VDBSCAN and α-Bisecting Spherical K-Means in distributed information retrieval systems

Daniel Jiménez González[1], Vicente Vidal Gimeno[1] and Leroy Anthony Drummond[2]

[1] Department of Computer Systems and Computation
Polytechnic University of Valencia (Spain)
dajigon@inf.upv.es ; vvidal@dsic.upv.es

[2] Computational Research Division
Lawrence Berkeley National Laboratory
Berkeley, California, USA
ladrummond@lbl.gov

Abstract. We present a comparative study between VDBSCAN and α-Bisecting Spherical K-Means algorithms. The first algorithm, VDBSCAN, is a variation of the DBSCAN algorithm, which produces the same results as DBSCAN but add the possibility of selecting a parameter for the DBSCAN. The second algorithm, α-Bisecting Spherical K-Means, is a variation of the well known K-Means algorithm that improves the K-Means performance. Here, we have implemented parallel versions of the VDBSCAN and α-Bisecting Spherical K-Means algorithms and compare the performance of these implementations.

Keywords: DBSCAN, K-Means, Density-Based clustering, Parallel computing, Dataset clustering, Hierarchical divisive clustering

1 Introduction

This paper shows a comparative study between parallel implementations of VDBSCAN and α-Bisecting Spherical K-Means algorithms. Two of the most important clustering algorithms are the DBSCAN, a density-based clustering method, and the K-Means, a hierarchical divisive clustering method.

VDBSCAN [1] is a variation of DBSCAN. It modifies the algorithm used to manage noise, while it maintains the same final results. These variations are important because they allow selecting an ε parameter with a heuristic algorithm [1]. The α-Bisecting Spherical K-Means [2, 3] obtains better performance than basic K-Means.

In this article we study the improvements of the parallel VDBSCAN and parallel α-Bisecting Spherical K-Means. We assume that the problem to be solve has already been distributed because generally this is the case in information retrieval systems in which the document subcollections are already distributed.

1.1 Clustering models

Clustering models are based on a partition of a document collection into different clusters, where the membership to a cluster is decided by the similarity between documents [4, 5, 6]. The base assumption is that similar documents tend to be more related than those that are very different.

One of the main problems of the clustering techniques is how to determine the number of clusters in a dataset due to particular difficulties that algorithms normally ignore [6]. However DBSCAN does not present this problem since it can determine the number of clusters automatically, and the α-Bisecting Spherical K-Means has the number of clusters as a parameter.

Literature offers a large collection of clustering algorithms. We should not forget that there is no universal technique for clustering that can be applied to all datasets [7]. All clustering techniques can not find all clusters, and even if a technique finds them, the algorithm's complexity varies. Clustering methods can be classified in several types [4, 5, 7, 8]: partitional, hierarchical, density-based clustering, grid-based clustering, model-based clustering and categorical data clustering.

It is interesting to observe the increasing number of DBSCAN based algorithms in many different fields of application [9, 10, 11, 12, 13, 14, 15].

1.2 The K-Means

The K-Means algorithm [7, 16, 17, 18, 19] is an iterarive process and it usually converges. K-Means is algorithm widely used iterative divisive technique, and the most important reasons are [7, 20, 21, 22]:

- Its time complexity is $O(Ikn)$ where n is the number of documents, k is the number of clusters and I is the number of iterations to converge. Tipically, k and I are fixed and set beforehand. So, the algorithm has a time complexity linear to the collection's size.
- Its spatial complexity is $O(k+n)$ plus the necessary space for the matrix.
- The document order of analyse is not influence by seed, although the initial seed influences the performance.
- It benefits from tipycal weighted matrix dispersity.

Algorithm 1. Typical K-Means

```
Input: Document collection and number of clusters (k)

Output: $k$ disjoint clusters

Step-1: $k$ centers of cluster are ramdonly selected.

Step-2: Each document is assigned to a cluster of
nearer center.

Step-3: The $k$ centers are recalculated.
```

```
Step-4: If the convergence critera is met then end,
else go to step-2

Note: Normally, the convergence critera is defined as
either no further changes in centers or the changes
are minimum.
```

1.3 The DBSCAN Algorithm

DBSCAN (Algorithm 2) [8] has two input parameters: ε, and MinEle. The first defines the maximum distance between two neighbor elements, and the second defines the minimum number of elements that must be neighbors in a cluster [6]. DBSCAN has the following characteristics [5, 8, 23, 24, 25, 26, 27]:

- It was designed to discover efficiently database clusters and noise.
- It works well on Euclidean spaces of two or three dimensions and on high dimensionality spaces too.
- It finds clusters of arbitrary size and shape, but it is influenced by the function used to calculate the distance between two objects.
- It is not deterministic in a strict sense, its behavior depends on the input order of objects. However, the results (clusters formed) are always the same.
- It is a very efficient and effective clustering algorithm. One important reason is that only needs one evaluation through the database.
- It is robust with respect to the noise.
- It works at every metric space, not only at vectorial space.
- It can be easily implemented.
- It presents a computational complexity $O(n^2)$, which can be improved using space indexes to $O(n\log(n))$.

Algorithm 2. DBSCAN

```
Input: $M \in \Re^{m \times n}, \varepsilon, MinEle \in \mathrm{N}$ (M is document collection)

Output: k disjoint clusters $\{\pi_j\}_{j=1}^{k}$, R (Noise)

Step-1: j=1

Step-2: For all documents d still to classify do

Step-2.1: V=EpsNeighborhood(d,M,$\varepsilon$) (Elements of M which
are to distance $\varepsilon$ of d)

Step-2.2: If |V|<MinEle then $R = R \cup d$
```

```
Step-2.3: Else
Step-2.3.1: $\pi_j = \pi_j \cup d$ and $V = V - d$
Step-2.3.2: While $V$ has elements
Step-2.3.2.1: Select $x \in V$
Step-2.3.2.2: $W$=EpsNeighborhood($x$,$M$,$\varepsilon$) (Elements of $M$
which are to distance $\varepsilon$ of $x$)
Step-2.3.2.3: If |$W$|<$MinEle$ then $V = V \cup W$ and $\pi_j = \pi_j \cup W$
Step-2.3.2.4: $V = V - x$
Step-2.3.3: $j = j + 1$
```

DBSCAN (Algorithm 2) verifies that documents are within a distance ε of each other in the collection. And it also verifies if *MinEle* documents are in this area. If an element has *MinEle* elements at distance ε then it is a core element of a cluster. If two core elements are to a distance ε then their clusters are merged. Elements of a cluster that do not include *MinEle* or more elements to a distance ε are edge elements. The algorithm finishes when all documents are assigned to a cluster or classified as noise. [8, 25, 27]

2 α-Bisecting Spherical K-Means

The α-Bisecting Spherical K-Means (Algorithm 3) [3] uses the α-Bisecting K-Means [3] to obtain an approximated initial solution and later it uses the Spherical K-Means to refine it, combining the best of both algorithms. The same idea was used in a previous version of the α -Bisecting Spherical K-Means [2], which was based on a more standard bisection version, without an α parameter.

The K-Means algorithm has been parallelizated the most [28, 29, 30, 31] starting almost always by a dataset centralized or repeated. But we start with an already distributed document collection.

Often in literature, one finds a simple parallelization of the K-Means algorithm, in which the sum and products are performed in parallel with one global reduction [28, 292, 30]. Here [3], we apply the same parallelization strategy to the α-Bisecting Spherical K-Means algorithm.

Algorithm 3. Parallel α-Bisecting Spherical K-Means

```
Note: This code is executed by every processing
element.
```

```
Input: $M_i \in \Re^{m \times ni}, tol, \max iter, k \in N$

Output: $k$ disjoint clusters $\{\pi_j\}_{j=1}^{k}$

Step-1: Compute k clusters $\{\pi_j^{(0)}\}_{j=1}^{k}$ using the parallel
•-Bisecting K-Means* and their normalized concept
vectors** $\{c_j^{(0)}\}_{j=1}^{k}$ in parallel. Initialize $t$=0.

Step-2: Build a new partition $\{\pi_j^{(t+1)}\}_{j=1}^{k}$ induced by $\{c_j^{(t)}\}_{j=1}^{k}$
according to: $\pi_j^{(t+1)} = \{\forall x \in M_i : x^T \cdot c_j^{(t)} > x^T \cdot c_l^{(t)}, 1 \le l \le k, j \ne l\}, 1 \le j \le k$

Step-3: Compute in parallel the new normalized concept
vector $\{c_j^{(t+1)}\}_{j=1}^{k}$ associated to the new partition

Step-4: Stop if $\left| f\left(\{\pi_j^{(t)}\}_{j=1}^{k}\right) - f\left(\{\pi_j^{(t+1)}\}_{j=1}^{k}\right) \le tol \cdot f\left(\{\pi_j^{(t+1)}\}_{j=1}^{k}\right) \right|$
( $f\left(\{\pi_j\}_{j=1}^{k}\right) = \sum_{j=1}^{k} \sum_{x \in \pi_j} x^T \cdot c_j$ ) is fulfilled or $t$ is equal to
$maxiter$ else increment $t$ and go to step-2.

* Each process element divides his subcolection into
two clusters. One formed by elements situed in a radio
$\alpha = 1 - \frac{\bar{x} - \sigma}{\sqrt[m]{k}}$ ($\bar{x}$ is the mean cosine distance and $\sigma$ the
standard deviation) and other formed by the rest of
elements. This last cluster is recursively divided in
two clusters until $k$ clusters are obtained. [2, 3]

** It basically computes a cluster normalized mean
with a reduction operations [3].
```

3 VDBSCAN

3.1 DBSCAN vs VDBSCAN

The original DBSCAN algorithm [8] uses the minimum number of elements that it looks for neighbors of an element. If the number of neighbors is less than the parameter then neighbors of this element are not considered. The main idea of the

VDBSCAN (Algorithm 4) is to change this feature. In VDBSCAN, the minimum number of elements is only used when a cluster has been classified a real cluster or noise. In other words, VDBSCAN always consider the neighborhood of one element, independently of its size.

With this variant the algorithm we yield two very important characteristics:

1. We reduce approximately to one the number of times we calculate the neighborhood in each algorithm iteration, and with fewer elements. The final cost is $O\left(\frac{n^2}{2}\right)$, which is half of the DBSCAN one.
2. We can obtain, with a heuristic, a good value to the minimum distance beetwen two elements into the same cluster [1].

Algorithm 4. VDBSCAN

```
Input: $M \in \Re^{m \times n}, \varepsilon, MinEle \in N$ ($M$ is document collection)

Output: $k$ disjoint clusters $\{\pi_j\}_{j=1}^{k}$, $R$ (Noise)

Step-1: $NoU=M$ and $j=1$

Step-2: While $NoU$ has elements

Step-2.1: Include random element of $NoU$ in $S$

Step-2.2: Initialize $U=\{\}$

Step-2.3: While $S$ has elements

Step-2.3.1: Select $x \in S$

Step-2.3.2: $U = U \cup \{x\}$ and $NoU = NoU - S$

Step-2.3.3: $V$= EpsNeighborhood($x, NoU, \varepsilon$) (Elements of
$NoU$ which are to distance $\varepsilon$ of $x$)

Step-2.3.4: $S = (S \cup V) - U$

Step-2.4: If $|U|<MinEle$ then $R = R \cup U$

Step-2.5: If $|U| \geq MinEle$ then $\pi_j = U$ and $j=j+1$

Step-2.6: $NoU = M - \{\pi_i\}_{i=1}^{j} - R$
```

3.2 Parallel VDBSCAN

The parallel VDBSCAN (Algorithm 5) focuses on information retrieval although it is not limited to this context, it can be used in any area where DBSCAN is used.

Every process element works with a part of the collection. And every processing element also saves global information about the location of classified documents within clusters, which documents are being classified and which are still pending. The main interprocess communication is used to maintain global information, and query neighborhood information regarding a given document.

Algorithm 5. Distributed VDBSCAN

```
Note: This code is executed by every processing
element.

Input: $M_i \in \Re^{m \times ni}, \varepsilon, MinEle \in N$

Output: $k$ disjoint clusters $\{\pi_j\}_{j=1}^{k}$, $R$ (Noise)

Step-1: $NoUG = M_i$ and $j=1$

Step-2: While $NoUG$ still has elements

Step-2.1: We include in $SG$ an aleatory element from
$NoUG$ (Sincronization, the same element in all
processing elements).

Step-2.2: We initialize $UG=\{\}$

Step-2.3: While $SG$ has elements

Step-2.3.1: We select $x \in SG$ (Sincronization, the same
$x$ element in all process elements).

Step-2.3.2: $UG = UG \cup \{x\}$

Step-2.3.3: $NoUG=NoUG-SG$

Step-2.3.4: $V$=PEpsNeighborhood($x$, $NoUG$, $\varepsilon$) (Each
processing element work with his part of $NoUG$ to
obtain which are to $\varepsilon$ distance of $x$).

Step-2.3.5: $SG = (SG \cup V) - UG$

Step-2.4: If $|UG| < MinEle$ Then $R = R \cup UG$
```

```
Step-2.5: If $|UG| \geq MinEle$ Then $\pi_j = UG$ and $j = j + 1$

Step-2.6: $NoUG = M_i - \{\pi_l\}_{l=1}^{j} - R$
```

We want to emphasize that in step-2.1 the selection of a random element causes a sincronization, because all processing elements have to use the same random element. In step-2.3.1, another sincronization is performed, because all processing elements have to know the selected element. The processing element that saves this selected element, distributes it, and the rest of processing elements to calculate the neighboord for their documents that are not used (step-2.3.4). The other communications are reductions, and these appear in step-2-4, step-2.5 and step-2.6.

4 Parallel Environment

Many times, the nature of the problem fores the data parallelization, because the data is already distributed and its centralization is not permissible. We focus on these cases.

The parallel algorithm is based in a SPMD (Single Program Multiple Data) model. The implementation has been developmented in a cluster of PCs. The communications are implemented using MPI [32, 33], thus, the code follows a standard and is more portable.

α-Bisecting Spherical K-Means and VDBSCAN start from a weighted matrix (sparse matrix). Three basic types of distributions exist when we work with a matrix: by rows (by terms), by colummuns (by documents) and by blocks.

We work in a distributed system where documents collection can be formed by real subcollections, which can physically be distributed. Therefore, given that the documents are already distributed, we are left with no other choice but to distribute the documents by documents (columns) as depicted in Fig. 1.

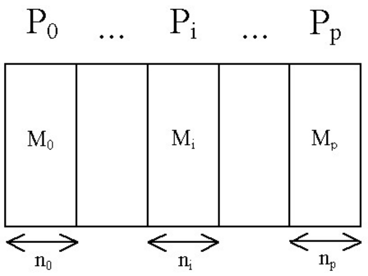


Fig. 1. Distribution by documents (by columns) of the document collection.

In this case, a distribution by rows (by terms) would incur great communication overhead during the query phase, so this distribution is rarely used [34]. The most common operation in the algorithm is the product array by array. In a distribution by rows, it would be implementated as a reduction with a cost equal to the number of columns. And, in a distribution by columns, it would be implemented as a broadcast with a cost equal to the number of rows. The retrieval information system have more columns than rows, it is an inherent proble. Thus, broadcast is less expensive. Beside, a broadcast implementation in TCP/IP model is simpler than a reduction implementation.

5 Experimental Results

For our experiments, wi use two collectios formed by a document collection, a queries set and their relevant documents in collection.

1. **Times Magazine 1963** contains 425 papers from Times Magazine of 1963. The main subjet is world news, mainly political news. It generates a weight matrix with 6545 rows (terms) and 425 columns (documents).
2. **TREC-DOE** has 226087 abstracts of USA department of energy. It generates a weight matrix with 87417 rows (terms) and 226087 columns (documents).

The first collection is used to evaluate retrieval performance and not overall computational performance. This collection generates a weighted matrix, which is a small sparse matrix with 6545 rows and 425 columns. Its sparsity factor is 0.027017. All values are beetwen 0 and 1.

The second collection represents a more realistic document collection and it is used to evaluate overall computational performance. The generated weighted matrix is a sparse matrix with 87417 rows and 226087 columns and its sparsity factor is 5.3025e-004. All values are beetwen 0 and 1.

5.1 Quality Retrieval Performance

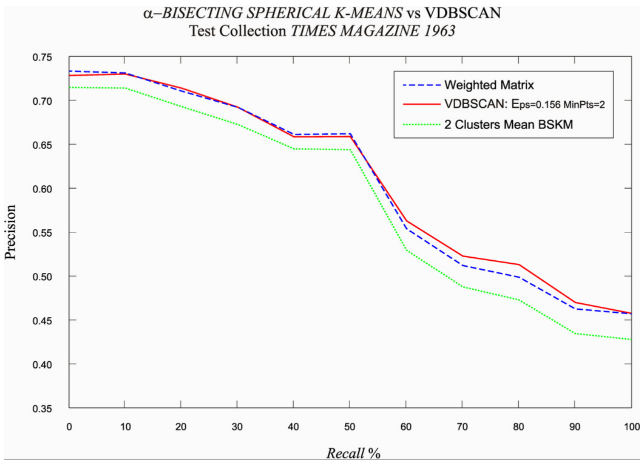


Fig. 2. Behavior of both methods with Times collection.

The studies are based at precision and recall measure, concretely we have used the curve for eleven standard leves, to compare both measure [34].

Fig. 2 shows how VDBSCAN performs better than α-Bisecting Spherical K-Means, here we show the mean performance curve for the α-Bisecting Spherical K-Means because its behavior depends on the unknown parameters in the algorithms which yielding to an unpredictable set of curves.

5.2 Computational Performance

In this subsection, we take a closer look at the different phases of the algorithms and compared them using the experiments set around the two document collections, TIMES and TREC-DOE. First, we look at the Modelization phase and then the Query Evaluation Phase.

5.2.1 The Modelization Phase

From the computational times of the Modelization phase, we can observe that the VDBSCAN is more expensive than α-Bisecting Spherical K-Means and the VDBSCAN's parallelization is more beneficial to its algorithmic performance. Resulting in a greater impact as the number of documents in the collection is larger.

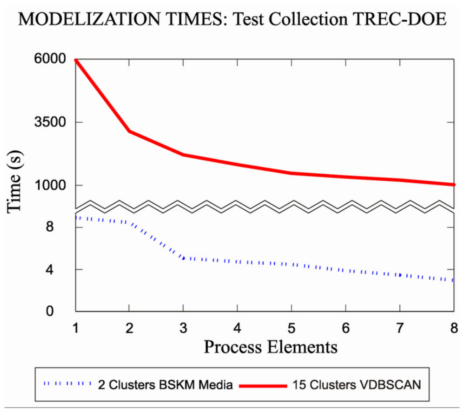
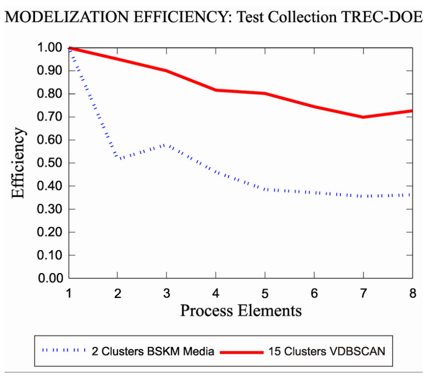


Fig. 3. Parallel performance of the modelization phase for the TREC-DOE collection.

Results from the experiments using the TIMES document collection show very little difference in the computational time. The number of documents in the TREC-DOE collection is larger than in the TIMES one, thus the difference in computational time between the two algorithms is emphasized and well depicted by this experiment (see Fig. 3). Increasing the number of processing elements in the VDBSCAN also exhibits better improvement in its performance. As shown in Fig. 4, the efficiency of VDBSCAN is consistently greater than 70% when scaling the number of processing elements. The efficiency of the α-Bisecting Spherical K-Means is not nearly as good. The speed-ups obtained with the VDBSCAN are also greater. When using the TIMES document collection, the efficiency started to degrade for both algorithms after two processing elements and this is due to the small size of the collection.



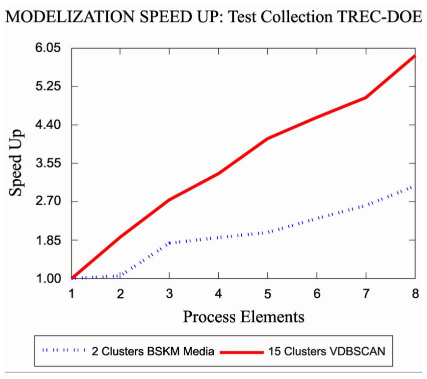


Fig. 4. Speed up and efficiency of the modelization phase for the TREC-DOE collection.

The results of efficiency and speed up study (Fig. 4) present analogous behavior to time study. When we use a little collection (Times) efficiency is acceptable using until two process elements, and there are not important differences with none of the methods. When we use a big collection (TREC-DOE) performance improve outstandingly. VDBSCAN maintains the efficiencia at any moment greater of 70%.

5.3.2 The Query Evaluation Phase

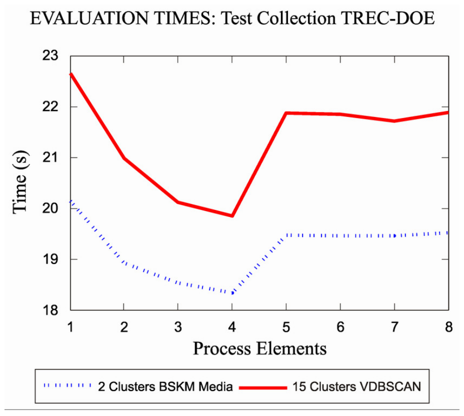


Fig. 5. Parallel performance of the query evaluation phase for the TREC-DOE collection.

α-Bisecting Spherical K-Means always shows a smaller evaluation time than VDBSCAN, although the difference is small, mainly when the number of process elements used grows. In this analysis, we use the mean behavior of α-Bisecting

Spherical K-Means. The parallel performances improve until four process elements, after performance decrease, see Fig. 5.

In Fig. 6 we can see the efficiency and speed up of the parallel implementations. Both algorithms exhibit poor efficiency, which is under 50% even with 2 processing elements, it drops drastically as the number of processing elements is increased. And speed up grow to four process elements, after it decrease. VDBSCAN reaches a little more speed up.

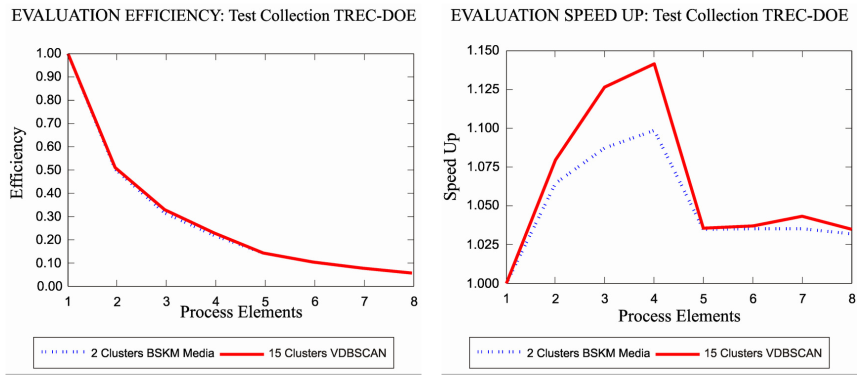


Fig. 6. Speed up and efficiency of the query evaluation phase for the TREC-DOE collection

6 Conclusions

VDBSCAN improves the performance of DBSCAN by simply introducing the use heuristics to select input parameters to the algorithm. In the α-Bisecting Spherical K-Means, the number of clusters to build, the principal parameters and initial seed are unknowns that lead to a variable performance, making the algorithms computational performance highly dependent on a good selection of these parameters.

The performance of the information retrieval for VDBSCAN, with the correct parametric selection, is similar to a weighted matrix. Sometimes exhibiting a minor improvement or worsening. When comparing information retrieval performance of the VDBSCAN and α-Bisecting Spherical K-Means, the VDBSCAN outperforms its counterpart.

Our computational experiments showed that the Modelization phase of VDBSCAN is more expensive than α-Bisecting Spherical K-Means, but the parallel performance of this phase in VDBSCAN is better. Additionally, the Modelization phase is only performed once, thus it's equivalent to an initial setup cost that is amortized by all the other benefits of the implementation.

For the evaluation time, α-Bisecting Spherical K-Means is slightly faster than VDBSCAN, but the difference is reduced as the number of processing elements is increased. Thus even when VDBSCAN's Modelization is computationally more expensive and slower than the α-Bisecting Spherical K-Means, its parallel performance is better as demonstrated by the speed-ups and efficiency analysis. Moreover, the computational time differences of the Modelization and Query Evaluation phases that tend to favor the α-Bisecting Spherical KMeans algorithm, are all counter-argued by the better performance of the VDBSCAN in the information retrieval time and the easy initial parametric selection.

The poor parallel performance found in the evaluation phase is intrinsic to the problems considered here because they induced poor data locality in its core operations. Namely, the sparsity of the matrices that lead to poor performance of the matrix-vector or vector-vector operations. Future implementations should consider the optimization of these products for very sparse arrays. Even with our current implementation of the products, we reduce the overall execution time as we increase the number of processing elements.

Acknowledgements

Partial support to perform this work has been received from the Spanish Ministerio de Educación y Ciencia under projects ENE2008-02669, TIN2008-06570-C04-04.

References

1. Jiménez, D. and Vidal, V: VDBSCAN: Variant DBSCAN faster. In phase of publication
2. Jiménez, D., Vidal, V., Enguix, C.F.: A comparison of experiments with the bisecting-spherical k-means clustering and svd algorithms. Actas congreso JOTRI (2002)
3. Jiménez, D., Vidal, V. In: Parallel Implementation of Information Retrieval Clustering Models. Volume 3402/2005 of Lecture Notes in Computer Science. Springer Berlin (2005) 129_141
4. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17 (2001) 107_145
5. Andritsos, P.: Data clustering techniques. Technical Report CSRG-443 (2002)
6. Halkidi, M., Vazirgiannis, M.: Clustering validity assessment: Finding the optimal partitioning of a data set. In: ICDM. (2001) 187_194
7. Jain, A.K., Murty, M.N., Flynn, P.J.:Data clustering: A review. ACM Computing Surveys 31 (1999) 264_323
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: In proceedings of 2nd International Conference on Knowledge Discovery and Data Mining. (1996) 226_231
9. Kailing, K., Kriegel, H.P., Kroeger, P.: Density-connected subspace clustering for high-dimensional data. In: In: Proc. SDM. 2004. (2004) 246 - 257
10. Jahirabadkar, S., Kulkarni, P.: Isc - intelligent subspace clustering, a desity based clustering approach for high dimensional dataset. World Academy of Science, Engineering and Technology 55 (2009) 69 - 73
11. Achtert, E., peter Kriegel, H., Pryakhin, A., Schubert, M.: Hierarchical densitybased clustering for multi-represented objects. In: In: Workshop on Mining Complex Data (MCD 2005) at ICDM05. (2005)
12. Huang, J., Ertekin, S., Giles, C.: Efficient name disambiguation for large-scale databases. Knowledge Discovery in Databases: PKDD 2006 (2006) 536 - 544

13. Liu, S., Dou, Z., Li, F., Huang, Y.: A new ant colony clustering algorithm based on dbscan. In: Proceedings of the 3rd International Conference on Machine Learning and Cybernetics. (2004) 1491 - 1496
14. Roy, S., Bhattacharyya, D.K.: An approach to find embedded clusters using density based techniques. In: Distributed Computing and Internet Technology, Second International Conference. (2005) 523 - 535
15. Ruiz, C., Spiliopoulou, M., Ruiz, E.M.: C-dbscan: Density-based clustering with constraints. In: Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. (2007) 216 - 223
16. Savaresi, S., Boley, D.L., Bittanti, S., Gazzaniga, G.: Choosing the cluster to split in bisecting divisive clustering algorithms. Technical report (2000)
17. Savaresi, S.M., Boley, D.L.: On the performance of bisecting k-means and pddp. First Siam International Conference on Data Mining (2001)
18. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. KDD-2000 Workshop on Text Mining (2000)
19. Hartigan, J.: Clustering Algorithms. Wiley (1975)
20. Dhillon, I.S., Fan, J., Guan, Y.: Efficient clustering of very large document collections. In R. Grossman, C. Kamath, V.K., Namburu, R., eds.: Data Mining for Scientifec and Engineering Applications. Kluwer Academic Publishers (2001) Invited Book Chapter.
21. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Technical report, IBM (2000)
22. Bradley, P.S., Fayyad, U.M.: Re_ning initial points for k-means clustering. In: Proc. 15th International Conf. on Machine learning, Morgan Kaufmann, San Francisco, CA (1998) 91_99
23. Kailing, K.: New Techniques for Clustering Complex Objects. PhD thesis, Ludwig-Maximilians Universität München, Munich, Germany (2004)
24. Su, Z., Yang, Q., Zhang, H., Xu, X., Hu, Y.: Correlation-based document clustering using web logs. In: Proc. of the 34th Hawaii International Conference on System Sciences, IEEE Computer Society (2001) 5022
25. Iváncsy, R., Babos, A., Legány, C.: Analysis and extensions of popular clustering algorithms. In: Proc. of the 6th International Symposium of Hungarian Researchers on Computational Intelligence. (2005) 390_400
26. Januzaj, E., Kriegel, H.P., Pfeie, M.: Towards effective and efficient distributed clustering. In: Workshop on Clustering Large Data Sets, 3rd Int. Conf. on Data Mining (ICDM 2003). (2003) 49 _ 58
27. Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: SIAM International Conference on Data Mining (SDM'03). (2003)
28. Kantabutra, S., Couch, A.L.: Parallel k-means clustering algorithm on nows. NECTEC Technical Journal 1 (2000) 243_248
29. Xu, S., Zhang, J.: A hibrid parallel web document clustering algorithm and its performance study. (2003)
30. Dhillon, I.S., Modha, D.S.: A parallel data-clustering algorithm on distributed memory multiprocessors. In: Large-Scale Parallel Data Mining, Lecture Notes in Arti_cial Intelligence. Volume 1759. Springer-Verlag, New York (2000) 245_260
31. Li, X., Fang, Z.: Parallel clustering algorithms. Parallel Computing 11 (1989) 275_290
32. The MPI standard, http://www.mcs.anl.gov/research/projects/mpi/index.htm
33. Message Passing Interface Forum, http://www.mpi-forum.org/
34. Baeza-Yates R. A. et al, Modern Information Retrieval. ACM Press / Addison-Wesley (1999).