains 5 clusters of different sizes, and a number of outlier points some of which connect two clusters. The dataset along with the clustering result, of the unsupervised k-windows algorithm when the whole dataset resides in one single site, is exhibited in Fig. 4. The number of the initial windows for this run was set to 256.

At a next step the dataset was randomly permutated and was distributed over 4, 8 and 16 sites. In Figs. 5, 6 and 7, the results of the algorithm for 4, 8 and 16 sites respectively are exhibited. As it is obvious from the figures the results are correct in all three cases. It should be noted that for the cases of 8 and 16 sites a different extra cluster is identified by the algorithm, but it is not considered important since in both cases it holds a small amount of points and does not affect the correct identification of the 5

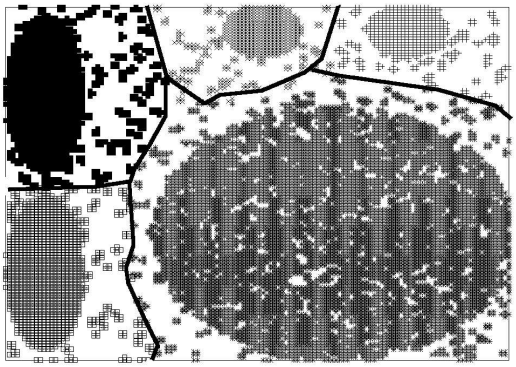

Figure 4. $dataset_1$ with the clustering result of the standard k-windows algorithm

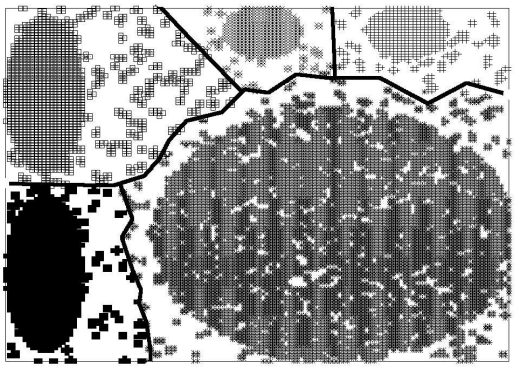

Figure 5. $dataset_1$ with the clustering result for 4 sites and 64 initial windows per site

main clusters. In the case of 8 sites the cluster resides in the botom left as it is exhibeted in Fig 6. On the other hand in the case of 16 sites the extra cluster resides in the top right corner (Fig 7).

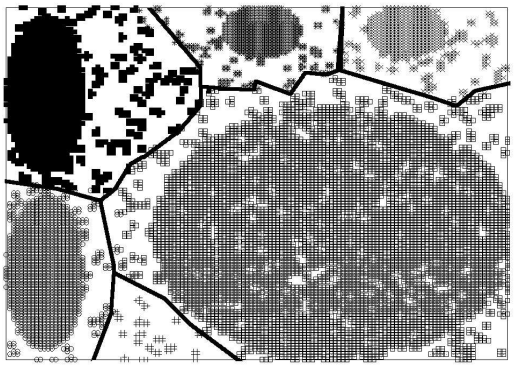

Figure 6. $dataset_1$ with the clustering result for 8 sites and 32 initial windows per site

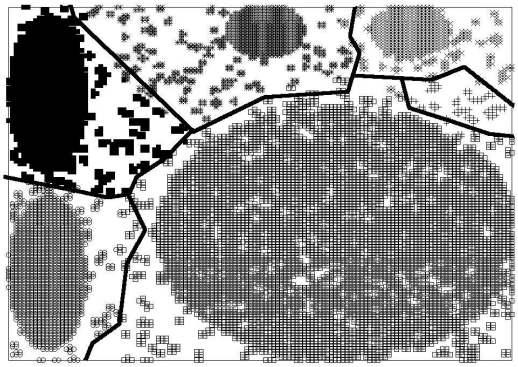

Figure 7. $dataset_1$ with the clustering result for 16 sites and 16 initial windows per site

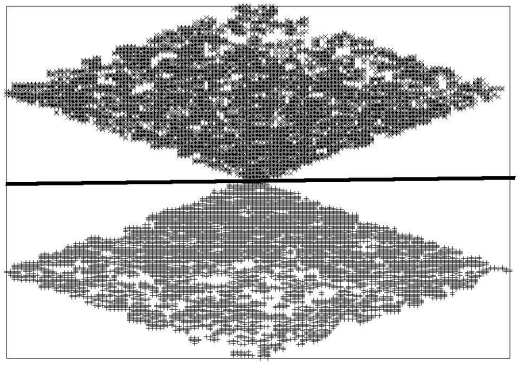

Figure 8. $dataset_2$ with the clustering result for 1, 4, 8 and 16 sites

In Fig 8 another dataset, $dataset_2$ is exhibited. This dataset contains 13488 points organized in two clusters close to one another whose density is unequally distributed. The result of the application of the algorithm, also exhibeted in Fig 8, remains unaffected by using 1, 4, 8 and 16 sites. The number of initial windows used was 1024, 256, 128, and 64 for the 1, 4, 8 and 16 sites respectively. As it is obvious the algorithm remains efficient even if clusters are very close to each other as in the case of $dataset_2$.

Another interesting feature of the distributed clustering approach is the speedup gained to the running time of the algorithm. To test the speedup obtained for the proposed version, the PVM [6] parallel computing library was employed. The developed software distributed the dataset over computer nodes (PIII 1200MHz 256MB RAM), that were connected through a 100Mbit Fast Ethernet network, and then executed the algorithm in parallel. When all the nodes returned the results back the central node (PIII 1400 MHz) performed the window merging. In Fig. 9 the speedup obtained is illustrated for the $dataset_2$, using 1, 4, 8 and 16 nodes. As it is obvious the algorithm achieves an almost 5 times faster running time.

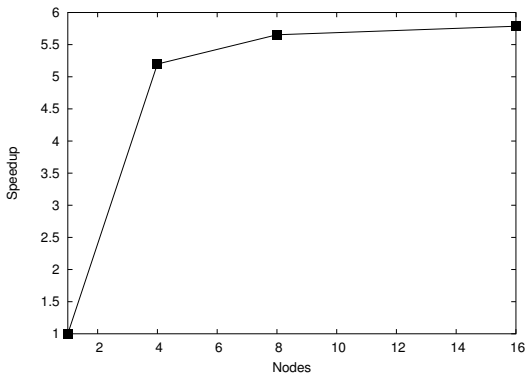

Figure 9. Speedup obtained for the $dataset_2$

5 Conclusions

Clustering is a fundamental process in the practice of science. Due to the increasing size of current databases, constructing efficient distributed clustering algorithms has attracted considerable attention. The present study presented the distributed version of a recently proposed algorithm, namely k-windows. The specific algorithm is characterized by the highly desirable property that the number of clusters is not user defined, but rather endogenously determined during the clustering process.

In this paper we restricted our attention to the case where no actual data exchange is allowed among the sites. The distributed version proposed is able to achieve high quality results, in the datasets examined. It is also worth noting that for the two datasets considered, the algorithm was able to identify the number of clusters correctly. In a future correspondence we intend to investigate the perfomance of the distributed algorithm on more datasets.

Acknowledgments

The authors gratefully acknowledge the contribution of Mr. S. Stavropoulos and Mr. J. Maramatakis of the Computer Laboratory of the Department of Mathematics in the University of Patras, for the provision of the computational resources required.

References

[1] M.S. Aldenderfer and R.K. Blashfield. *Cluster Analysis*. Quantitative Applications in the Social Sciences. SAGE Publications, London, 1984.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proc. of ACM SIGMOD International Conference on Management of Data*, 1990.

[3] P.K. Chan and S.J. Stolfo. Sharing learned models among remote database partitions by local meta-learning. In *Knowledge Discovery and Data Mining*, pages 2–7, 1996.

[4] I.S. Dhillon and D.S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*, pages 245–260, 2000.

[5] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[6] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, 1994.

[7] J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[9] H. Kargupta, W. Huang, K. Sivakumar, and E.L. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[10] W. Lam and A.M. Segre. Distributed data mining of probabilistic knowledge. In *Proceedings of the 17th International Conference on Distributed Computing Systems, Washington*, pages 178–185. IEEE Computer Society Press, 1997.

[11] V. Ramasubramanian and K. Paliwal. Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding. *IEEE Transactions on Signal Processing*, 40(3):518–531, 1992.

[12] 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):169–194, 1998.

[13] 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*, 18:375–391, 2002.

[14] X. Xu and J. Jgerand H.P. Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Mining and Knowledge Discovery*, 3:263–290, 1999.