Parallel Processing Of Data Mining Functions Using Clustering, Optimization And Classification Techniques

S. Starlin Jini, and Dr. N. Chenthalir Indra

Abstract: Data mining is the process of extracting information from huge sets of data. Data mining is also known as Knowledge Discovery in Database (KDD). Data mining is highly useful in the domains such as market analysis & management, corporate analysis & risk management, and fraud detection. Data mining functions or algorithms are used to perform the specific process or task. In recent decade parallel processing of these data mining functions are encouraged in many research. The parallel processing of data mining functions can help to reduce the execution time. Thus the parallelization of data mining function can reduce the computational complexity and execution time, while handling large data set. In this paper the most used five parallel data mining functions and its performance is analyzed. The functions or algorithms considered in this performance analysis are Parallel K-Means (PKM) for clustering, Parallel Genetic Algorithm (PGA) for optimization, Graph-based Substructure pattern (gSpan) for parallel graph mining, Parallel Apriori (PA) for frequent itemset mining, Parallel Bayesian Network (PBN) for classification. These five techniques are implemented in common platform and its performance are evaluated based on execution time.

Index Terms: Minimum parallel data mining function, parallel optimization, parallel classification, parallel frequent itemset, parallel clustering, parallel graph mining..

1 INTRODUCTION

In today's world data mining is very important because huge amount of data is present in different types of organizations. It is not easily possible for humans to extract information from this large data. So data mining technology is used to process the data and fast enough to extract information from it. Data mining is also called as knowledge discovery, knowledge extraction, data/pattern analysis, information harvesting, etc. It has many applications in various fields for the purpose of information retrieval, searching, and grouping. Data mining functions or algorithms are used to perform the specific process or task [64-66]. Data mining systems face a lot of challenges and issues in today's world. Some of them are, mining methodology, user interaction issues, performance issues and Issues relating to the diversity of database types. Among them performance issue due to the computational complexity is one of the most noticeable issues in the execution of data mining function. In recent time the computational complexity is avoided by employing parallel processing. In parallel processing a work or task is segmented into small and which is executed separately using different processor to speed up the process. The parallel execution can reduce the computational complexity [67-70]. In data mining, the functions or algorithms are divided into partitions or sub function and the data are separately feed to the function to perform particular operation [71, 72]. The data mining function parallelization is one of the recent research area, in which the function or algorithm as well as the input data are processed in parallel [73]. The parallel processing of data mining function can provide better performance in terms of both time and complexity.

- *S. Starlin Jini is currently Research Scholar in the Department of Computer science, M.S. University, Tirunelveli. E-mail: starlinjini@gmail.com*
- *Dr. N. Chenthalir Indra is currently Research Supervisor, M.S University and Assistant Professor, Department of Computer Science in S.T. Hindu College, Nagercoil. E-mail: chenthalirindra@gmail.com*

In literature, the most used data mining function for parallel processing are clustering/Segmentation, classification, optimization, association rule mining, and graph mining. In this paper, some of the research related to the parallel processing of data mining, since 2014 is listed. The detail about the literature survey summary is given in table 1. These literatures are categorized under five functions such as, classification, optimization, clustering, association rule mining, and graph mining. Then a simple algorithm from each category is implemented in a common platform using same database to analysis their performance in terms of execution time. The rest of this paper is organized as follows; in section 2 the simple classification function and its review is given. In section 3 the parallel processing of clustering function and its review is given. Parallel processing of Graph mining function and review is provided in section 4. In section 5 the optimization function and its review is given. In section 6 the frequent itemset mining (Association Rule Mining) function and its review is given.

2 REVIEW ON CLASSIFICATION FUNCTION

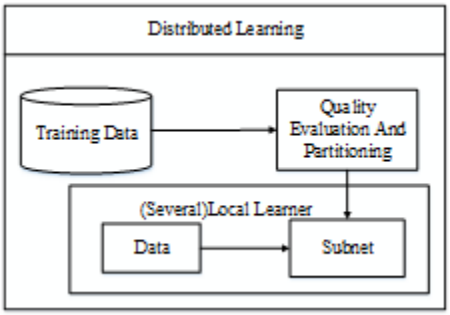
Classification is one of the major data mining functions that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. The classification model, by using training data set predicts the value of classifying attribute or class label. Classification can be performed on structured or unstructured data. The two important steps of classification are;

Model construction: A predefine class label is assigned to every sample object. These object data are known as training data set. The classification model based on the training system is characterized by classification rules, conventional trees or mathematical formulas.

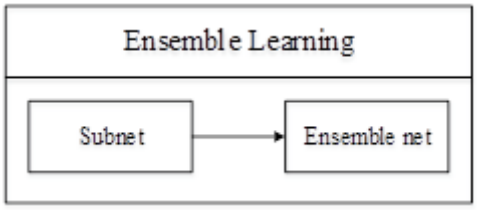
Model usage: The constructed model is used to perform classification of unknown objects. A class label of test sample is compared with the resultant class label. Test sample data and training data sample are always different.

2.1 Parallel processing of classification function

In this section we describe the parallel processing of BN classifier function [62]. A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG). Bayesian Network Learning Approaches can be divided into two sections. The first category is based on domain experience, constructing network structure artificially. The other way is based on current data, learning structure from calculation automatically. Intelligent distributed computing model based on Bayesian network learning approach named Scalable Bayesian Network Learning (SBNL) workflow. SBNL is a two-phase algorithm. Phase I performs the distributed learning and phase II is performs the ensemble learning. In phase I the Training data set D is segmented into n blocks$\{D_1, D_2, \dots D_n\}$, which are used as input data sets. Phase II completes the ensemble learning with subnets$\{N_1, N_2, \dots N_n\}$. Subnets were transferred into an ensemble network via a weighted matrix.



a) Distributed learning



b) Ensemble Learning

Figure 1: *MMHC - Bayesian network learning algorithm*

The procedure of learning Bayesian network structure automatically by calculation can be descripted as this. Given training Data set D and priori knowledge K, identifying the best network structure that can model probabilistic relationships best. i.e., maximizing the post probability $P\ (N\ |D,k)$

$$N = argmax_N\ P\ (N\ |D,k) \quad (1)$$

Several approaches were proposed to leverage time and space cost of BN learning, including K2 algorithm and hill-climbing algorithm. This method decide learning BN structure with MMHC algorithm. This algorithm is shown to have superior learning accuracy and robustness. Current MMHC model is not be reliable size can be measured to a higher dimension and a larger scale Data size. One main reason is the calculation times will intensively increase, causing model being less likely converge in a reasonable time.

2.2 Review based on the classification

Some of the literature based on the parallel processing of the classification function is listed as follows; Kun Yue et al. [18] have presented the parallel and incremental approach for data-intensive learning of BNs. by extending the original MDL-based scoring and search algorithm using MapReduce. In this method the theoretical and experimental result were scalable and more effective. Chunti Shen et al. [22] have introduced the parallelized text classification algorithm for processing large scale TCM clinical data with MapReduce. To improve the efficiency of data processing, each training data was independent and executed in parallel. Bo Zhao et al. [52] have presented the ZenLDA algorithm. To perform the efficient and scalable LDA training on the distributed data-parallel platform. The experimental results demonstrated that ZenLDA can achieve comparable and even better computing performance with the state-of-the-art dedicated systems. Debora G. Reis et al. [53] have presented an approach, which had two-phase parallel learning to schema matching based on metadata. Each phase represented a level of relational metadata aggregation. Hao Peng et al. [59] have calculated the several Bayesian classifiers implemented in SCALATION and demonstrated the effects of parallelization on those classifiers. The algorithms were implemented using Scala and compared with the bnlearn library in R and Weka.

3 REVIEW ON CLUSTERING FUNCTION

Clustering is the process of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. Clustering analysis is broadly used in many applications such as market research, pattern recognition, data analysis, and image processing. Partitioning method: Initially having a database of ‘n’ objects and the partitioning method constructs ‘k’ partition of data. Each partition represents a cluster and k ≤ n. It means that it classifies the data into k groups. It satisfies the following requirements.

- Each group contains at least one object.
- Each object must belong to exactly one group.

Hierarchical method: Hierarchical method creates a hierarchical decomposition of the given set of data objects. There are two approaches here one is agglomerative approach and other is divisive approach. In Agglomerative approach, each object forming a separate group. This approach is also known as the bottom-up approach. In Divisive approach all the objects in the same cluster. This approach is also known as the top-down approach.

Density-based method: This method is based on the concept of density

Grid-based method: In this method objects together from a grid. The object space is quantized into finite number of cells that form a grid structure. The major advantage of this method is fast processing time.

Model-based methods: In this method a model is hypothesized for each cluster to find the best fit of data for a given model. This method locates the clusters by clustering the density function. It reflects spatial distribution of the data points.

Constraint-based method: In this method, the clustering is performed by the incorporation of user or application-oriented constraints. A constraint refers to the user expectation or the

properties of desired clustering results.

3.1 Parallel processing of Clustering Function

In this section we describe the parallel processing of K-Means clustering [31]. K-Means is a non-supervised machine-learning mechanism widely used for signal processing, image gallery and data mining. K-means clustering aims to partition n observations into K clusters. Each observation is allocated to the cluster with the nearest mean, with the mean value of a cluster serving as a prototype for each cluster. K-Means Clustering is one of the most popular centred-based clustering mechanisms due to its simplicity. K-Means clustering on three different platforms such as OpenMP, MPI and CUDA.

The three major contributions of this technique are:

- K-Means implementation that converges based on Data set and user input.
- Comparison of different styles of parallelism using different platforms for K-Means implementation.
- Speed-up the algorithm by parallel initialization.

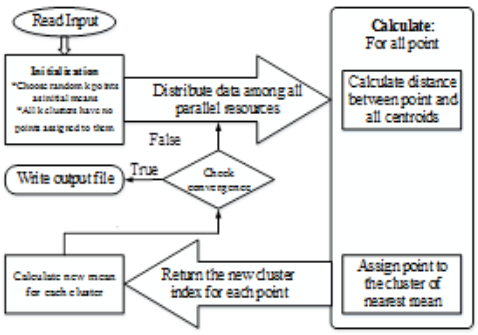


Figure 2: *Simple Classification Model*

1 **Input:** Dataset, K, Tolerance
2 **Output:** Mean table, Assignment of each data point to a cluster
3 *Initialize*
4 Assign data to nearest cluster
5 Calculate new means
6 *if* Converged *then*
7 Output assignment of data
8 **return;**
9 *else*
10 Continue from step 4 with new means

Figure 3: *K-Means clustering algorithm*

K-Means is an iterative method. For all the steps calculation of the distance of each data point to all the means is independent of other data points. The algorithm consists of three stages initialization, computation and convergence. The algorithm spends 85% to 90% of its execution time in computing. K-Means clustering algorithm is below fig 3.

3.2 Review based on the Clustering

Some of the literature based on the parallel processing of the clustering function is listed as follows; Fabio Fumarola et al. [7] have proposed the Distributed Dynamic Clustering Algorithm (D2CA) to deal with spatial Data set s. The D2CA approach was based on the K-means Algorithm. The overall process was designed to be efficient at the time and memory allocation. M. Omair Shafiq et al. [8] have presented the parallel K-Medoids clustering algorithm and it was based on MapReduce paradigm to perform clustering on large-scale of data. Map-Reduce was performed for parallel computation process. Gregory Buehrer et al. [11] have presented the Localized approximate miner (LAM) that scales linearithmically with the input data. Described the parameter free algorithm for determining the interesting patterns in very large data. Ahmed Shabib et al. [16] have described the two methods of parallelizing the DTW algorithm, such as multi core CPU and Apache Spark Cluster. The UCR DTW algorithm was developed in a single CPU core. Kemilly Dearo Garcia et al. [19] have presented the MapReduce clustering algorithm to execute multiple parallel runs of k-means with different initializations and number of clusters. The algorithm was experimentally compared with the Apache Mahout Project's MapReduce implementation of k-means. Janki Bhimani et al. [31] have presented the K-Means clustering implementation, which tackles the three challenges of K-Means algorithm. They were shared memory using Open MP, distributed memory with message passing (MPI), and heterogeneous computing with NVIDIA Graphics Processing Units (GPUs) programmed with CUDA-C. Ilias K. Savvas et al. [38] have presented the three-phase parallel version of DBSCAN. DBSCAN algorithm was a density-based spatial clustering technique. DBSCAN was executed in three consecutive phases. Eftim Zdravevski et al. [40] have proposed the parallel implementation of clustering technique. It was based on writing Pig Latin scripts that were compiled into MapReduce, which was executed on the Hadoop clusters. During the experimentation the speed of the parallelization was compared with one-node cluster. Mohammed Alandoli et al. [41] have presented the implementation of the clustering-based community detection algorithm. They utilized the GPUs to speed-up the performance of community detection in Social Networks. Xiaoming Gao et al. [42] have focused the parallel stream analysis module of Cloud DIKW and presented the work in applying it to support one representative application on clustering of social media data streams. Md.Mostofa Ali Patwary et al. [45] have investigated PARDICLE an approximate algorithm for the density-based clustering algorithm called DBSCAN. A parallel DBSCAN algorithm was proposed to improve load balancing and locality. Son T. Mai et al. [51] have proposed the unique approach for accelerating the structural graph clustering algorithm called as anySCAN. The anySCAN was the first anytime and parallel structural graph clustering algorithm. Cong JI et al. [60] have proposed the parallel K-means clustering algorithm based on CUDA (compute uniform device architecture).

4. REVIEW ON GRAPH MINING FUNCTION

Graph mining has its applications in various domains including bioinformatics, chemical reactions, program flow structures, computer networks, social network, etc. The task of graph mining is to extract patters (sub-graphs) of interest from graphs, which describe basic data and can be used more. e.g., for classification or clustering. Many sub-graphical mining has multiple map mining tools, which are used for large database. Graph mining has a vast number of applications in biological networks, Cheminformatics and web data.

4.1 Parallel processing of Graph Mining Function

This section describes the frequent subgraph mining problem

proposed in [12]. The basic algorithmic steps of the gSpan algorithm are presented. The most known graph pattern mining problem is the frequent subgraph mining (FSM). It focuses on the number of the frequent subgraphs, either in a single graph or in a graph database. The frequent subgraph mining methods use two different algorithmic approaches, such as the apriori-based approach and the pattern growth approach. The apriori-based algorithms start with small-sized graphs and extend them by joining subgraphs. On the other hand, the algorithms that use the pattern growth approach like gSpan extend a frequent graph by adding a new edge in every possible position. The subgraph isomorphism is the most time-consuming part of the algorithm. Two solutions have been proposed for this problem. They are embedding lists and canonical labelling. The first one keeps a list of all the occurrences of a certain label in the database. These lists keep all the appearances of the labels of specific graph structures in the database. On the other hand, the canonical labelling approach uniquely presents the structure of a graph so the repetition of information is minimal. The gSpan algorithm is one of the most efficient frequent subgraph mining algorithms. It maps each graph into a minimum Depth First Search (DFS) code based on lexicographic ordering. In that way the duplication of the graphs while minimizing the search tree. Subgraph mining procedure are consisting of three steps. First the frequency of the processed subgraph in the input Data set is calculated. Second, the subgraph's isomorphism test takes place. The last step is the recursive examination for any possible extension of the current subgraph.

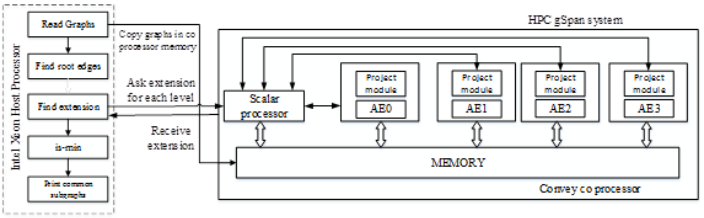


Figure 4: *HPC-gSpan System Architecture*

HPC-gSpan System Architecture given in fig 4, which consists of two basic modules in the proposed system. They are Extension module and Project module. The Extension module finds all the possible extensions of a subgraph. This module takes as input a single graph (Graph structure block) and a subgraph (DFS Structure block) and it outputs all the possible valid one-edge extensions of the input subgraph. In other type Project module is the top-level module that is mapped to a single FPGA chip. The Project module consists of eight parallel Extension modules and an edge filter component for the final output.

4.2 Review based on the Graph Mining

Some of the literature based on the parallel processing of the graph mining function is listed as follows; Athanasios Stratikopoulos et al. [12] have presented the first FPGA-based implementation of the most efficient and well-known algorithm for the Frequent Subgraph Mining (FSM) problem, i.e. gSpan. David Apar´ıcio et al. [17] have presented a scalable algorithm to Count subgraph frequencies for multicore architectures. They used the methodology of state-of-art g-trie data structure. Simon Fong et al. [20] have presented the scalable data stream mining methodology called stream-based Holistic Analytics and Reasoning in Parallel (SHARP). SHARP was scalable, that depends on the available computing resources during the runtime, and the components executed in parallel. Kartik Nayak et al. [28] have introduced the Graph SC to efficiently implement an oblivious version of graph-based algorithms that executed in parallel with minimal communication overhead. Linchuan Chen et al. [34] have described the system for graph processing utilizing both the CPU and the MIC chip. It supported the vertex-centric high-level API that can express a graph application easily. Umut A. Acar et al. [44] have presented the strongly work-efficient PDFS algorithm. They proved that the algorithm achieves strong work efficiency by bounding important scheduling overheads. Shaden Smith et al. [49] have presented the parallel algorithm for computing the truss decomposition of massive graphs on a shared-memory system. The shared-memory parallel algorithm was based on peeling as called as multi-stage peeling (MSP). Anand Tripathi et al. [56] have presented the transactional model of parallel programming. It provided a conceptually simple approach for harnessing amorphous parallelism in graph problems. Xinyu Wu et al. [58] have presented the Path-Dimension Graph Cube model based on the concept of relation path set for heterogeneous network to solve the problem of insufficient analysis of existing Graph OLAP Model.

5. REVIEW ON OPTIMIZATION FUNCTION

The purpose of optimization is to achieve the "best" design relative to a set of prioritized criteria or constraints. These include maximizing factors such as productivity, strength, reliability, longevity, efficiency and utilization. This heuristic searching process is known as optimization. Optimization is defined as the process of finding the conditions that give the minimum or maximum value of a function where the function represents the effort required or the desired benefit.

5.1 Parallel processing of Optimization Function

In this section, we describe the parallel genetic algorithm for the parallel processing of optimization from the ref [61]. Genetic mechanisms were successfully used for many optimization problems. Parallel genetic Algorithms are an extension of genetic algorithms. They are designed for parallel computers. When genetic algorithms are based on idealized population model. The algorithm simulates a limited population evolution very realistic. The algorithm can be explained by the system theory of evolution. Each individual is part of the environment of other individuals, selection is totally distributed. The evolutionary process is motivated by individuals. In addition to the optimization of artificial problems. The parallel genetic algorithms can be used also investigate about the population genetics problems. In fact, genetic algorithm and popular genetics research is facing the same problem. To understand why evolution with powerful genetic operators and selection works so effectively. The basic method is described as follows:

Step 1: Define a genetic representation of the optimization problem
Step 2: Create 'N' individual
Step 3: Each individual does local hill climbing (increases its fitness)
Step 4: Each individual chooses a partner for mating in the 'neighbourhood' - - local selection

Step 5: Create a new offspring with genetic operators (e.g. crossover, mutation)
Step 6: Replace the individual
Step 7: If not finished, go to Step 3

The neighborhood is defined as individuals live in an environment e.g. a two-dimensional grid. Each individual has a grid point. The neighborhood is then given by some distance. Mating and selection of each individual is done in its neighborhood. The climbing of the local hill has the following effect: Recombination is now done in the space of all genotypes representing some local fitness maxima. Parallel genetic algorithms have been described in [C*87,P*87,Tan87]. In these algorithms, a standard genetic algorithm operates on Subpopulations. Subpopulations are evaluated in parallel. Each individual is evaluated asynchronously in parallel. Most of the genetic algorithms developed so far are based on an idealized population model. This model describes the development of an infinite population with random mating and selection according to

$$p_i(t+1) = p_i(t)\frac{f_i}{\bar{f}} \quad (2)$$

The fundamental problem of random mating (crossover) was already pointed out by Darwin's [Dar59]. Interpolation plays a very important role in maintaining character in the uniform of individual. Darwin's conjecture that local populations with some isolation evolve faster than a large population was confirmed later by experiments and qualitative analysis [Wri32].

5.2 Review based on Optimization

Few recent researches related to the parallel optimization for the data mining function is given below.

Jianguo Chen et al. [4] have presented the Parallel Random Forest (PRF) algorithm on Apache Spark platform. The Random Forest algorithm was a suitable data mining algorithm for big data. PRF algorithm was based on a hybrid parallel approach. Suejb Memeti et al. [21] have introduced the novel algorithm for Parallel Regular Expression Matching (PaREM). The algorithm was optimized to do very accurate speculations on the possible initial states for each of the sub inputs. Jiangling Yin et al. [32] have presented the novel method to Optimize Parallel Data Access on Distributed File Systems referred to as Opass. The goal of Opass was to reduce remote parallel data accesses and achieve a higher balance of data read requests between cluster nodes. Lina Yang et al. [39] have developed the ABC algorithm to determine the optimal solution to the Endmember extraction in hyperspectral remote sensing. The ABC algorithm was implemented on an established multiagent platform for parallel execution. Scott Sallinen et al. [43] have presented the several modern parallelization methods of SGD on a shared memory system, in the context of sparse and convex optimization problems. They developed the optimized parallel implementations of SGD algorithms. Bart van Strien et al. [54] have introduced the extended forelem framework and to optimising a Hadoop MapReduce. The application was determining its effectiveness on non-database applications. TA graph ranking algorithm was developed to find the most important web pages in a web graph.

6. REVIEW ON FREQUENT ITEM SET MINING (ASSOCIATION RULE MINING)

Frequent itemsets are a form of frequent pattern. These are the sets of items and a minimum frequency, any set of items that occurs at least in the minimum number of examples is a frequent itemset. Frequent itemset mining is a data analysis method that was originally developed for market basket analysis. It aims at finding regularities in the shopping behavior of the customers of supermarkets, mailorder companies, and online shops. Frequent Itemset Mining (FIM) algorithms are used for finding common and potentially interesting patterns in large-scale databases. In FIM algorithms, the data in the database are called transactions, each of which is a set of items labelled by a unique ID. The purpose of FIM is to find the most frequently-occurring subsets from the transactions.

6.1 Parallel Processing of Frequent Itemset Mining (Association Rule Mining) Function

In this section we describe the Parallel Apriori Algorithm based on Bodon's work for parallel association rule mining, which is referred from [63]. Data mining known as knowledge discovery in databases (KDD) is the process of automatically extracting useful hidden information from very large databases. One of central themes of data mining is association rules mining between itemsets in vary large databases. The parallel apriori algorithm for the association rule mining is described below. The original Apriori algorithm was implemented using hash tree. Bodon implemented the Apriori algorithm using the trie structure. The depth of the root is zero. There are edges between two nodes and each edge is labeled by a letter. Apriori implementation is provides more efficiency than the other implementations.

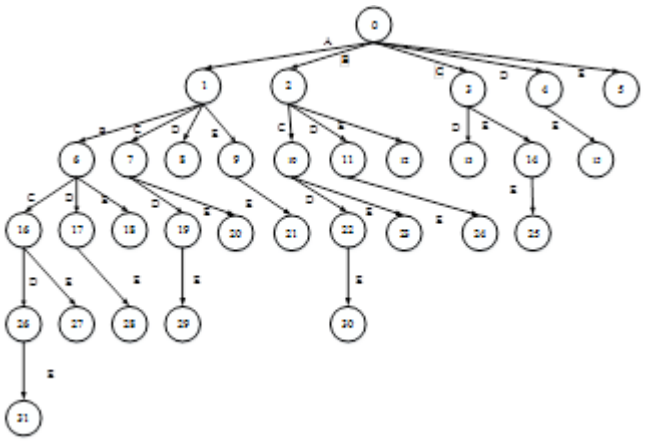


Figure 5: *A trie containing itemsets.*

The Apriori algorithm has been revised in several ways. One revision of the Apriori algorithm is to partition a transaction database into disjoint partitions $TDB1, TDB2, TDB3, \ldots TDBn$. Partitioning a transaction database may improve the performance of frequent itemsets mining, the parallel apriori algorithm is given in fig 6.

```
Inputs
        TDB: Transaction Database
        minsup: Minimum Support
Outputs
        i: Frequent Itemset // Supp(Itemset) >= minsup
Temporary Variables
        Ck: Candidate Itemset
Get L
        1 for 1-itemsets;
        k = 2;
        While (Lk-1 <> empty)
                        k= Candidate_Generation(Lk-1)
                        For each transaction T in TDB
                                For each itemset c in C
                                        If T includes c then
                                                count[c] = count[c] + 1;
                                        End;
                                End;
                        End;
                        k= Prune Candidate (Ck, minsup);
                        k= k + 1
        End;
Output L1, L2,....,Lk;
```

Figure 6: *Apriori algorithm*

6.2 Review Based on the Frequent Itemset Mining (Association Rule Mining)

Some of the literature based on the parallel processing of the Frequent Itemset Mining (Association Rule Mining) function is listed as follows; Hongjian Qiu et al. [1] have proposed the YAFIM algorithm. YAFIM was a parallel Apriori algorithm based on the Spark RDD model. Frequent itemset mining technique are used. One of the important techniques in real world application. Jin Soung Yoo et al. [3] have presented the Spatial association rule mining was a useful tool for discovering correlations and interesting relationships among spatial objects. The parallel co-location mining algorithm was developed on MapReduce. Jinggui Liao et al. [5] have proposed the adaption to the big data mining parallel algorithm based on the Map reduced Pre-Post (MRPrePost). The MRPrePost proposed algorithm was suitable for large scale Data set for mining association rules. Fabio Fumarola et al. [6] have presented the MrAdam, an algorithm for frequent itemset mining from Big Data using Map Reduce. These data set were created in parallel to the input of the database MapReduce by exploiting the parallel computing structure. Albino Altomare et al. [10] have presented the workflow-based parallel approach for discovering patterns and rules from trajectory data in a Cloud-based framework. To implemented a parallel methodology modeled by the workflow formalism for pattern discovery from trajectory data. Yaling Xun et al. [14] have applied the MapReduce programming model to develop a parallel frequent itemsets mining algorithm called FiDoop. In FiDoop, three MapReduce jobs were implemented to complete the mining task. Mayank Tiwary et al. [23] have introduced the integration of GPU in mapreduce programming model to solve the apriori data mining technique in a very time efficient manner. Massimo Cafaro et al. [24] they have presented the four CUDA based parallel implementations of the Space-Saving algorithm for determining frequent items on a GPU. They implemented a warp-based ballot mechanism to accelerate the Space-Saving updates. Hao Lin et al. [25] have described the highly parallel R system called RABID. The RABID was the R Analytics for Big Data. The performance of RABID on two applications RHIPE and Hadoop, and has speedups of up to 20x and 10x. Oguz Kaya et al. [26] have discussed the efficient parallelization of the alternating least squares-based Tucker decomposition algorithm for sparse tensors in shared and distributed memory systems. A fine-grain parallel algorithm was used in a distributed memory environment. Junbo Zhang et al. [29] have processed a large-scale incomplete data with rough set theory in parallel process. Rough set theory, which used successfully in solving problems in pattern recognition, machine learning, and data mining. Yunquan Zhang et al. [30] have introduced the basic framework of MapReduce and its outstanding features as well as deficiencies compared to DBMSs. MapReduce was outstanding for better simplicity, scalability, and fault-tolerance. Adib Haron et al. [33] have presented the communication efficient mapping of a large-scale matrix multiplication algorithm on the CIM architecture. A memristor-based CIM architecture was a new computing architecture that integrates computation and memory in the same physical location rather than Von Neumann architecture. Paweł Czarnul et al. [35] have presented the implementation of similarity measure for big data analysis in a parallel environment. The simulations allowed to determine be performed on a particular hardware. Ruben Mayer et al. [36] have proposed a pattern-sensitive stream partitioning model. The proposed pattern-sensitive stream partitioning method supported the consistent parallelization of a wide class of important CEP operators. Themis Palpanas [46] have described the past efforts in designing techniques for indexing and mining truly massive collections of data series. It was based on indexing techniques for fast similarity search of mining algorithms. Fabrizio Marozzo et al. [47] have described the design and implementation of the Data Mining Cloud Framework (DMCF). A system that integrated a visual workflow language and a parallel runtime with the SaaS model for enabling the scalable execution of complex data analysis. Ivan I. Kholod et al. [48] have described the representation of a mining model in a hierarchy of lists that can be processed concurrently. This representation was based at PMML standard that defines the mining model's elements and their relationships. Hao JIANG et al. [50] have proposed the PFP-growth algorithm based on GPU. Uses of GPU's powerful parallel computing ability to mine frequent itemsets, greatly improved the mining efficiency. Zonyin Shae et al. [55] have proposed the new blockchain distributed parallel computing architecture for big data analysis. Fabian Gieseke et al. [57] have described the efficient implementation for this brute-force approach. Our key contribution was an efficient massively-parallel implementation. Guangchen Ruan et al. [9] have presented the sequential pattern mining framework PESMiner, which means quantitative sequential patterns from large-scale interval-based temporal data in a distributed function. The experimental results demonstrated PESMiner's promising performance in terms of both quality and scalability.

7. REVIEW ON PARALLEL PROCESSING OF OTHER DATA MINING FUNCTION

In the previous sections the list of literatures based on the parallel processing of data mining functions includes classification, clustering, optimization, association rule and graph mining is given. In this section few literatures based on the parallel processing of data mining function but not comes under the above category is given. Nagarajan Kathiresan et al. [15] have used the BWA MEM algorithm for genome alignment and it's dominated by memory intensive operations. The

performance of proposed data-parallel with concurrent parallelization was 45% better than traditional parallelization approaches. Fenghua Zhu et al. [27] have expanded to the PTMS for new generation of intelligent transportation systems. Smart transportation was playing an important role for the implementation of smart city. Ioannis Konstantelos et al. [37] have used the High-Performance Computing (HPC) for implement the massively parallel dynamic security assessment (DSA) platform. The iTesla developed a prototype toolbox for DSA of large-scale grids that combines offline and online functionality.

8. SURVEY SUMMARY

In this literature survey we have reviewed more than 60 literatures published between 2014 and 2019 the detail of the literature is given in table 1. From the table we can clear that the most of the research is concentrated on the Association rule mining for the frequent item set mining. The second largest literature work is clustering and graph. Based on these summaries we can conclude that more research should be encouraged in parallelization of classification and optimization.

Table 1: *Summary of Literature*

Year	No of Literature					
	assification	Optimization	Clustering	Association	Graph	Other
2014	-	1	2	8	3	1
2015	2	2	6	4	3	-
2016	1	2	3	4	3	2
2017	2	1	2	5	-	-
2018	1	-	-	1	-	-
Total	6	6	13	22	9	3

9. COMPARATIVE ANALYSIS

The major intension of this work is to study the various techniques for the parallel processing of data mining function based on its performance. Hence five parallel data mining functions, such as PKM, PGA, gSpan, PA, and PBN are described. In this section these algorithms are implemented on two Data set s using Matlab in windows computer.

9.1 Data Set Description

Data set #1: Student Performance (SP) Data Set

This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features [74].

Data set #2: Chronic Kidney Disease (CKD) Data Set

This dataset can be used to predict the chronic kidney disease and it can be collected from the hospital nearly 2 months of period [75].

9.2 Performance Analysis

The performance of considered five data mining functions in two case such as with and without parallel processing is compared. The performance of clustering function by k-means and parallel k-means (PKM) is given in table 2.

Table 2: *Clustering Performance*

Technique	No of Cluster	Execution time	
		SP	CKD
K-means	2	0.0530	0.1999
Parallel k-means		0.0493	0.1839
K-means	4	0.0176	0.0964
Parallel k-means		0.0163	0.0887

The table 2 shows the clustering performance based on the execution time of normal and parallel algorithm. It is clearly shows that the parallel cluster technique comparatively consumes less time to the normal technique. Then the optimization performance in given in table 3.

Table 3: *Optimization Performance*

Technique	No of Iteration	Execution time	
		SP	CKD
GA	100	0.2031	0.3048
Parallel GA		0.1950	0.2896
GA	500	0.2519	0.4877
Parallel GA		0.2418	0.4682

The optimization performance is given in table 3, in which the execution time for the optimization of 100 and 500 iteration by normal and parallel GA is evaluated. In this evaluation we can understand the betterment of parallel technique. Because in both the cases the parallel technique executed fast as compared to the normal optimization technique. The performance of graph mining is given in table 4.

Table 4: *Graph Mining Performance*

Technique	Execution time	
	SP	CKD
Graph Mining	2.1900	0.5588
gSpan	1.9710	0.5085

The performance of graph mining and gSpan is evaluated based on the execution time and is given in table 4. The normal graph mining technique takes 0.5588 sec to complete the CKD data set. But the parallel gSpan technique executed in 0.5085 sec. Based on this comparison we can confirm that the parallel processing can reduce the execution time. The performance of association rule mining is given in table 5.

Table 5: *Performance of Association rule mining*

Technique	Execution time	
	SP	CKD
Apriori	8.1997e-05	0.1815
Parallel Apriori	7.5437e-05	0.1688

The table 5 gives the comparison of association rule mining in normal and parallel case. The parallel apriori for the association rule mining executed fast as compared to the normal apriori. The performance of classification function is given in table 6.

Table 6: *Performance of Classification function*

Technique	Execution time	
	SP	CKD
BN	0.4434	0.4221
PBN	0.4168	0.4052

The classification technique in parallel and normal case is compared in the table 6. The execution time-based comparison of classifier provided better performance in parallel processing. The overall comparison of normal and parallel technique is given in fig 7.

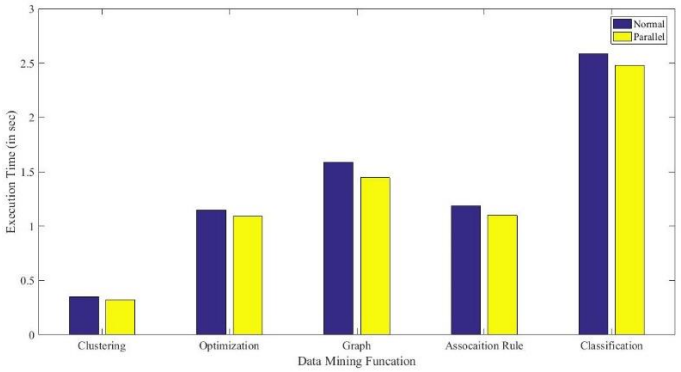


Figure 7: *Comparison chart*

The chart given in fig 7 shown above gives the comparison of normal and parallel data mining functions. The chart shows the execution time of various data mining techniques. The chart clearly proves that the parallel technique provided better performance than the normal technique. Hence based on this performance evaluation we can suggest that the parallel processing of data mining function has its advantages and executes faster than the normal function. So it is evident that the parallel execution of data mining function should be encouraged for the future development.

10. FINDING FROM THE SURVEY

The survey is conducted to understand the recent study on parallel processing of data mining function and its performance is evaluated by implementing five different data mining function with and without parallel processing. Based on the survey and performance analysis, it is evident that the parallel processing should provide reasonable benefit to improve the computation speed. In literature most of the researchers proposed parallelization technique for association rule mining and in the next level graph and clustering. Thus in future the better technique to parallelization of classification and optimization is encouraged.

11. CONCLUSION

Big data processing is an encouraging research topic in resent decade. The parallelization in big data is beneficial to speed up the process, which can be achieved by big data processing techniques and tools. But the implementation of these tools is complex, and insecure. Hence the researchers are suggested to parallel data mining functions. It can reduce the complexity in handling the large data. In this paper the researched published during past five years is reviewed and are categorised under five category. Then a simple method from each category is implemented on a same platform using same database to analysis performance in terms of execution time. The performance analysis strongly proved the betterment of using parallel processing. The survey suggested that the past research is mostly concentrated on association rule mining. Hence in future a better strategy for the parallelization of classification and optimization function in data mining is encouraged.

12 REFERENCES

[1] H. Qiu, R. Gu, C. Yuan, and Y.Huang. "Yafim: a parallel frequent itemset mining algorithm with spark." In 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, pp. 1664-1671. IEEE, 2014.

[2] Kholod, "Framework for multi threads execution of data mining algorithms." In 2015 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), pp. 82-88. IEEE, 2015.

[3] J.S. Yoo, D. Boulware and D. Kimmey. "A parallel spatial co-location mining algorithm based on MapReduce." In 2014 IEEE International Congress on Big Data, pp. 25-31. IEEE, 2014.

[4] J. Chen, K. Li, Z.Tang, K. Bilal, S. Yu, C. Weng, and K. Li. "A parallel random forest algorithm for big data in a spark cloud computing environment." IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 4, pp. 919-933, 2017.

[5] J. Liao, Y. Zhao and S. Long. "MRPrePost—A parallel algorithm adapted for mining big data." In 2014 IEEE Workshop on Electronics, Computer and Applications, pp. 564-568, IEEE, 2014.

[6] F. Fumarola and D. Malerba "A parallel algorithm for approximate frequent itemset mining using MapReduce." In 2014 International Conference on High Performance Computing & Simulation (HPCS), pp. 335-342, IEEE, 2014.

[7] M. Bendechache and M-T. Kechadi. "Distributed clustering algorithm for spatial data mining." In 2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM), pp. 60-65. IEEE, 2015.

[8] M.O. Shafiq and E. Torunski. "A parallel K-medoids algorithm for clustering based on MapReduce." In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 502-507. IEEE, 2016.

[9] G. Ruan, H. Zhang and B. Plale. "Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data." In 2014 IEEE International Conference on Big Data (Big Data), pp. 32-39. IEEE, 2014.

[10] Altomare, E. Cesario, C. Comito, F. Marozzo and D. Talia. "Trajectory pattern mining for urban computing in the cloud." IEEE Transactions on Parallel and Distributed Systems, vol.28, no. 2, pp. 586-599, 2017.

[11] G. Buehrer, R.L.D. Oliveira, D. Fuhry and S. Parthasarathy. "Towards a parameter-free and parallel itemset mining algorithm in linearithmic time." In 2015 IEEE 31st International Conference on Data Engineering, pp. 1071-1082. IEEE, 2015.

[12] Stratikopoulos, G. Chrysos, I. Papaefstathiou and A. Dollas. "HPC-gSpan: An FPGA-based parallel system for frequent subgraph mining." In 2014 24th International Conference on Field Programmable Logic and Applications (FPL), pp. 1-4. IEEE, 2014.

[13] H. Gao, Z. Yang, J. Bhimani, T. Wang, J. Wang, B. Sheng and N. Mi. "AutoPath: harnessing parallel execution paths for efficient resource allocation in multi-stage big data frameworks." In 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1-9. IEEE, 2017.

[14] Y. Xun, J. Zhang, and X. Qin. "Fidoop: Parallel mining of frequent itemsets using mapreduce." IEEE transactions on Systems, Man, and Cybernetics: systems, vol.46, no. 3, pp. 313-325, 2016.
[15] N. Kathiresan, M.R. Temanni and R. Al-Ali. "Performance improvement of BWA MEM algorithm using data-parallel with concurrent parallelization." In 2014 International Conference on Parallel, Distributed and Grid Computing, pp. 406-411. IEEE, 2014.
[16] Shabib, A. Narang, C. P. Niddodi, M. Das, R. Pradeep, V. Shenoy, P. Auradkar, T.S. Vignesh and D. Sitaram. "Parallelization of searching and mining time series data using Dynamic Time Warping." In 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 343-348. IEEE, 2015.
[17] D.O. Aparício, P.M.P. Ribeiro and F.M.A. D. Silva. "Parallel subgraph counting for multicore architectures." In 2014 IEEE International Symposium on Parallel and Distributed Processing with Applications, pp. 34-41. IEEE, 2014.
[18] Kun Yue, Qiyu Fang, Xiaoling Wang, Jin Li, and Weiyi Liu. "A parallel and incremental approach for data-intensive learning of Bayesian networks." IEEE transactions on cybernetics 45, no. 12 (2015): 2890-2904.
[19] K.D. Garcia, and M.C. Naldi. "Multiple parallel mapreduce k-means clustering with validation and selection." In 2014 Brazilian Conference on Intelligent Systems, pp. 432-437. IEEE, 2014.
[20] S.Fong, Y. Zhuang, R. Wong and S.Mohammed. "A scalable data stream mining methodology: stream-based holistic analytics and reasoning in parallel." In 2014 2nd International Symposium on Computational and Business Intelligence, pp. 110-115. IEEE, 2014.
[21] S. Memeti and S. Pllana. "Parem: A novel approach for parallel regular expression matching." In 2014 IEEE 17th International Conference on Computational Science and Engineering, pp. 690-697. IEEE, 2014.
[22] X. Fei, X.F. Li and C. Shen. "Parallelized text classification algorithm for processing large scale TCM clinical data with MapReduce." In 2015 IEEE International Conference on Information and Automation, pp. 1983-1986. IEEE, 2015.
[23] M. Tiwary, A.K. Sahoo and R. Misra. "Efficient implementation of apriori algorithm on HDFS using GPU." In 2014 International Conference on High Performance Computing and Applications (ICHPCA), pp. 1-7. IEEE, 2014.
[24] M. Cafaro, I. Epicoco, G. Aloisio and M. Pulimeno. "Cuda based parallel implementations of space-saving on a gpu." In 2017 International Conference on High Performance Computing & Simulation (HPCS), pp. 707-714. IEEE, 2017.
[25] H. Lin, S. Yang and S.P. Midkiff. "RABID: A distributed parallel R for large datasets." In 2014 IEEE International Congress on Big Data, pp. 725-732. IEEE, 2014.
[26] O. Kaya and B. Uçar. "High performance parallel algorithms for the tucker decomposition of sparse tensors." In 2016 45th International Conference on Parallel Processing (ICPP), pp. 103-112. IEEE, 2016.
[27] F. Zhu, Z. Li, S. Chen and G. Xiong. "Parallel transportation management and control system and its applications in building smart cities." IEEE Transactions on Intelligent Transportation Systems, vol.17, no. 6, pp. 1576-1585, 2016.
[28] K. Nayak, X.S. Wang, S. Ioannidis, U. Weinsberg, N. Taft and E. Shi. "GraphSC: Parallel secure computation made easy." In 2015 IEEE Symposium on Security and Privacy, pp. 377-394. IEEE, 2015.
[29] J. Zhang, J-S. Wong, Y. Pan and T. Li. "A parallel matrix-based method for computing approximations in incomplete information systems." IEEE Transactions on Knowledge and Data Engineering, vol.27, no. 2, pp.326-339, 2015.
[30] Y.Zhang, T.Cao, S. Li, X. Tian, L. Yuan, H. Jia and A.V. Vasilakos. "Parallel processing systems for big data: a survey." Proceedings of the IEEE, vol. 104, no. 11, pp. 2114-2136, 2016.
[31] J. Bhimani, M. Leeser, and N. Mi. "Accelerating K-Means clustering with parallel implementations and GPU computing." In 2015 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1-6. IEEE, 2015.
[32] J. Yin, J. Wang, J. Zhou, T. Lukasiewicz, D. Huang and J. Zhang. "Opass: Analysis and optimization of parallel data access on distributed file systems." In 2015 IEEE International Parallel and Distributed Processing Symposium, pp. 623-632. IEEE, 2015.
[33] Haron, J. Yu, R. Nane, M. Taouil, S. Hamdioui and K. Bertels. "Parallel matrix multiplication on memristor-based computation-in-memory architecture." In 2016 International Conference on High Performance Computing & Simulation (HPCS), pp. 759-766. IEEE, 2016.
[34] L. Chen, X. Huo, B. Ren, S. Jain and G. Agrawal. "Efficient and simplified parallel graph processing over cpu and mic." In 2015 IEEE International Parallel and Distributed Processing Symposium, pp. 819-828. IEEE, 2015.
[35] P. Czarnul, P. Rościszewski, M. Matuszek and J. Szymański. "Simulation of parallel similarity measure computations for large data sets." In 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF), pp. 472-477. IEEE, 2015.
[36] R. Mayer, B. Koldehofe and K. Rothermel. "Predictable low-latency event detection with parallel complex event processing." IEEE Internet of Things Journal, vol.2, no. 4, pp. 274-286, 2015.
[37] Konstantelos, G. Jamgotchian, S.H. Tindemans, P. Duchesne, S. Cole, C. Merckx, G. Strbac and P. Panciatici. "Implementation of a massively parallel dynamic security assessment platform for large-scale grids." IEEE Transactions on Smart Grid, vol.8, no. 3, pp.1417-1426.
[38] I.K. Savvas and D.Tselios. "Parallelizing dbscan algorithm using mpi." In 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 77-82. IEEE, 2016.
[39] L. Yang, X. Sun, L. Peng, X. Yao and T. Chi. "An agent-based artificial bee colony (ABC) algorithm for hyperspectral image endmember extraction in parallel." IEEE Journal of Selected Topics in Applied

Earth Observations and Remote Sensing, vol.8, no. 10, pp. 4657-4664, 2015.
[40] E. Zdravevski, P. Lameski, A. Kulakov, S. Filiposka, D.Trajanov, and B.Jakimovskik. "Parallel computation of information gain using Hadoop and MapReduce." In 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 181-192. IEEE, 2015.
[41] M. Alandoli, M. Al-Ayyoub, M. Al-Smadi, Y. Jararweh and E. Benkhelifa. "Using dynamic parallelism to speed up clustering-based community detection in social networks." In 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pp. 240-245. IEEE, 2016.
[42] X.Gao, E. Ferrara and J. Qiu. "Parallel clustering of high-dimensional social media data streams." In 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 323-332. IEEE, 2015.
[43] S. Sallinen, N. Satish, M. Smelyanskiy, Samantika S. Sury, and Christopher Ré. "High performance parallel stochastic gradient descent in shared memory." In 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 873-882. IEEE, 2016.
[44] U.A. Acar, A. Charguéraud and M. Rainey. "A work-efficient algorithm for parallel unordered depth-first search." In SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1-12. IEEE, 2015.
[45] M.M.A. Patwary, N. Satish, N. Sundaram, F. Manne, S. Habib and P. Dubey. "Pardicle: Parallel approximate density-based clustering." In SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 560-571. IEEE, 2014.
[46] T. Palpanas, "The parallel and distributed future of data series mining." In 2017 International Conference on High Performance Computing & Simulation (HPCS), pp. 916-920. IEEE, 2017.
[47] F. Marozzo, D. Talia and P. Trunfio. "A workflow management system for scalable data mining on clouds." IEEE Transactions on Services Computing, vol.11, no. 3, pp. 480-492, 2018.
[48] I.I. Kholod and A.V. Shorov. "Unification of mining model for parallel processing." In 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), pp. 450-455. IEEE, 2017.
[49] S. Smith, X. Liu, N. K. Ahmed, A.S. Tom, F. Petrini and G. Karypis. "Truss decomposition on shared-memory parallel systems." In 2017 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1-6. IEEE, 2017.
[50] J. Zhou, K-M. Yu and B-C. Wu. "Parallel frequent patterns mining algorithm on GPU." In 2010 IEEE International Conference on Systems, Man and Cybernetics, pp. 435-440. IEEE, 2010.
[51] S.T. Mai, M. S. Dieu, I. Assent, J. Jacobsen, J. Kristensen and M.Birk. "Scalable and interactive graph clustering algorithm on multicore CPUs." In 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 349-360. IEEE, 2017.
[52] B. Zhao, H. Zhou, G. Li and Y. Huang. "ZenLDA: Large-scale topic model training on distributed data-parallel platform." Big Data Mining and Analytics, vol.1, no. 1, pp. 57-74, 2018.
[53] D.G. Reis, R. N. Carvalho, R.S. Carvalho and M. Ladeira. "Two-phase parallel learning to identify similar structures among relational databases." In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1020-1023. IEEE, 2017.
[54] V. Strien, Bart, K. Rietveld and H. Wijshoff. "Deriving highly efficient implementations of parallel pagerank." In 2017 46th International Conference on Parallel Processing Workshops (ICPPW), pp. 95-102. IEEE, 2017.
[55] J. Tsai, "Transform blockchain into distributed parallel computing architecture for precision medicine." In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 1290-1299. IEEE, 2018.
[56] Tripathi, V. Padhye T.S. Sunkara, J. Tucker, B.D. Thirunavukarasu, V. Pandey and R.R. Sharma. "A transactional model for parallel programming of graph applications on computing clusters." In 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), pp. 431-438. IEEE, 2017.
[57] F. Gieseke, K. L. Polsterer, A. Mahabal, C. Igel and T. Heskes. "Massively-parallel best subset selection for ordinary least-squares regression." In 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1-8. IEEE, 2017.
[58] X. Wu, B. Wu and B. Wang. "P&D Graph Cube: Model and parallel materialization for multidimensional heterogeneous network." In 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 95-104. IEEE, 2017.
[59] H.Peng, Z. Jin and J. A. Miller. "Bayesian Networks with Structural Restrictions: Parallelization, Performance, and Efficient Cross-Validation." In 2017 IEEE International Congress on Big Data (BigData Congress), pp. 7-14. IEEE, 2017.
[60] C. Ji, Z. Xiong, C.Fang, L.V. Hui and K. Zhang. "A GPU Based Parallel Clustering Method for Electric Power Big Data." In 2017 4th International Conference on Information Science and Control Engineering (ICISCE), pp. 29-33. IEEE, 2017.
[61] Mühlenbein, Heinz. "Parallel genetic algorithms, population genetics and combinatorial optimization." In Workshop on Parallel Processing: Logic, Organization, and Technology, pp. 398-406. Springer, Berlin, Heidelberg, 1989.
[62] J. Hu, G. Wu, P. Sun and Q. Xiong. "A parallel Bayesian network learning algorithm for classification." In 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 259-263. IEEE, 2016.
[63] Y. Ye and C-C. Chiang. "A parallel apriori algorithm for frequent itemsets mining." In Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), pp. 87-94. IEEE, 2006.

[64] S. Karthick, "Semi Supervised Hierarchy Forest Clustering and KNN Based Metric Learning Technique for Machine Learning System." Journal of Advanced Research in Dynamical and Control Systems, vol.9, no. 1, pp. 2679-2690, 2017.
[65] Y.P. Arul Teen, T. Nathiyaa, K.B. Rajesh, and S. Karthick. "Bessel Gaussian beam propagation through turbulence in free space optical communication." Optical Memory and Neural Networks, vol.27, no. 2, pp. 81-88, 2018.
[66] S. Karthick, "TDP: A Novel Secure and Energy Aware Routing Protocol for Wireless Sensor Networks." International Journal of Intelligent Engineering and Systems, vol.11, no. 2, pp. 76-84, 2018.
[67] S. B. Aher and L.M.R.J. Lobo. "Data mining in educational system using Weka." In International Conference on Emerging Technology Trends (ICETT), vol. 3, pp. 20-25. 2011.
[68] T. Sathish, and S. Karthick. "HAIWF-based fault detection and classification for industrial machine condition monitoring." Progress in Industrial Ecology, an International Journal vol.12, no. 1-2, pp. 46-58, 2018.
[69] S. Karthick, S. P. Sankar and Y.P. Arul Teen. "Trust-Distrust Protocol for Secure Routing in Self-Organizing Networks." In 2018 International Conference on Emerging Trends and Innovations in Engineering And Technological Research (ICETIETR), pp. 1-8. IEEE, 2018.
[70] R. Bellazzi, and B. Zupan. "Predictive data mining in clinical medicine: current issues and guidelines." International journal of medical informatics, vol.77, no. 2, pp. 81-97, 2008.
[71] S. Karthick, S. P. Sankar and T.R. Prathab. "An Approach for Image Encryption/Decryption Based on Quaternion Fourier Transform." In 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR), pp. 1-7. IEEE, 2018.
[72] S.P. Sankar, N. Vishwanath, and H.J.Lang. "An effective content based medical image retrieval by using abc based artificial neural network (ann)." Current Medical Imaging Reviews, vol.13, no. 3, pp. 223-230, 2017.
[73] S. Karthick, E.S. Devi and R.V. Nagarajan. "Trust-distrust protocol for the secure routing in wireless sensor networks." In 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), pp. 1-5. IEEE, 2017.
[74] P. Cortez and A. Silva. “Using Data Mining to Predict Secondary School Student Performance”. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.
[75] https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease#