MR-Mafia: Parallel Subspace Clustering Algorithm Based on MapReduce For Large Multi-dimensional Datasets

Zhipeng Gao
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
gaozhipeng@bupt.edu.cn

Yidan Fan
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
fanyidanyouxiang@163.com

Kun Niu
School of Software Engineering
Beijing University of Posts and Telecommunications
Beijing, China
niukun@bupt.edu.cn

Zhenyi Ying
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
yingzhenyi0619@gmail.com

***Abstract*—The mission of subspace clustering is to find hidden clusters exist in different subspaces within a dataset. In recent years, with the exponential growth of data size and data dimensions, traditional subspace clustering algorithms become inefficient as well as ineffective while extracting knowledge in the big data environment, resulting in an emergent need to design efficient parallel distributed subspace clustering algorithms to handle large multi-dimensional data with an acceptable computational cost. In this paper, we introduce MR-Mafia: a parallel mafia subspace clustering algorithm based on MapReduce. The algorithm takes advantage of MapReduce's data partitioning and task parallelism and achieves a good tradeoff between the cost for disk accesses and communication cost. The experimental results show near linear speedups and demonstrate the high scalability and great application prospects of the proposed algorithm.**

Keywords- Subspace clustering, Mafia, MapReduce, Parallel and distributed

I. INTRODUCTION

With the increase of the internet and the improvement in communications, information in all walks of life is now more abundant than ever and accumulates daily in databases. The field of data mining that extract information and knowledge from large datasets have become very popular. Clustering analysis is an active topic of data mining, which can automatically group unlabeled data into clusters of similar characteristics, but most existing clustering algorithms cannot work on the increasing high dimensional datasets because of the curse of dimensionality. Instead of considering all dimensions of an input dataset, subspace clustering manages high dimensional data by finding clusters under different subsets of dimensions called subspaces within a dataset [1], representative algorithms such as CLIQUE [2], MAFIA [3] and PROCLUS [4]. As data accumulates in bulk volumes and goes beyond the processing power of single-processor machines [5], traditional subspace clustering algorithms cannot meet the efficiency requirements nor directly process the big data. Thus create a growing demand to increase the scalability and performance of existing algorithms. Parallelization and distributed computing comes out as a natural solution to enable subspace clustering algorithms to scale up to extreme sizes and high dimensions. Recently, a distributed programming model called MapReduce [6] which can deal with big data in a highly parallel manner causes researchers and developers concerns. It aims at supporting parallel and distributed computation on large datasets through the use of a large cluster of machines while making each machine independently running the single-node logic with scalability and fault-tolerance guarantees. MapReduce is great for batch processing and has become the most widely used framework for mining large-scale datasets in parallel and distributed environment. But according to our survey, we found the lack of improved subspace clustering algorithms on top of MapReduce.

2375-9356/18/$31.00 ©2018 IEEE
DOI 10.1109/BigComp.2018.00045

In this paper, we propose MR-Mafia, a parallel and distributed algorithm on top of MapReduce that overcomes the memory and CPU speed limitations of Mafia and highly improves its scalability. While Mafia runs on a single machine and fail to process large high-dimensional datasets, MR-Mafia takes advantage of MapReduce and is able to execute Mafia in parallel under the distributed environment. MR-Mafia avoids unnecessary I/O cost and minimize the communication cost between nodes, achieves independent calculation and load balance in each node as well. By using indexing, MR-Mafia only stores the whole dataset in the main node instead of replicating it in each node and thus bring less computational cost.

The main content of the whole paper and viewpoint of each sector are as follows: Section 1 presents an overview of the related work. Section 2 we briefly introduce Mafia algorithm and present our MR-Mafia. In Section 3 we test and evaluate our algorithm on scale-up and accuracy. Finally, we make a brief summary and outlook on our work in Section 4.

II. RELATED WORK

A. Subspace Clustering

With the advent of the era of big data, collecting and storing data becomes cheap, users tend to record everything without considering the relevance for their task, thus make the dimension of the datasets to be processed very high. The problem of clustering high dimensional data faced by clustering analysis is crucial and is closely related to whether many clustering algorithms are suitable to be used in many fields or not. Often in high dimensional data, many dimensions are irrelevant and may mask existing clusters in noisy data [1]. When the number of dimensions in a dataset increases to very high, it will be common for all points in the dataset to be almost equally far apart and distance measures become meaningless, and because of the dimension disaster caused by the exponential increase of dimensions, traditional distance-based clustering algorithms are prone to be inefficient as well as ineffective, frequently unable to meet quality needs of clustering results. To tackle this problem, R. Agrawal [2] outlines for the first time the concept of subspace clustering, i.e. finding a multi-subspace representation that best fits a collection of points taken from a high-dimensional space. In light of the finding that even though data sets are high dimensional, their intrinsic dimension is often much smaller than the dimension of the ambient space so that a multi-dimensional dataset can often be better understood by grouping it in subspaces. In the past two decades, a number of approaches to subspace clustering have been proposed and they are mainly divided into bottom-up and top-down groups based on their approach to searching for subspaces.

The bottom-up search method generates bins for each dimension and forms a multi-dimensional grid firstly, start from low-D subspaces and search higher-D subspaces only when there may be clusters in such subspaces by taking advantage of the downward closure property of the Apriori principle. The algorithm proceeds until there are no more dense units found. CLIQUE, ENCLUS, MAFIA, CBF and CLTree are all density and grid based bottom-up subspace clustering algorithms. The grid used by CLIQUE and ENCLUS to divide each dimension is static and its size is set by user parameters. MAFIA extends CLIQUE via applying an adaptive grid method. CBF uses a cell creation algorithm that creates optimal partitions whereas CLTree uses a decision tree based strategy to determine the cut-points for the bins on each dimension. Most of the bottom-up algorithms have the capability to find clusters in different size and shape and scale linearly with the dimension and size of data sets [1].

The top-down search method starts from full space and searches smaller subspace recursively. The algorithms make the use of full set of dimensions and start from an initial set of subspace with equally weighted dimensions, then update of the weight for each dimension in each cluster and regenerate the set of subspaces iteratively. The iterative process continues until the clustering result meets the quality requirement and the clusters generated by this method are non-overlapping. As multiple iterations are costly, top-down algorithm often use a sampling technique to improve performance. The two parameters often required by top-down algorithms are the number of clusters and the average number of dimensions for the clusters, the output clusters are of similar dimensionality due to the second parameter. PROCLUS, ORCLUS and FINDIT are all distance and partition based top-down subspace clustering algorithms that use different measure strategies [1, 7].

The major challenge for data clustering lies in its scalability and availability especially when it comes to very large databases, algorithms always have to balance the trade-off between accuracy and runtime [8]. Currently no subspace clustering algorithm can process very large datasets in feasible time, parallel and distributed computing seems to be ideally suited to address big data issues [9]. In this case we have to re-think, re-design and re-implement existing subspace clustering algorithms in order to be applicable in parallel and distributed processing [9].

B. MapReduce

With the advent of the era of big data, lots of data is being collected and warehoused, which need data mining techniques to mine the potential valuable information. Traditional data mining algorithms have found it difficult to extract value and insights from massive datasets running on single machines [10]. How to gain the effective knowledge from big data have attracted tremendous interest from domestic and abroad. Hadoop MapReduce which was proposed in 2004 by Google is a software framework for easily writing applications which process and generate vast amounts of data with a parallel, distributed algorithm on large clusters in a reliable, fault-tolerant manner [6]. A MapReduce program is composed of a Map() method that performs filtering and sorting and a Reduce() method that performs a summary operation [11].

A work flow of MapReduce is showed by Figure 1 and it can be generally divided into five steps: prepare the Map() input, run the user-provided Map() code, "shuffle" the Map output to the Reduce processors, run the user-provided Reduce() code and produce the final output. Provided that each mapping and reducing operation is independent of the others, all maps and reduces can be performed in parallel. The input and output of Map and Reduce functions must be in form of (key, value) pairs [6].

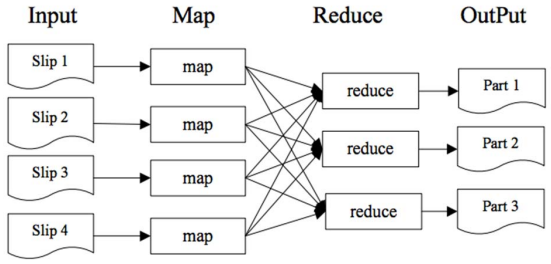

Figure 1. MapReduce operation process

The programmer is oblivious of the details of the data distribution, replication, load balancing, etc and needs to provide only two functions, a map and a reduce [12]. The MapReduce framework orchestrates the processing by automatically parallelizing the computation across large-scale clusters of machines, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance [11].

Due to parallel computing nature of MapReduce, parallel data mining algorithms with the help of the MapReduce model has received considerable attention from the research community [13]. By applying the MapReduce paradigm, researchers have accomplished parallel scaling up of many data mining algorithms on large datasets, such as Kmeans [8] and Canopy [14].

III. MR-MAFIA: PARALLEL SUBSPACE CLUSTERING ALGORITHM BASED ON MAPREDUCE

A. *Mafia algorithm*

Mafia is a density and grid based clustering, which is an extension of Clique, one of the first algorithms that aim to find clusters within subspaces of the dataset [1]. The difference between them lies in the adaptive grid, Clique generates 1-D histogram for each dimension and selects dense units whose density above a given threshold, the histogram is constructed by partitioning each dimension into a number of non-overlapping equal length intervals. This method may divide dense regions with clusters into a large amount of candidate dense units or confuse noise data with dense units. Instead of directly using the uniform grid, Mafia follows an adaptive interval size to partition the dimension relying on the distribution of data in a particular dimension. By requiring a user-defined threshold, Mafia merges adjacent intervals within the given threshold to form larger intervals. Like Clique, Mafia take advantage of the downward closure property of density to reduce the search space. But Mafia's number of generated candidate is much larger compared to Clique, because k-dimensional candidate dense units are obtained by merging (k-1)-dimensional dense units which share any (k-2) dimensions (not only first dimensions). Mafia makes an order of magnitude improvement in the computation time over methods like Clique and provides much better quality of clustering [4]. As Mafia is one of the most representative top-down algorithms and has the nature characteristic of parallelism, we try to make further improvement to gain better performance and expand application range.

B. *MR-Mafia algorithm*

To implement Mafia on MapReduce framework, the main tasks are to design Map and Reduce functions. The Mafia algorithm can be generally divided into two MapReduce jobs, first generate the adaptive grid, then form candidate subspaces and select dense subspaces iteratively by performing a bottom-up traversal. Mafia requires three user-provided parameters: density threshold, window-merge threshold and default grid size, so does MR-Mafia. Table 1 lists the used symbols.

TABLE I. TABLE OF SYMBOLS

Symbols	Definitions
^{d}S	A input d-mensional dataset
d	Dimensionality of dataset ^{d}S
N_s	Number of instances in the dataset
N_p	Initial partition number of each dimension
T_c	Density threshold of clusters
T_w	Merge threshold of adjacent windows

Defining the input dataset ^{d}S a d-dimensional dataset and let each line in ^{d}S be represented as a 1-D feature vector, {Line_i, D_{i1}, D_{i2}, ..., D_{ij}, ..., D_{id}}. For the generation of the adaptive grid, MR-Mafia first call the Map function to split each vector into key-value pairs based on dimension, i.e. <1, D_{i1}>,...,<j, D_{ij}>,...., <d, D_{id}>, then all of the values under the same dimension will automatically be grouped into the same reducer, the Reduce function will create histogram for each dimension and mark dense subspaces according to the distribution of data. For a certain dimension j, Rj is the range of it and it will be divided into Np intervals, $W_{jk} = [l_{jk}, r_{jk})$ is the No.k window, $W_{jk} \in R_j$. Algorithm 1 describes its processing procedure. The average time complexity of the merge process is $O(N_p \log N_p)$. As each Reducer deals with one dimension, the computational time increases linearly with the increase of dimensionality.

Algorithm 1 Adaptive grid computation

for each input <key,value> pair, <j,<D_{j1}, D_{j2}, ..., D_{jN}>>, j∈ (1,...d), D_{jk} ∈ R_j **do**
  Divide R_j into N_p intervals.
  Compute the the number of data points contained in each window and set the value of the window to it.
  Merge two adjacent windows from left to right if the distance between them within T_w iteratively until no windows can be merged.
  if number of windows == 1 **then**
    Divide R_j based on the default grid size.
  for each 1-D window W_{ji} on dimension j **do**
    if its value is no less than T_c **then** mark W_{ji} as a 1-D dense subspace
    Take j as key1' and take the detail of W_{ji} as value1'
    output <key1', value1'>.
    Take W_{ji} as key2' and construct value2' as a string comprise of the data points contained in W_{ji}
    output < key2', value2'>.
end

Focusing on the obtained adaptive grid and no need to pay attention to the concrete data, we can gain 2-D candidate subspace by direct combination of every two 1-D dense subspace and form k-D candidate subspace by combining (k-1)-D dense subspaces which share any (k-2) subspaces, the maximum time complexity of this process is $O(N^2)$, N is the total number of the (k-1)-D dense subspaces. Each subspace is represented as a sequence of dense windows numbered consecutively by the dimension id. Thus the file that stores grid information is small in size, we utilize the Distributed Cache to distribute the corresponding relation to every node in the cluster and work directly with grid information to form candidate subspaces when run setup() method of the second job in Map phase. The pseudo code of the process is shown in Algorithm 2.

Algorithm 2 Form candidate subspaces

for the adaptive gird **do**
  if the grid stores 1-D dense subspaces **then**
    Combine each two different 1-D dense subspaces to form 2-D candidate subspace.
  else for each two (k-1)-D dense subspaces **do**
    if they share the same (k-2)-D subspaces **then**
      Form the k-D candidate subspace and store the correspondence between the candidate subspace and its two components in the Hash Map.
end

For each input line <$S_{k-1,i}$, D_i>, where $S_{k-1,i}$ is the i^{th} (k-1)-D dense subspace that arranges every window id of the k-1 dimensions involved in sequence, D_i is a string comprise of the data points contained in $S_{k-1,i}$, a Map function is used to match each $S_{k-1,i}$ or $S_{k-1,j}$ to the corresponding k-D candidate subspace $S_{k,ij}$ though the HashMap obtained from setup function, the key-value pair < $S_{k,ij}$, D_i> is the output of Mapper. By this way, each Reducer handles a k-D candidate subspace and the data points contained in two (k-1)-D dense subspaces comprise it, first compute the intersection of two (k-1)-D dense subspaces D_{ij} and calculate the number of the co-exist data points. If the number is no less than Tc, then output the k-D dense subspace < $S_{k,ij}$, D_{ij}> and $S_{k,ij}$ with the related k dimensions <d_1...d_k, $S_{k,ij}$ >, they will be the input of the next iteration. Otherwise, output the two (k-1)-D dense cluster <$S_{k-1,i}$, D_i> and <$S_{k-1,j}$, D_j>. The above steps or the second MapReduce job will be repeated until no higher-dimensional subspaces are found or no more than two (k-1)-D dense subspaces can be merged into a k-D dense subspace.

IV. Performance Evaluation

We did a series of experiments on both synthetic and real datasets in order to evaluate the performance of MR-Mafia. The experiments were executed on a master node consist of 16-core processor and 64GB of memory and a commodity cluster of 30 slave nodes. Hadoop was configured to use 10GB of memory per Map or Reduce task. The machines were running CentOS 6.6 operating system and Hadoop 2.6.2.

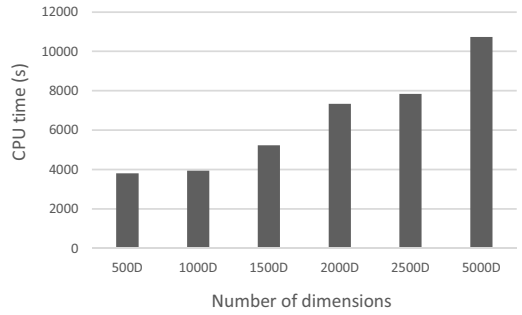

Figure 2. Execution time used for adaptive grid formation (N_s =15000)

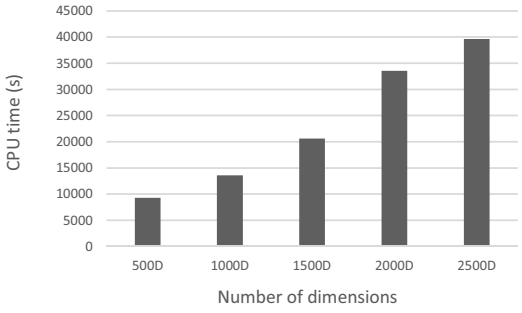

Figure 3. Execution time w.r.t. dimensionality (N_s =20000)

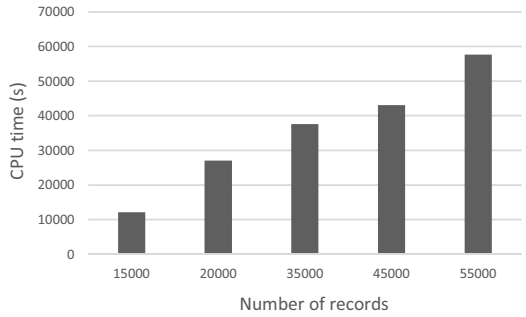

Figure 4. Execution time w.r.t. dataset size(d =2000)

A. Scalability with Dataset Size and Dimensionality

We have performed the scalability evaluation on simulated data with different size and dimensionality. Figure 2 and Figure 3 show the results for scalability with the dataset size of 1600 records with the dimensionality ranging from 500 to 5000. We can see that MR-Mafia scales well with the increase in the number of dimensions, both total CPU time and the time spent on forming adaptive grids almost showed linear relation. Meanwhile, Figure 4 also shows MR-Mafia achieved near linear behavior with the number of rows in the 2000-dimensional dataset increased from 15000 to 55000 while Mafia algorithm which was implemented in a single-machine environment failed to handle the datasets containing over a thousand high-dimensional records when used the same fine initial partition as MR-Mafia. Therefore, MR-Mafia can cope with the size and dimensionality of the dataset well.

B. Accuracy

In order to evaluate the accuracy of MR-Mafia algorithm, we used both simulated dataset and real dataset to compare the results of MR-Mafia, Mafia and Clique. The accuracy is measured by counting the number of correctly assigned instances and dividing it by the total amount of instances. As we had excepted, Clique resulted in a loss of data accuracy due to the fixed grid adopted especially when dealing with the simulated data, Table 2 shows the comparison of accuracy of results between MR-Mafia and Clique. Meanwhile, the matched experiment results of MR-Mafia and Mafia showed that MR-Mafia well maintains the accuracy and superiority of Mafia.

TABLE II. COMPARISON OF ACCURACY

Algorithm	Simulated dataset ($d = 500$, $N_s = 500$)	Real dataset ($d = 50$, $N_s = 7680$)
MR-Mafia	0.86	0.71
Clique	0.62	0.60

V. CONCLUSIONS

In this paper we present MR-Mafia: a parallel and distributed algorithm by making full use of MapReduce. MR-Mafia takes advantage of data partition and parallel mechanism and improves traditional Mafia algorithm on scalability while retain the accuracy of the results, thus makes Mafia can be used in distributed environment and deal with large high-dimensional datasets effectively.

ACKNOWLEDGMENT

This study is supported by National Key Research and Development Program (2016YFE0204500), and National Science & Technology Pillar Program (2015BAH03F02).

REFERENCES

[1] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: A Review," SIGKDD Explorations, vol. 6, no. 1, pp. 90-105, 2004.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," Proc. ACM SIGMOD Int. Conf. Management of Data, pp. 94–105, 1998.

[3] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," Proceedings of the 1999 ACM SIGMOD international conference on Management of data, pp. 61–72,1999.

[4] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and scalable subspace clustering for very large data sets," Technical Report CPDC-TR-9906-010, Northwestern University, June 1999.

[5] Jianwei Li, Ying Liu, Wei-keng Liao, and Alok Choudhary, "Parallel Data Mining Algorithms for Association Rules and Clustering," Handbook of Parallel Computing: Models, Algorithms and Applications. Sanguthevar Rajasekaran and John Reif, ed., CRC Press, 2007.

[6] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communications of the ACM, vol.51, no. 1, pp. 107-113, January 2008.

[7] E. Muller, S. Gunnemann, I. Assent, and T. Seidl, ¨Evaluating clustering in subspace projections of high dimensional data," PVLDB, vol. 2, no. 1, pp.1270–1281, 2009.

[8] Weizhong Zhao, Huifang Ma, and Qing He, "Parallel k-means clustering based on MapReduce," SpringerVerlag Berlin Heidelberg, LNCS 5931, pp. 674–679, 2009.

[9] Ferreira Cordeiro, Robson Leonardo, et al, "Clustering very large multi-dimensional datasets with mapreduce," Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 690-698, 2011.

[10] Liu X, Wang X, Matwin S and Nathalie J, "Meta-mapreduce for scalable data mining," Journal of Big Data, vol. 2, no. 1, pp. 1-21, 2015.

[11] https://ja.wikipedia.org/wiki/MapReduce.

[12] Tsourakakis, Charalampos E, "Data Mining with MAPREDUCE: Graph and Tensor Algorithms with Applications," Diss. Master's thesis, Carnegie Mellon University, 2010.

[13] Nandakumar, D. R. A. N., and Nandita Yambem, "A Survey on Data Mining Algorithms on Apache Hadoop Platform," International Journal of Emerging Technology and Advanced Engineering, vol. 4, no.1, pp. 563-565, 2014.

[14] Poonam Ghulia, Archit Shuklaa, Raj Kirana, Sheraaz Jasona, and Rajashree Shettara, "Multidimensional canopy clustering on iterative MapReduce framework using Elefig tool," IETE Journal of Research, vol. 61, no. 1, pp. 14-21, 2015.

[15] Maniatty, William A., Mohammed J. Zaki, "Systems support for scalable data mining," ACM SIGKDD Explorations Newsletter, vol. 2, no. 2, pp. 56-65, 2000.

[16] Bissam Zerhari, Ayoub Ait Lahcen and Salma Mouline, "big data clustering:Algorithms and Challenges," BDCA'15 program, Tetuan, May 2015.

[17] Dr. Sujni. Paul, "Parallel and Distributed Data Mining," New Fundamental Technologies in Data Mining, book edited by Kimito Funatsu, ISBN 978-953-307-547-1, January 21, 2011.

[18] R. Vidal, "Subspace Clustering," Signal Processing Magazine, vol. 28, no. 2, pp. 52-68, 2011.

[19] Shraddha Masih, Sanjay Tanwani, "Data Mining Techniques in Parallel and Distributed Environment - A Comprehensive Survey," International Journal of Emerging Technology and Advanced Engineering, vol. 4, no. 3,pp. 453-461, March 2014.

[20] B.Hari Babu, N.Subash Chandra, and T. Venu Gopal, "Clustering Algorithms For High Dimensional Data – A Survey Of Issues And Existing Approaches," Special Issue of International Journal of Computer Science & Informatics, vol. 2, no. 1, 2, pp. 142-148, 2011.

[21] Btissam Zerhari, Ayoub Ait Lahcen, and Salma Mouline, "Big Data Clustering: Algorithms and Challenges," International conference on Big Data, Cloud and Applications, Tetuan, Morocco, May 2015.

[22] Nagesh, H. S., S. Goil, and A. Choudhary, "A scalable parallel subspace clustering algorithm for massive data sets," ICPP, Toronto, pp. 477-484, August 2000.

[23] Amardeep Kaur, Amitava Datta, "A Novel Algorithm for Fast and Scalable Subspace Clustering of High-Dimensional Data," Journal of Big Data, vol. 2, no. 17, pp.1-24, 2015.

[24] William A. Maniatty and Mohammed J. Zaki, "A requirements analysis for parallel kdd systems," Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing, pp. 358-365, May 2000.

[25] Kim W, "Parallel clustering algorithms: survey," CSC 8530 parallel algorithms, 2009.

[26] B. Zhu, A. Mara, and A. Mozo, "CLUS: parallel subspace clustering algorithm on spark," ADBIS (Short Papers and Workshops) 2015, pp. 175-185, 2015.

[27] H. Xiao, "Towards parallel and distributed computing in large-scale data mining: A survey," Technical Report, Technical University of Munich, Germany, pp. 1-30, April 2010.