Data Mining in High-Performance Computing: A Survey of Related Algorithms

Pradip Kumar Majumder and Mohuya Chakraborty

Abstract Present days parallel, distributed or cloud computing technologies have been able to regulate large data sets efficiently. An important part of information technology is extensive data processing. This is because of availability and accelerated surge of data. Data mining is the process of examining large preexisting databases or raw data to generate new information for further use. Data mining algorithms must be efficient and effective in order to produce some meaningful output. The applications for these are limitless, from predicting a specific disease to very complex applications. In this research paper, we have compared most popular data mining algorithms with respect to conceptual architectures and advantages and disadvantages of them. Finally, we have proposed a new efficient data mining algorithm named as sifted K-means with independent K-value (SKIK) algorithm.

Keywords Cloud computing · Distributed/cluster technology
High-performance clusters (HPC) · Data mining algorithms
K-means algorithm

1 Introduction

Cloud computing is a new technology containing pool of resources with large number of computers. The computation task is distributed to this pool and also provides us with unlimited storage and computing power which will benefit us to mine large amount of data.

Tightly coupled systems including shared memory systems (SMS), distributed memory machines (DMM), or clusters of SMS workstations are connected with

P. K. Majumder (✉)
Department of IT, University of Engineering & Management, Kolkata, West Bengal, India
e-mail: majumder.pk@gmail.com

M. Chakraborty
Department of IT, Institute of Engineering & Management, Kolkata, West Bengal, India
e-mail: mohuyacb@iemcal.com

© Springer Nature Singapore Pte Ltd. 2019

M. Chakraborty et al. (eds.), *Proceedings of International Ethical Hacking Conference 2018*, Advances in Intelligent Systems and Computing 811,
https://doi.org/10.1007/978-981-13-1544-2_35

a fast network in a parallel data mining environment. Loosely coupled processing nodes/computers are connected by the high-speed network in a distributed computing environment. Each node contributes to the execution or distribution/replication of data. It is generally called as a cluster of nodes. Usually, a cluster framework is used to set up a cluster.

The HPC clusters exploit parallel computing to exert more computation power for the resolution of a problem. HPC clusters have a large number of computers called "nodes," and mostly these nodes would be configured identically. Externally, the cluster looks like one system. Client programs that run on a node are called jobs, and they are constantly monitored through a queuing technique for proper use of every accessible resource. HPC jobs include replication of numerical models or study of information from logical instrumentation which allow scientists to construct new science at the use of high-performance computing [1].

Data mining is the process of examining large preexisting databases or raw data to generate new information for further use. Data mining algorithms must be efficient and effective in order to produce some meaningful output. Among the numerous available data mining algorithms, the popular ones are Apriori, DIC, GSP, SPADE, SPRINT, and K-means algorithms. In earlier days, due to limitations in computing power, data mining process was slower. Nowadays, the data mining process has speed up many folds due to the presence of high-performance parallel and distributed computing environments. However, the data available today are very large and growing in an exponential rate, which need more effective and accurate data mining algorithm to be deployed. K-means algorithm, despite being one of the most effective algorithms to be used in a parallel computing environment, has some major limitations, and we have worked on proposed SKIK algorithm to overcome one such limitation.

The organization of the paper is as follows: Sect. 1 holds the introduction, and Sect. 2 contains brief discussion of data mining concepts used in HPC environment along with their pros and cons. Section 3 describes the proposed sifted K-means with independent K-value (SKIK) algorithm created based on the advantages and disadvantage of existing algorithms, and Sect. 4 shows the complexity measurement of SKIK algorithm. Section 5 highlights conclusion to this paper with few focus points on imminent works.

2 Data Mining Concepts and Related Algorithms

Different data mining concepts including types of machine, parallelisms, load balance, database layouts, and candidates are discussed in detail in Sect. 2.1. Performance analyses of some of the most popular algorithms vis-a-vis concepts are provided in Table 1. In Sect. 2.2, various related and most popular algorithms are explained briefly. A comparative analysis of these algorithms highlighting their advantages and disadvantages are pointed out in Sect. 2.3.

Table 1 Comparisons among common concepts used with data mining algorithms [2]

Algorithm	Base algorithm	Type of machine	Parallelism type	Load balance type	DB layout	Concepts	DB type
CD	Apriori	DMM	Data	Static	Horizontal	Replicated	Partitioned
PDM	Apriori	DMM	Data	Static	Horizontal	Replicated	Partitioned
FDM	Apriori	DMM	Data	Static	Horizontal	Replicated	Partitioned
IDD	Apriori	DMM	Task	Static	Horizontal	Partitioned	Partitioned
HD	Apriori	DMM	Hybrid	Hybrid	Horizontal	Hybrid	Partitioned
CCPD	Apriori	SMS	Data	Static	Horizontal	Shared	Partitioned
PCCD	Apriori	SMS	Task	Task	Horizontal	Partitioned	Shared
HPA	Apriori	DMM	Task	Static	Horizontal	Partitioned	Partially replicated
APM	DIC	SMS	Task	Static	Horizontal	Shared	Partitioned
HPSPM	GSP	DMM	Task	Static	Horizontal	Partitioned	Partially replicated
SPADE	SPADE	SMS	Task	Dynamic	Vertical	Partitioned	Shared
D-MSDD	MSDD	DMM	Task	Static	Horizontal	Partitioned	Replicated
SPRINT	SPRINT	DMM	Data	Static	Vertical	Replicated	Partitioned
PDT	C4.5	DMM	Data	Static	Horizontal	Replicated	Partitioned
MWK	SPRINT	SMS	Data	Dynamic	Vertical	Shared	Shared
SUBTREE	SPRINT	SMS	Hybrid	Dynamic	Vertical	Partitioned	Partitioned
HTF	SPRINT	DMM	Hybrid	Dynamic	Vertical	Partitioned	Partitioned
P-Cluster	K-means	DMM	Data	Static	Horizontal	Replicated	Partitioned

2.1 Concepts

Type of machine used. The main two types of machines are distributed memory machines (DMM) and shared memory systems (SMS). In DMM, the effort is communication optimization and hence synchronization is implicit in message passing. For SMS, synchronization occurs via locks and barriers, and the aim is to minimize these points. Data decomposition is very important for DMM, but not for SMS. SMS typically use serial I/O, while DMM use parallel I/O [2].

Parallelism type. Task and data parallelism are two major parallelisms used. In data parallelism, the database is partitioned among P processors. Each processor performs evaluating candidate patterns/models on its local part of the database. In task parallelism, the processors perform different computations independently, but have/need access to the entire database. SMS can connect to whole data, but for DMM can do this through careful reproduction or specific connection to the local data. It is also possible for a hybrid parallelism having properties of both task and data parallelisms.

Load balance type. Two main load balancing types are static and dynamic load balancing. In static load balancing, work is partitioned among the processors using heuristic cost function, and there is no subsequent correction of load imbalances resulting from the dynamic nature of mining algorithms. Dynamic load balancing distributes work from heavily loaded processors to lightly loaded ones. Dynamic load balancing is important in multi-user environments and in heterogeneous platforms, which have different processor and network speeds.

Database layout type. Usually, the recommended database for data mining is a relational table having R rows, called records, and C columns, called attributes. Horizontal database design is used in numerous data mining algorithms. Here, they collect transaction id (tid) as a unit including attribute values for that transaction. Other procedures use a vertical database design. Here, they collect a list of all tids (called tidlist) containing the item with each attribute and the related attribute value.

Candidate concepts. Dissimilar mining procedures use either shared or replicated or partitioned candidate concept generation and evaluation. All processors check out a single copy of the candidate set in shared concept. The candidate concepts are copied on each system, and checked locally, before overall outcomes are achieved by fusing them in replicated concept. Each processor creates and examines a dislocated candidate set in the partitioned concept.

Database type. The database itself can be shared (in SMS or shared-disk architectures), partitioned (using round robin, hash, or range scheduling) among the available nodes (in DMM) or partially or totally replicated.

Table 1 shows a comparison among the common concepts used with most popular data mining algorithms.

2.2 Most Popular Algorithms

Apriori Algorithm. This algorithm is used for mining common itemsets in large data sets. The point of view is "bottom up." We call it candidate generation, where frequent subsets are available one item at a time, and groups of candidates are checked against the data. It is aimed to operate on transaction database.

Frequent Itemsets: All the sets holding the item with the least support (designated by D_i for ith itemset).

Apriori Property: All subgroups of frequent itemset have to be frequent.

Join Operation: A set of candidate k-itemsets are generated by joining D_{k-1} with itself to find D_k.

Prune Step: Any sparse $(k-1)$-itemset cannot be a subset of a frequent k-itemset.

C_k: Candidate itemset of size k
D_k: frequent itemset of size k
$D_1 = \{\text{frequent items}\}$;

STEP 1: Have the support S of each 1-itemset by examining the given database, correlate S with $\sup_{\min}$, and prepare a support of 1-itemsets, D_1

STEP 2: Use D_{k-1} and join D_{k-1} to create a set of candidate k-itemsets. And use Apriori property to prune the infrequent k-itemsets from this set.
STEP 3: Examine the given database to find the support S of each candidate k-itemset in the find set, correlate S with sup_{min}, and prepare a set of frequent k-itemsets D_k
STEP 4: Is the candidate set = Null, if YES go to STEP 5 else go to STEP 2
STEP 5: Produce all nonempty subsets of 1 for every common itemset 1,
STEP 6: If confidence C of the rule "s => (1 − s)" (=support of 1/support S of s)' min_conf, output the rule "s => (1 − s)", for every nonempty subset s of 1 [3].

Dynamic Itemset Counting Algorithm (DIC). It is an alternative to Apriori algorithm. As the transactions are read, the itemsets are dynamically inserted and removed. Assumptions are made that all subgroups of frequent itemset have to be frequent. After every T transactions, algorithm stops to add more itemsets. Itemsets are tagged in four different ways as they are counted:

Solid box: ☐ confirmed frequent itemset—an itemset we have completed counting and exceeds the support threshold sup_{min}

Solid circle: ◯ confirmed infrequent itemset—we have completed counting and it is below sup_{min}.

Dashed box: ⬚ imagined frequent itemset—an itemset being counted that surpasses sup_{min}

Dashed circle: ◌ imagined uncommon itemset—an itemset being counted that is below sup_{min}

STEP 1: Tag the empty itemset with a solid square. Tag the 1-itemsets with dashed circles. Discard all other itemsets untagged.
STEP 2: While any dashed itemsets remain.

1. Read *M* transactions (if at the end of the transaction file, continue from the beginning). For each transaction, step up the corresponding counters for the itemsets that appear in the transaction and are tagged with dashes.
2. If a dashed circle's count surpasses sup_{min}, make it a dashed square. Insert a new counter for it and make it a dashed circle if any next superset of it has all subsets as solid or dashed squares.
3. If a dashed itemset has already been counted through all the transactions, make it solid and stop counting it [4].

Generalized Sequential Pattern Algorithm (GSP). A sequence database is formed of ordered elements or events. In GSP algorithm, horizontal data format is used and the candidates are generated and pruned from frequent sequences using Apriori algorithm.

STEP 1: Each item in database is a candidate of magnitude 1 at the beginning.
STEP 2: for each level (i.e., order of magnitude k) do

1. Examine database to gather support count for every candidate order.
2. Generate candidate magnitude (k + 1) orders from magnitude k frequent orders using Apriori.

STEP 3: Do it over till no common order or no candidate can be found [5].

Sequential PAttern Discovery using Equivalence classes (SPADE). It is an algorithm to frequent sequence mining using vertical ID list database format, where each sequence is related with a list of objects in which it appears. Then, frequent sequences can be found surely using intersections on ID lists. The procedure lowers the number of database scans and hence also lowers the execution time.

STEP 1: Sequences having singular item, in a single database scan, are measured as the number of 1-sequences.
STEP 2: Convert the vertical depiction into horizontal depiction in memory and measure the number of sequences for each pair of items using a two-dimensional matrix for 2-sequences calculation. Thus, this step can also be performed in only one scan.
STEP 3: Following n-sequences can then be formed by joining $(n - 1)$-sequences using their ID lists. The size of the ID lists is the count of sequences in which an item occurs. If this number is higher than minsup, the sequence is a frequent one.
STEP 4: If no frequent sequences available, the algorithm stops.

The algorithm can use a breadth-first or a depth-first search procedure to discover new sequences [6].

Scalable PaRallelizable INduction of decision Trees (SPRINT) Algorithm. This algorithm builds a model of the classifying characteristics based upon the other attributes. Classifications provided are called a training set of records having several attributes. Attributes are either continuous or categorical.

SPRINT algorithm frees all of memory restrictions in contrast with SLIQ algorithm. This is also fast and scalable and can be easily parallelized.

Original SPRINT algorithm has the following steps:

```
Division(Data D)
if (every point in D are of same class) then
    Return;
for every attribute A do
    calculate splits on attribute A;
Use best split found to divide D into D1 and D2;
Division(D1);
Division(D2);
```

Original Call: Division(Training Dataset)
Prune step is done using SLIQ algorithm [7].

K-means Clustering Algorithm. K-means is an unsupervised learning algorithm classifies a given data set through a certain number of clusters (assume k clusters) fixed a priori. Different cluster positions will cause different outcomes. So we must define k centers, one per cluster which should be placed in a clever manner. So, placing them as far as possible from each other seems a better choice. This algorithm tries to minimize “squared error function” given by:

$$J(X) = \sum_{i=1\rightarrow S} \sum_{j=1\rightarrow Si} \left(||w_i - x_j||\right)^2$$

where

"$||w_i - x_j||$" is the Euclidean distance between w_i and x_j.
"s_i" is the number of data points in ith cluster.
"s" is the number of cluster centers

Let $W = \{w_1, w_2, \ldots, w_n\}$ be the set of data points and $X = \{x_1, x_2, \ldots, x_s\}$ be the set of centers.

STEP 1: Randomly select "s" cluster centers.
STEP 2: Calculate the distance between each data point and cluster centers.
STEP 3: Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
STEP 4: Recalculate the new cluster center using:

$$x_i = (1/s_i) \sum_{j=1\rightarrow Si} w_j$$

where "s_i" means the data points number in ith cluster.

STEP 5: Recalculate the distance between each data point and new obtained cluster centers.
STEP 6: If no data point was reassigned, then stop, otherwise repeat from STEP 3 [8, 9].

2.3 Advantages and Disadvantages of the Algorithms

Table 2 shows a comparative study about pros and cons of the above-mentioned algorithms.

3 Proposed SKIK Algorithm

K-means algorithm generates K clusters of the known data set, where every cluster can be expressed by a centroid which is a concise expression of all the objects present in a cluster. The main flaws of K-means algorithm are: (i) it is difficult to anticipate the number of clusters (value of K) and (ii) initial centroids have a big effect on the concluding outcome. Here, we are introducing a new algorithm sifted K-means with independent K-value (SKIK) algorithm to overcome these issues.

In data mining, we work on very large data set. We have proposed to sort these data based on any attribute as per user requirement. We would use parallel heap sort

Table 2 Advantages and disadvantages of popular algorithms

Algorithm	Advantages	Disadvantages
Apriori [10]	1. It is easy to apply and easy to figure out 2. It can be used on large itemsets	1. Sometimes, a large number of candidate rules are required which can be costly to compute 2. It goes through entire DB, hence calculating support is costly
DIC [11]	1. If the data are similar throughout the file and the interval is fairly small, it normally makes on the order of two passes 2. It can add and delete itemsets on the fly and thus extended to parallel and incremental versions	1. It is very delicate to dissimilar data 2. If the date are very associated, DIC counts most of the DB and itemset is realized to be actually large 3. It has performance issues as to how to increment the relevant counters for a specific transaction
GSP [12, 13]	1. We can enter bounds on the time separation between adjoining items in an arrangement 2. The items present in the pattern element can stretch a transaction set within a time frame specified by user 3. Detection of frequent patterns in various levels as needed by user is possible. Detecting generalized sequential patterns is also possible	1. Repeated DB scanning to compute the support of candidate patterns is costly for a large database 2. It may create patterns that do not exist in the DB as it generates candidates by linking smaller patterns without accessing the DB 3. All frequent sequences of length k is kept in memory to create patterns of length $k+1$ as it is BFS pattern mining algorithm. It consumes a good memory
SPADE [14, 15]	1. It uses vertical DB layout 2. The search space is expressed as lattice formation and breaks up original lattice into sub-lattices to process using either BFS or DFS 3. Efficient support counting method based on the idlist structure is used and is nearly twice faster than GSP algorithm. It shows linear scalability w.r.t. the number of sequences	1. A huge set of candidates generated, specially 2-item candidate sequence 2. Multiple scans of database in mining. magnitude of every candidate increases by one at every database scan 3. Inefficient for mining long sequential patterns. A long pattern causes an exponential number of short candidates
SPRINT [16, 17]	1. It removes memory constraints that limit existing decision tree algorithms 2. It consciously averts the need for any centralized, memory resident data structure 3. It allows analysis of nearly any sized data set, and it is fast 4. It is easily parallelized which needs few inclusions to serial algorithm	1. High workload in precisely locating the optimal splitting point 2. The main time expense is used to sort all records of the attribute table in the entire process
K-means [18]	1. Easy to deploy with a massive number of variables, and it is faster than hierarchical clustering (if K is small) 2. K-means can produce tighter clusters than hierarchical clustering 3. An instance can change cluster when the centroids are recalculated	1. Difficult to predict the number of clusters (K-value) 2. Initial seeds have a strong impact on the final results 3. Rescaling the data sets (normalization or standardization) will completely change results

[19] to sort as it uses a parallel approach across the cluster utilizing the available architecture.

Steps to find initial centroids:

1. From n objects, determine a point by arithmetic mean. This is the first initial centroid.
2. From n objects, decide next centroids so that the Euclidean distance of that object is highest from other decided original centroids. Keep a count of the centroids.
3. Repeat Step 2 until $n \leq 3$ [20].

We will get initial centroids from here and can use them in the proposed algorithm to calculate "optimal" centroids and K-value.

Determination of K:

The K-means algorithm creates compact clusters to minimize the sum of squared distances from all points to their cluster centers. We can thus use the distances of the points from their cluster center to measure if the clusters are compact. Thus, we adopt the inner-cluster distance, which is usually the distance between a point and its cluster center. We will take the median of all of these distances, described as

$$D_{wc} = (1/N) \sum_{i=1 \to k} \sum_{w \varepsilon Si} ||w - c_i||^2$$

where N is the count of components in the data set, k is the count of original clusters equal to the number of originally determined centroids, and c_i is the center of cluster s_i.

We can also measure the between-cluster distance. We take the minimum of the distance between cluster centers, defined as

$$D_{bc} = \min\left(||c_i - c_j||^2\right), \text{where } i = 1, 2, \ldots, k-1 \text{ and } j = i+1, \ldots, k.$$

Now genuineness, $G = D_{wc}/D_{bc}$.

We need to decrease the inner-cluster distance, and this measure is in the numerator. So we need to decrease the genuineness measure. The between-cluster distance measure needs to be increased. Being in the denominator, we need to decrease the genuineness measure. Hence, clustering with a "lowest value" for the genuineness measure will provide us the "optimal value" of K for the K-means procedure [21].

We can also evaluate the "optimal" K using both inner-cluster and between-cluster scatter using the method proposed by Kim and Park [22].

Steps of SKIK:

1. Start.
2. Load the data set.
3. Sort the data using parallel heap sort.
4. Find initial centroids using previously mentioned procedure.
5. Determine K (number of clusters) from the centroids.
6. Calculate the distance between each data point and cluster centers.

7. Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
8. Recalculate the new cluster center using:

$$x_i = (1/s_i) \sum_{j=1 \rightarrow Si} w_j$$

where "s_i" means the data points number in *i*th cluster.

9. Recalculate the distance between each data point and new obtained cluster centers.
10. If no data point was reassigned, then stop, otherwise repeat from Step 7.

4 Complexity Measurement of SKIK

Sorting may imply initial workload, but once done it will decrease computation time in many folds.

Time complexity of sorting at Step 3 with n data elements [19]

$$O(n\log n).$$

For the Step 4 to find initial centroids, time complexity for segregating the n data items into k parts and deciding the mean of each part is O(n). Thus, the total time complexity for discovering the initial centroids of a data set containing n elements and m attributes (where m is way less than n) is

$$O(n\log n).$$

Step 5 is again a partitioning procedure having complexity [22]

$$O(n\log n).$$

Steps 6–10 are same as the original K-means algorithm and hence take time

$$O(nKR).$$

where n is the number of data points, K is the number of clusters, and R is the number of iterations. The algorithm converges in very less number of iterations as the initial centroids are calculated in a clever method in harmony with the data dispersion.

So, the general complexity of SKIK is the maximum of

$$\{O(n\log n) + O(n\log n) + O(n\log n) + O(nKR)\}$$
$$\text{i.e., } O(n\log n + nKR) \quad .$$
$$\text{i.e., } O(n(\log n + KR)$$

5 Conclusion

This paper has provided a detailed comparison among six most popular data mining algorithms which have significant contribution in high-performance cluster computation and artificial intelligence. The algorithms are Apriori, DIC, GSP, SPADE, SPRINT, and K-means. The paper presents short algorithmic steps about the main algorithms, explanation of their features, and respective advantages and disadvantages. Several variations of the algorithms exist, and they have been proved to be suitable based on certain scenarios. In the present days, research has been progressing with the most effective data mining algorithms, applicable with parallel and high-performance cloud computing, like SPRINT and K-means. We have proposed SKIK algorithm to improve K-means algorithm to be used with large data set and HPC architecture. We have shown the measurement of complexity of SKIK as well. In-depth research work needs to be conducted for extending the capabilities and complete performance analysis of the SKIK algorithm with respect to other available variations of K-means algorithm.

References

1. Arefin, A.: Introduction to High Performance Computing (HPC) Clusters. https://learn.scientificprogramming.io/introduction-to-high-performance-computing-hpc-clusters-9189e9daba5a
2. Zaki, M.: Parallel and Distributed Data Mining: An Introduction. LNCS, vol. 1759, p. 3 (2002)
3. INSOFE: Apriori Algorithm (2014). https://www.slideshare.net/INSOFE/apriori-algorithm-36054672
4. Brin, M., Ullman, T.: Dynamic itemset counting and implication rules for market basket data. SIGMOD Rec. **6**(2), 255–264 (1997)
5. Mining Sequential Patterns. https://www.slideshare.net/Krish_ver2/53-mining-sequential-patterns?next_slideshow=1, slide 7
6. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. In: Machine Learning, vol. 42, pp. 31–60 (2001)
7. Wang, Z., et al.: A searching method of candidate segmentation point in SPRINT classification. J. ECE **2016**, Article ID 2168478 (2016)
8. k-means clustering algorithm. https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm
9. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7) (2002)
10. Jain, R.: A beginner's tutorial on the apriori algorithm in data mining with R implementation. http://blog.hackerearth.com/beginners-tutorial-apriori-algorithm-data-mining-r-implementation

11. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. SIGMOD Rec. **6**(2), 255–264 (1997)
12. Grover, N.: Comparative study of various sequential pattern mining algorithms. Int. J. Comput. Appl. (0975–8887) **90**(17) (2014)
13. Fournier-Viger, P., et al.: A survey of sequential pattern mining. Ubiquitous Int. **1**(1), 6 (2017)
14. Slimani, T., Lazzez, A.: Sequential mining: patterns and algorithms analysis. IJCER **2**(5), 4 (2013)
15. Georgia Institute of Technology: Sequential Pattern Mining. https://www.cc.gatech.edu/~hic/CS7616/pdf/lecture13.pdf
16. Shafer, J., Agrawal, R., Mehta, M.: SPRINT: a scalable parallel classifier for data mining. In: Proceedings of the 22nd VLDB Conference, Mumbai, India, pp. 544–555 (1996)
17. Ding, Y., Zheng, Z., Ma, R.: Improved SPRINT algorithm and its application in the physical data analysis. TELKOMNIKA Indones. J. Electr. Eng. **12**(9), 6909–6920 (2014)
18. Santini, M.: Advantages & Disadvantages of k-Means and Hierarchical Clustering (Unsupervised Learning), ML4LT, p. 3. Department of Linguistics and Philology, Uppsala University (2016)
19. Zhenhua, W., Zhifeng, L., Guoliang, L.: Parallel optimization strategy of heap sort algorithm under multi-core environment. In: 7th ICMTMA, June 2015. https://doi.org/10.1109/icmtma.2015.190
20. Baswade, M., Nalwade, P.S.: Selection of initial centroids for k-means algorithm. IJCSMC **2**(7), 161–164 (2013)
21. Ray, S., Turi, R.H.: Determination of number of clusters in K-means clustering and application in colour image segmentation. In: The 4th International Conference on Advances in Pattern Recognition and Digital Techniques, pp. 137–143 (1999)
22. Kim, D.J., Park, Y.W.: A novel validity index for determination of the optimal number of clusters. IEICE Trans. Inf. **E84-D**(2), 281–285 (2001)
23. Ortega, J.P., Rocio, M.D., Rojas, B., Garcia, M.J.S.: Research issues on K-means algorithm: an experimental trial using Matlab. In: CEUR Workshop Proceedings, vol. 534 (2009)
24. Tan, S., Ghosh, K.: The k-Means Algorithm—Notes
25. Qin, J., Fu, W., Gao, H., Zheng, W.X.: Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory. IEEE Trans. Cybern. **47**(3), 772–783 (2017)
26. Xu, Y.H., et al: Implementation of the K-Means Algorithm on Heterogeneous Devices: A Use Case Based on an Industrial Dataset. Advances in Parallel Computing, vol. 32: Parallel Computing is Everywhere, pp. 642–651. IOS Press (2018)
27. Qi, H., Di, X., Li, J., Ma, H.: Improved K-means algorithm and its application to vehicle steering identification. In: Advanced Hybrid Information Processing, pp. 378–386. Springer International Publishing (2018)
28. Goel, L., Jain, N., Srivastava, S.: A novel PSO based algorithm to find initial seeds for the k-means clustering algorithm. In: Proceedings of the Communication and Computing Systems ICCCS 2016, Gurgaon, India, p. 159
29. Bai, L., Cheng, X., Liang, J., Shen, H., Guo, Y.: Fast density clustering strategies based on the k-means algorithm. Pattern Recognit. **71**, 375–386 (2017)