Efficient Results Merging for Parallel Data Clustering Using MapReduce

Abdelhak Bousbaci and Nadjet Kamel

Abstract Data clustering is partitioning data into sub-groups using a distance measure. Clustering a large data amount requires an important execution time. Several works have been proposed to overcome this problem using parallelism. One of the parallel techniques consists in partitioning data and processing each partition apart, the results obtained from each partition are merged to get the final clusters configuration. Using an inappropriate merging technique leads to an inaccurate final centroids and a middling clustering quality. In this paper, we propose two merging techniques to improve the clustering quality.

In a first solution, the results are merged using the K-means algorithm, and in a second one using the genetic algorithm. The results proved the efficiency of the proposed strategies.

Keywords Data clustering · K-means · Parallelism · MapReduce · Results merging · Genetic algorithm

1 Introduction

Data clustering is partitioning data into sub-groups using a distance function such that data in the same group are similar [1]. Clustering a large data amount requires an important execution time. In the literature, several works have been proposed to overcome this problem using parallelism [2] [3] . One of parallelism techniques consists in partitioning data and processing each partition apart [4] [5] [6], using a cluster of machines or a multi-core CPU. In both cases, the results obtained from each partition are merged to obtain the final clusters configuration.

A. Bousbaci(✉) · N. Kamel
LRIA, Computer Science Department, USTHB Algiers, Bab Ezzouar, Algeria
e-mail: abousbaci@usthb.dz

N. Kamel
Computer Science Department, Faculty of Sciences, UFAS Setif, Setif, Algeria
e-mail: nkamel@univ-setif.dz

S. Omatu et al. (eds.), *DCAI, 13th International Conference*,
Advances in Intelligent Systems and Computing 474,
DOI: 10.1007/978-3-319-40162-1_38

The merging techniques have a direct impact on the clustering quality. Using an inappropriate fusion technique, results inaccurate centroids and a middling clustering quality.

One of the existing merging methods, consists in gathering the clusters that overlap in data space, according to a defined threshold. If we have two clusters A and B, and there are data points from one of these two clusters that can belong to the other one, then these two clusters are merged together and constitute a new single cluster. In [4], the authors proposed a parallelization for the k-means algorithm. In their solution, the initial data is divided randomly on several machines and each one processes its own data partition independently. In a final step, the clusters from all the machines that overlap in data space were merged. This method showed efficiency, but the obtained final number of centroids, can differ from the one set in the beginning (K). This can affect the clustering quality in the case where the initial chosen number of clusters (K) is the optimal one.

Another existing simple merging technique consists in merging the centroids from the different partitions, such that each centroid from a given partition is merged with the *N-1* nearest centroids from each of the other partitions; where N is the number of all the partitions. This merging method is not efficient, according to [7], using such simple merging techniques leads to inaccurate new centroids. If there are two centroids A and B to merge, which have respectively M and N data points in their clusters; with $M > N$, the new centroid will be closer to $A's$ cluster data points. This will affect the performance of the algorithm.

In our previous work (*SPKmMR*) [6], we proposed a parallelization for the Sampling-PSO k-means algorithm (*SPKm*) [8] using MapReduce and shared memory parallelism, and a simple merging technique to obtain the final clusters.

To improve the *SPKmMR* algorithm [6], we propose two different strategies for an efficient final results selection. The first one is based on the K-means algorithm, we obtain the final configuration by clustering the centroids resulted from all the partitions. In the second one, we supposed that involving all centroids from the different partitions may not be the optimal solution, so we proposed to choose the best K centroids among the N centroids using genetic algorithm; where N is the number of all the centroids from the different partitions.

This paper is presented as follows: Section 2 introduces the related works to our contribution. In section 3, our proposed strategies are detailed. Section 4 contains the experimentation and results. Finally, in section 5 we present the conclusion and perspective for future works.

2 Background and Related Work

2.1 K-means

K-means is one of the most used clustering algorithm for its simplicity and efficiency. Several works based on k-means have been proposed. Otherwise its main objective

which is clustering data, K-means has been used in several work to perform many tasks in relation with data clustering. In [9], k-means has been used for the initial centroids selection.

In [8], the authors used k-means algorithm for data sampling, by partitioning data into small sub-sets and applying k-means on each one, the resulted centroids from each partition represent the full data set.

2.2 Genetic Algorithm

Genetic algorithm (GA) is a meta-heuristic used in optimization problems. GA has been used in many clustering tasks. It is known for its efficiency in the initial centroids selection problem for the K-means algorithm. In [10], the authors proposed a solution for initial centroids selection based on GA. The algorithm select the K optimal centroids from the initial DataSet, the resulted centroids are then used as the starting clusters' centers for the K-means algorithm. In [11], an algorithm for clustering data has been proposed based only on the GA.

2.3 Parallel Sampling-PSO-Multi-core-K-means Algorithm using MapReduce (SPKmMR)

The algorithm presented in [6] is a parallelization of the Sampling-PSO-K-means algorithm (SPKm). In that work we proposed to parallelize the *SPKm* algorithm using MapReduce and shared memory parallelism. This algorithm can be summarized in the following steps:

- Initial data is divided on the set of machines.
- Apply *SPKm* on each machine using its data partition to find local clusters.
- Merge the clusters from the previous step to get the final solution.

This method showed efficiency and enhanced the rendering of the *SPKm* algorithm, but we aim to improve this algorithm by improving the merging step.

By analyzing each data partition's centroids, we noticed in some cases that the centroids resulted from a single partition give better results than the ones obtained after the merging process. This proved that the used merging technique failed to obtain an optimal solution and using an inappropriate merging technique provides inaccurate and malformed centroids.

Parallel clustering by partitioning is sensitive to the used merging method. We aim to propose an approach to improve the final selection step of our previous work [6]. We propose two techniques for the final selection process. The first one is based on k-means' work cited above, the use of k-means in initial selection and in sampling algorithms inspired us to use it in the merging step. The second approach is inspired

from the hybrid genetic-k-means algorithm works cited above. In our case, we use the genetic algorithm to select the K best centroids from the set obtained from the different data partitions. The two proposed solutions will be detailed in the next section.

3 Proposed Approach

We present in this section our two contributions in the merging step for the *SPKmMR* algorithm. Our initial objective was to use k-means algorithm to get the final centroids configuration. This proposition involves all the centroids obtained from the different data partitions, as mentioned earlier, using all the centroids obtained from the different machines may not be optimal because of the random data distribution between the machines in the first step. For this reason we propose a solution where only the best K centroids are chosen in the final selection step using the genetic algorithm. Before detailing these two approaches we can summarize all the process in these steps:

1. Distributing randomly initial data input on the set of machines.
2. The algorithm *SPKmMR* is applied on each machine.
3. N local clusters are obtained from the previous step; with $N = K \times M$; where K is the number of clusters and M is the number of machines.
4. One of the proposed merging strategies is applied to obtain the final clusters configuration.

3.1 K-means Algorithm Merging

In this solution, we use the k-means algorithm for selecting the final configuration. In [8], the authors used the k-means algorithm for sampling the initial data and reduce its size. The resulted centroids from the sampling step can represent all the initial data set, so they applied the PSO-K-means algorithm on the results of the sampling step instead of applying it on the entire data set. In [6], we applied *SPKm* on many data partitions, and K centroids were obtained from each one.

We can consider each K centroids obtained from a data partition as the representative data points of this one, so the ensemble of all obtained centroids can represent as well the entire initial data set.

Thus, at the end of *SPKmMR*, we have N centroids; with $N = K \times M$; where K is the chosen number of clusters and M is the number of machines in the cluster. Finally, instead of applying the overlap technique for merging the centroids, we consider the N centroids as a sample for the entire initial data set. So we apply the k-means algorithm on these centroids to get the final K centroids.

3.2 Genetic Algorithm Merging

The second proposed strategy for the final selection step is based on the GA. As mentioned earlier, involving all the centroids may not be the optimal solution. Some centroids can be malformed due to a bad initialization when applying *SPKmMR*, or due to the random data distribution on the machines as proved in [12]. To avoid this, we propose to select among the set of centroids the K best ones. This can be seen as the k-means centroids initialization problem [10]. Therefore, we propose to use the genetic algorithm to select the best K centroids among the ones resulted from each data partition.In this solution, the genetic algorithm takes as input data the centroids set obtained from the executions of the *SPKmMR* algorithm on the different data partitions, the GA takes also as input data the entire initial data set, it is used when evaluating (fitness calculation) the obtained solutions (chromosomes evaluation) of the GA.

Population and Chromosomes. The population of the genetic algorithm is generated randomly from the centroids ensemble obtained from the *SPKmMR* algorithm executed on the different data partitions. N chromosome are generated, and at each iteration the best N ones are selected. Each chromosome from the population is a vector of K random centroids selected from the centroids ensemble. Each chromosome is a potential clusters solution.

Crossover and Mutation Operations. At each iteration of the genetic algorithm, new chromosomes are obtained by applying mutation and crossover operations on the population. The crossover operation combines two randomly selected solutions and generates two new solutions. The mutation operation in our case consists in changing randomly one of the K elements of a given chromosome and replace it by another one from the set of centroids. The best solutions are selected for the next iteration and so on until the last iteration. At the end of the genetic algorithm, the best obtained solution represents the final clusters configuration.

4 Implementation

To analyze our propositions, we implemented our previous algorithm SPKmMR and the two proposed merging strategies. The tests were done on "The Individual household electric power consumption" data set [13], which contains 2075259 data points with 9 dimensions.

Before staring the clustering process, data is preprocessed. This step is very important to obtain good results. In our case, it consists of cleaning data, eliminating the insignificant attributes, normalize the data values and finally each data point is represented with a numerical vector which fits with the MapReduce data structure (*Key/Value pair*).

4.1 Evaluation

To evaluate a solution we should determine how close the objects of a same cluster are. We use the formula (1) to calculate the average distance between a centroid and the data points of its cluster. The smaller this value is the better is the clustering quality. It is defined as follows:

$$f = \frac{\sum_{i=1}^{k} \left\{ \frac{\sum_{j=1}^{n_i} d(o_i, p_{ij})}{n_i} \right\}}{K} \tag{1}$$

with p_{ij} is the j^{th} data point in the i^{th} cluster; o_i is the center of the i^{th} cluster; $d(o_i, p_{ij})$ represents the distance between the data point p_{ij} and the centroid o_i; n_i is the number of data instances in the cluster C_i and K is the defined number of clusters. To calculate the distance between two data points we used the euclidean distance.

For the evaluation of the solutions (chromosomes) generated by the GA, we also used the formula (1), because evaluating a chromosome means to determine how compact is the clusters configuration contained in this chromosome.

5 Experimentation

To evaluate our proposed solutions, we implemented the following algorithms: K-means, Sampling+PSO+Kmeans (*SPKm*), k-means using MapReduce (*KmMR*) and Sampling+PSO+McK-means using MapReduce (*SPKmMR*).

We tested the proposed merging strategies with the parallel algorithms from the list above (*KmMR*, *SPKmMR*).

To evaluate our propositions, we used a cluster of 5 machines. Each node is equipped with a Dual Core CPU and 2 GB of RAM. The cluster runs on Linux Ubuntu 10.04. We used the framework Hadoop 1.2.1 which is an open source implementation of the MapReduce framework. The experiments were done on the normalized data set "The Individual household electric power consumption".

Many algorithms have been implemented to test our approach, and each one has its own parameters. For the Sampling+PSO step, the used parameters are the same used in [6]. The table 1 summarizes the k-means algorithm step and the genetic algorithm (*GA*) parameters.

In this section we discuss the obtained results from the different implemented algorithms. To evaluate the clustering quality of the obtained solutions, we use the formula 1.

Table 2 shows the results of the implemented algorithm on "The Household Electrical Consumption" data set.

Table 1 Algorithms parameters

Number of clusters (K) in K-means	50-100 [7]
Number of iterations in k-means	25
Population size in GA	50
Number of generation in GA	100

Table 2 The performances of algorithms

		K = 50	K = 100
sequential algorithms	k-means	0.079233	0.067021
	SPKm	0.073251	0.063955
Simple merging	KmMR	0.077204	0.064242
	SPKmMR	0.071459	0.063574
K-means merging	KmMR	0.071763	0.061751
	SPKmMR	0.070317	0.059569
GA merging	KmMR	0.057541	0.051913
	SPKmMR	0.054363	0.049195

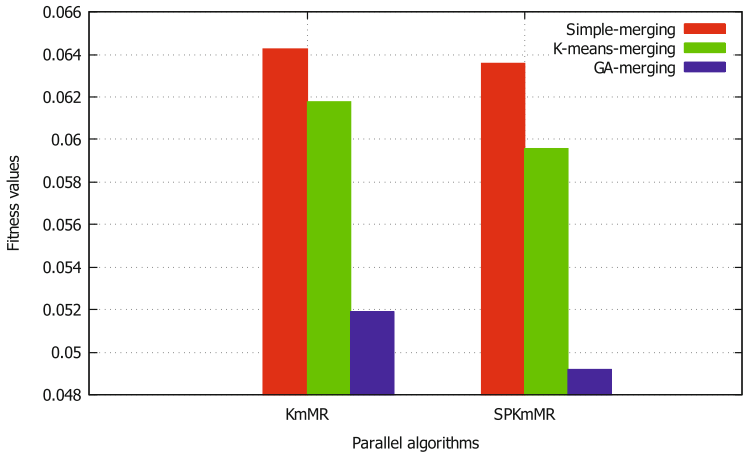
The results show that the proposed merging techniques give better results than the simple merging strategy. The simple merging strategy showed its inability to exploit the centroids obtained from different partition. We can notice this in the case of the *SPKmMR* algorithm where $K = 100$; the fitness value obtained after using a simple merging is equal to 0.063574 whereas the fitness obtained with the sequential algorithm *SPKm* is almost identical to it.

For the other cases of the simple merging strategy, we can see that for each algorithm, the results are slightly better than the ones obtained from those of the sequential versions.

The figure 1 illustrates the obtained results of the implemented parallel algorithms (KmMR and SPKmMR) with the different merging strategies in the case where *K=100*.

We notice that by using the k-means merging strategy we obtained a visible improvement compared to the simple merging strategy on both of the implemented parallel algorithms. We can see also that the GA merging technique gives the best fitness values especially when applied on our previous algorithm (SPKmMR) [6]. Another important point to highlight is that the improvement obtained with the SPKmMR algorithm compared to the KmMR algorithm changes according to the used merging strategy. We can see that the difference in the fitness value between SPKmMR and KmMR is more important when using K-means merging than when using the simple merging strategy. We can notice also that the difference in the fitness value between

Fig. 1 Performances of the merging strategies



SPKmMR and KmMR is the most significant when using the GA merging strategy compared to the two other merging techniques. This is explained by the fact that the GA merging strategy exploits the results generated by the SPKmMR algorithm better than the two other strategies.

6 Conclusion

In this paper, we proposed solutions to improve our work presented in [6]. We proposed to improve the merging step which showed deficiency in some cases. In a first place we proposed to use the K-means algorithm to get the final clusters configuration, next we suggested to use the genetic algorithm for selecting the K optimal centroids and consider them as the final solution. These two methods proved their efficiency by improving the performances of our previous work. In the Map phase of the MapReduce process, data is partitioned randomly on the cluster's machines. As future work, we aim to propose an efficient data partitioning approach to improve the clustering quality of our algorithm.

References

1. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, California, USA, vol. 1, pp. 281–297 (1967)

2. Ene, A., Im, S., Moseley, B.: Fast clustering using mapreduce. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 681–689. ACM (2011)
3. Guerrieri, A., Montresor, A.: Ds-means: distributed data stream clustering. In: Euro-Par 2012 Parallel Processing, pp. 260–271. Springer (2012)
4. Ferreira Cordeiro, R.L., Traina Junior, C., Machado Traina, A.J., López, J., Kang, U., Faloutsos, C.: Clustering very large multi-dimensional datasets with mapreduce. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 690–698. ACM (2011)
5. Mashayekhi, H., Habibi, J., Voulgaris, S., van Steen, M.: Goscan: Decentralized scalable data clustering. Computing **95**(9), 759–784 (2013)
6. Bousbaci, A., Kamel, N.: A parallel sampling-pso-multi-core-k-means algorithm using mapreduce. In: 2014 14th International Conference on Hybrid Intelligent Systems (HIS), pp. 129–134. IEEE (2014)
7. Cui, X., Zhu, P., Yang, X., Li, K., Ji, C.: Optimized big data k-means clustering using mapreduce. The Journal of Supercomputing **70**(3), 1249–1259 (2014)
8. Kamel, N., Ouchen, I., Baali, K.: A sampling-pso-k-means algorithm for document clustering. In: Genetic and Evolutionary Computing, pp. 45–54. Springer (2014)
9. Bradley, P.S., Fayyad, U.M.: Refining initial points for k-means clustering. In: ICML, vol. 98, pp. 91–99. Citeseer (1998)
10. Kwedlo, W., Iwanowicz, P.: Using genetic algorithm for selection of initial cluster centers for the k-means method. In: Artifical Intelligence and Soft Computing, pp. 165–172. Springer (2010)
11. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern recognition **33**(9), 1455–1465 (2000)
12. Hore, P., Hall, L., Goldgof, D.: A cluster ensemble framework for large data sets. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2006, vol. 4, pp. 3342–3347. IEEE (2006)
13. Lichman, M.: UCI Machine Learning Repository (2013)