Parallel Approaches of Utility Mining for Big Data

Vandna Dahiya
Research Scholar, Maharshi Dayanand University, Rohtak, Haryana.
E-mail: vandanadahiya2010@gmail.com

Sandeep Dalal
Assistant Professor, Maharshi Dayanand University, Rohtak, Haryana.
E-mail: sandeepdalal.80@gmail.com

Received May 08, 2020; Accepted July 15, 2020
ISSN: 1735-188X
DOI: 10.14704/WEB/V17I2/WEB17014

Abstract

Utility Itemset Mining (UIM) is a fundamental technique to find out various itemsets with interestingness measures in addition to their quantity. It helps in finding valuable items that cannot be tracked with frequent itemset mining. There are many techniques to mine the itemsets based on their utilities, but the need of the hour is to mine them from larger datasets. This paper presents a brief overview of various approaches for utility mining, which mine using the parallel framework to enhance the pace of computation. The paper is concluded with a discussion on various challenges and openings in the field of parallel mining and provides a way for further development of the prevailing methodologies of big data.

Keywords

Utility Mining, Big Data, Spark, Parallel Computing.

Introduction

Pattern mining or itemset mining is a technique to find remarkable, hidden, and useful patterns in a database. For example, if a person buys a cell phone, he may probably be buy its cover and screen protector. If a person goes to buy a packet of milk, he may be inclined to buy the bread and cookies also. These kinds of patterns are found on mining the huge set of transactions. This information then can be used in recommendation systems and to provide personalized services to the customers based on their purchasing history. Initially, the research in pattern mining focused on frequent itemset mining (FIM) and association rule mining (ARM). Although FIM was a great discovery to mine the itemsets that occur frequently in a database, it only considers the quantity of the item. The significance of profit is missing. For example, the sale of bread and butter may be frequent but the sale of

microwave seems to be occasional and it might not be suggested in the outcome of FIM. Accordingly, the notion of utility mining was introduced.

- **Utility Itemset Mining**

The term utility mining was coined in 2006 (Yao and Hamilton). It gives semantic significance among the items and integrates internal and external utilities of an item. The internal utility is determined based on the quantity of the item whereas the external utility indicates the interestingness measure of an item, which can be in any form-profit, quantity, gain, worth, or other factors. The external utility of an item is based on the preferences and choices of the user. The product of both these utilities defines the actual utility of an item and an itemset is called a high utility itemset (HUI) if its value is above a pre-defined value called as utility threshold. Thus UIM aims to determine the itemsets based on their importance and not only their occurrence frequency. For example, buying frequency of milk may be more than a wine bottle, but the later gives more profit value per unit.

Many algorithms have been developed for HUI mining. These algorithms mine the itemsets efficiently form small datasets. Most of the algorithms first generate the candidate sets, and then the actual high utility itemsets are discovered from these candidate itemsets. This method increases the time of computation and memory requirements when the size of the dataset increases. Therefore, with the arrival of the big data era, there is a need to compute efficiently through parallel computing. Many researchers have been working in the area of computing from large datasets using parallel or distributed computing. Lin et al. proposed a parallel UP-Growth (PHUI-Growth) algorithm based on the MapReduce architecture of the Hadoop framework. Chen et al. proposed a Spark based distributed algorithm PHUI-Miner, which is an extended version of HUI-Miner. Vo et al. proposed DTWU-mining, which is based on master slave architecture of parallel processing. A few more algorithms have been developed in this area, which are being reviewed in the next sections of this paper.

Distributed/ Parallel Paradigms

There are two major challenges in mining from big data. First, the speed of generating the data is more than what a machine can process in its accessible memory. Second, to compute the patterns from such a huge amount of data. So any framework for big data mining would require two assets –data access and its computation. In case of small data sets, mining can be done easily on a single machine where the entire computing can be done in the main memory of that machine. But, for larger datasets, it is not possible. Even

if the datasets can accommodate inside main memory, it might possible that the intermediate processing may not adapt to memory. More than one computing node or cluster is required for such computing-process. Thus, parallel processing is among the leading methods to tackle the problem of mining from large datasets. But to design an efficient algorithm for parallel processing is also a challenge, as the algorithm has to deal with numerous confronts such as load balancing among nodes, scalability, the partition of work, communication costs, synchronization, fault handling, and security of data, etc. Some of the parallel and distributed frameworks are being discussed in this section-

Grid Computing: It may be defined as a computer network where resources like processing power, memory, and data storage of a computer are communal and shared with other computers in the organization to achieve a common goal. The computers can work in heterogeneity, where each computer can have different task or job to perform. The whole system is designed to process in parallel with high performance especially when all the resources are not available locally.

Multi-core Computing: Multi-core computing is a paradigm where a single silicon chip is used for two or more independent processors. The parallelism, which is achieved here is higher than grid computing because of the pipelining and multithreading. While one instruction is there in one stage of pipeline, another instruction executes in some another stage of pipeline. Multi-core computing supports high scaling but with the cost of increase in difficulty level of fabricating and debugging of chips.

Graphics Processing Units: Graphics processing unit or GPU is a programmable computer chip, which can perform rapid calculations with a high degree of parallel processing. They were originally invented to provide a 3D visual effect on screens. Initially, CPU performed calculations but with an increase in demand for computations, GPU came and took the load of CPU. Some appliances have been proposed in recent times with the help of CPU and GPU for image processing, parallel graph evaluation, and other deep learning techniques.

Hadoop and MapReduce: Due to the explosion of a huge amount of data and the need for fast and efficient processing, there was a prerequisite for the development of distributed algorithms. To design and implementation of these algorithms, the distributed platform Apache Hadoop and the programming model MapReduce have been in use since the last couple of years. Hadoop is an assemblage of open-source utilities for vast data computation on a network of computers. It can be accommodated onsite datacenter or using clouds. Applications can be written using MapReduce to process the vast amounts

of data across numerous nodes or clusters. It is a Java-based programming model and used for distributed computing. MapReduce performs two important tasks Map and Reduce. It splits the input data into individual and independent blocks of data with some replicas, which are then processed in parallel by the mapper. The output is generated as the pairs of key-value; which is sorted and then fed to Reducer as the input. Reducer reduces the overall data based on the keys. The Mapreduce functions as an overall caretaker of scheduling and monitoring. It also re-executes the failed tasks without disturbing other tasks. It also performs load balancing by re-assigning the unfinished tasks of faulty or out of order nodes to other unoccupied nodes. HDFS or Hadoop Distributed File System is another component of Hadoop, which stores large datasets in a distributed and reliable manner. But there are certain limitations with Hadoop. The key-value paradigm of Hadoop may be difficult for some of the problems. Also, all the read and write operations are performed from disk. Every iteration needs to process from the disk again, which is a costly operation and restricts the flexibility and functioning of Hadoop.

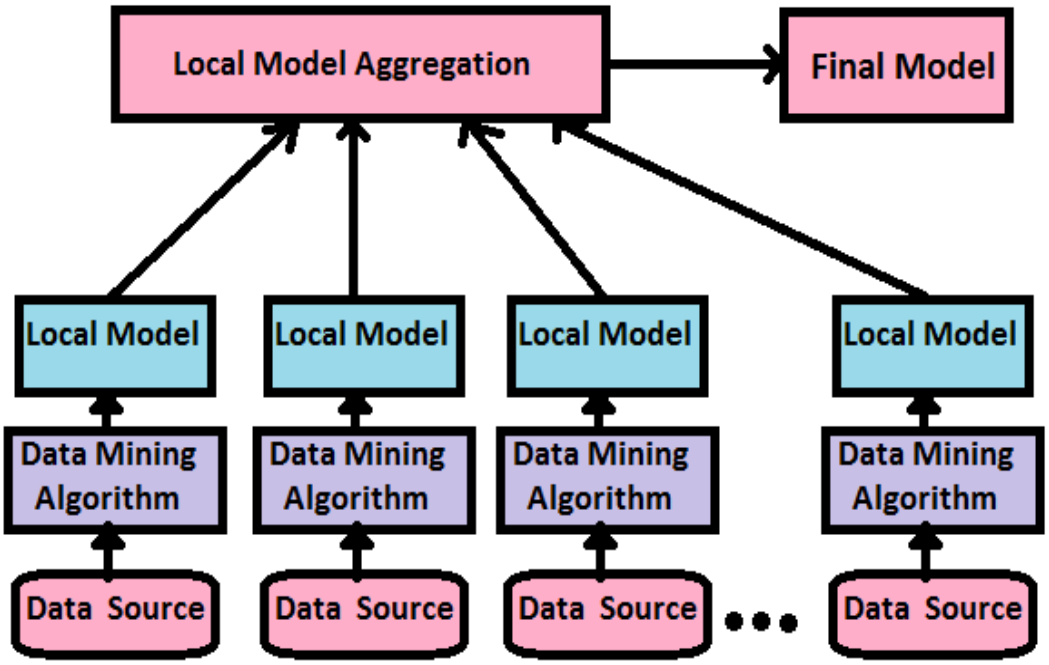

Figure 1 Distributed Data Mining Framework

Spark: Apache Spark has become the most accepted framework for distributed computing. It is an integrated analytics engine for large-scale processing of data and overtakes Hadoop with its in-memory computation-feature. Disk reading for each iteration is difficult for large data sets, which enhances the reason for the development and use of Spark over Hadoop. In-memory processing is done at each iteration even for large

clusters in the RDD – Resilient Distributed Data sets. RDD is a read-only collection of data items, which are spread across several clusters and make it fault-tolerant. In spark, there can be independent Map and Reduce operations that means a Reduce operation does not depend on Map operation and a Map operation need not be followed by a Reduce operation. This makes Spark more flexible than Hadoop. Figure 2 represents the Spark framework.

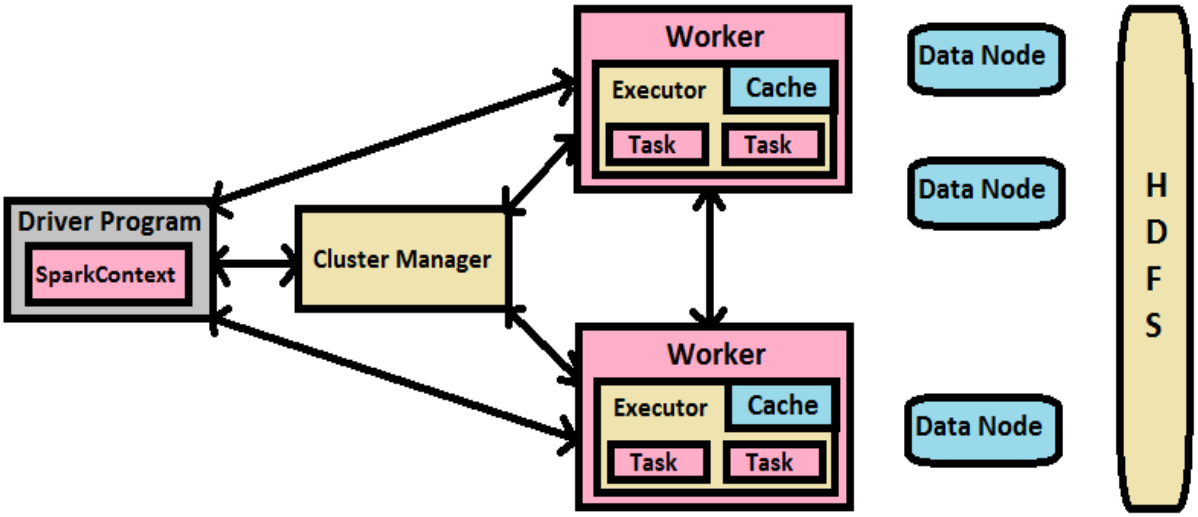

Figure 2 Framework of Apache Spark

Analysis Criteria

The architectures and frameworks discussed above are analyzed based on various crucial factors, such as how the search space is split, how the data is represented, the number of stages and overhead of communication, etc. They are briefly discussed here:

Search Space Division: The division of search space among various nodes imitates the overall performance of computation. The dataset or the problem is divided into sub-tasks. The division method constructs various projected databases form the input database, which are then distributed among the nodes. Each projected database must have the basic data to generate the local itemsets without being dependent on the outcome of other nodes. The local outcomes of each subtask are combined to generate the global output as the final result.

Database Layout: Data representation plays an important role in parallel mining. There are two types of data representations – Horizontal layout and Vertical layout. In a horizontal layout, data is stored traditionally in rows. In a vertical layout, data is stored in terms of key-value pairs in columns. The layout has a significant impact on speed,

scalability, and the overall process. In some algorithms, layout also decides the number of database scans.

Communication Cost: The overhead of communication is decisive in parallel programming because it can affect the whole speed of processing. The network can go into logjam if there is a frequent transfer of massive files. A good paradigm needs to have a reduction in transmission costs.

Load Balancing: Load balancing is a key factor in distributed processing as it affects the overall efficiency of the computation. A good distribution is strategic for balanced processing. Although, the sub tasks may have different computation-complexities, which may lead to different time complexities. Some nodes may be overburdened while some may rest in idle. Dynamic load balancing techniques are useful in such cases, so that optimum utilization of resources can be done.

HUIM Methodologies for Big Data

In this section, some of the algorithms are being discussed for high utility itemset mining, which are based on parallel computation, either with software or hardware.

DTWU-Mining –Vo et al. proposed this algorithm in 2009. This algorithm extends the TWU-Mining with distributed approach. It uses the master-slave design for parallel processing with the help of message passing procedure. The whole database is divided among the slave nodes, which then compute the high utility itemsets locally. The master node computes the itemsets, which satisfies some minimum constraints at slave nodes. WIT-tree structure is used at slave nodes to store the database. The algorithm needs only one scan for computing the itemsets at slave nodes. So, DTWU-Mining outperforms the TWU-Mining in terms of overall execution time. But, this algorithm lacks the important features of fault tolerance and fault recovery-process. If any slave node gets crashed, the whole process of mining can come to halt or might produce inaccurate results of mining.

FUM-D –Subramanian et al. proposed FUM-D or distributed fast utility mining algorithm in 2013. It is based on distributed architecture and FUM. It is a two-step process. Firstly, the candidate itemsets are computed locally and then their utility is estimated. In the second step, total utilities are calculated at the master node based on the utilities of slave nodes. This approach is roughly ten times faster than FUM but the cost of communication is very high.

PHUI-Growth – Lin et al. proposed this algorithm in 2015. It is an Apriori based parallel implementation of PHUI algorithm with the MapReduce framework. It uses several great features of Hadoop such as fault tolerance, fault recovery, low communication cost, high scalability, and easy utilization of commodity hardware. MapReduce splits the whole job into smaller autonomous sub-problems. Hadoop Distributed File System is used for storage and processing of data. DLR-MR (discarding local unpromising items in MapReduce framework) is introduced as the novel strategy of pruning to reduce the search space greatly by aborting the intermediate itemsets, which are less promising. This algorithm is very scalable and outperforms other algorithms but there is an overhead of multiple-scanning of input data.

PHUI-Miner–PHUI-Miner or Parallel High Utility Itemset Mining algorithm is an extended and parallel version of HUI-Miner. Chen et al. proposed this algorithm in 2016. It is based on Apache Spark. The database is divided into different nodes, which in turn mine the search space locally. To divide the search space, load-balancing approaches are also introduced. Other versions of this algorithm have also been proposed based on sampling and compression techniques and are used when there is the vast search space. As the size of the database increases, sampling size also increases and time to link the statistics from different nodes also increases. Although PHUI-Miner and other versions mine the data from large datasets, they only give approximate results, as there is a trade-off between accuracy and efficiency of the itemsets.

BigHUSP– Zihayat et al. proposed this algorithm in 2016. It is a Spark based algorithm to mine the high utility sequential patterns for big data. Big HUSP uses multiple MapReduce like steps to mine the data in parallel. Unpromising items are found using the overestimated utility model called as Global Sequence-Weight utility or GSWU, which has downward closure property. Utility information of the itemsets is stored in utility matrix. Two pruning strategies have been launched to minimize the search space, which then create lesser number of intermediate candidates. Pruning strategies decrease the search space. BigHUSP is further more capable in comparison of other state-of-the-art algorithms for big data.

EFIM-Par–Ashish Tamrakar proposed two MapReduce based algorithms based on the traditional EFIM in 2017. First, HUI-PR, which is for small datasets. A hash table is being used in this strategy for search space pruning. Other methods like Transaction Weighted Utilization, sub-tree utility, and local utility are also used for pruning the tree-structure. Second, EFIM-Par was introduced for big data. It is a Spark-based algorithm, where the input data is split into different blocks. These blocks are then distributed to different

worker nodes. The split dataset is also used to compute the utility-threshold value form the database itself. An improvement can be done in EFIM-Par by dividing the tasks to worker nodes in a more optimum manner.

P-FHM+ - Sethi et al. proposed this algorithm in 2018. This algorithm is a parallel implementation of FHM+. In FHM+, length constraint is being used for the itemsets, where itemsets of desired lengths are computed. Data is processed independently on multiple nodes in a distributed manner. HDFS is used for storing the data and Scala is used to write the programs. However, the P-FHM+ is better than FHM+ in conditions of time, the operating time grows linearly on increasing the data. Also, enhanced search techniques are necessitated for P-FHM+ as it distributes the load to worker nodes inefficiently.

pEFIM –Nguyen et al. proposed this algorithm in 2018. The algorithm is an extension to the traditional EFIM algorithm. Modern multi-core processor-based architecture is used to implement the algorithm in parallel to improve efficiency and performance. Shared memory systems are used and thus the load balancing becomes easy. Since the algorithm EFIM is based on depth-first search procedure, different nodes can be assigned with different search spaces of datasets without overlapping and over crossing. pEFIM runs approximately six times faster than traditional EFIM when used with two and four working threads. Although the memory utilization increases with an increase in threads as each thread has its own private data space.

PHAUIM–Sethi et al. proposed the algorithm Parallel High-Average Utility Itemset Miner in 2019. The implementation is done on the Apache Spark framework. PHUIM is an extended edition of HAUI-Miner. In HAUI-Miner, the number of candidate sets was very high. Also, a new upper bound has been introduced named as Average Utility Upper Bound or AUUB to measure the high-average utility of itemsets. Dataset is distributed among the nodes for parallel computation. The algorithm implements an improved search space division approach, where the search space is partitioned equitably to all the nodes and thus the overall performance is improved. On experimental evaluation is has been found that PHAUIM outperforms HAUIM with a huge margin in speed and scalability.

Table 1 Various Big Data based Algorithms for Utility Mining

Algorithm & Author	Year	Extends	Pros	Cons
PHAUIM, Sethi et al.	2019	HAUI-Miner	Itemset Mining is done based on average utility and search space is improved.	Run time increases with an increase in transactions.
pEFIM, Nguyen et al.	2018	EFIM	Static load balancing with parallel tasks.	Memory consumption is very high with high number of threads and dynamic load balancing is needed.
P-FHM+, Sethi et al.	2018	FHM+	Utility mining with a length constraint of itemsets.	Load distribution is very inefficient.
EFIM-Par, Ashish T	2017	EFIM	Job division is better among nodes and less number of candidates.	Tree construction is not efficient with poor grouping method.
BigHUSP, Zihayat et al.	2016	USpan	The efficient strategy of pruning search space.	Manifold jobs of MapReduce where computation time increases abruptly with growth in the database.
PHUI-Miner, Chen et al.	2016	HUI-Miner	The workload is balanced among the nodes. Sampling and compression techniques are used for search space pruning.	The sample size is not appropriate always and search space partition is not uniform.
PHUI-Growth, Lin et al.	2015	FP-Growth	High scalability on big data sets and effective pruning strategies.	Multiple scans of database and slow merging from different mappers.
DTWU-Mining, Vo et al.	2009	TWU-Mining	Low communication overhead and no merging of data from nodes is required.	No scalability and no fault tolerance.

Challenges and Future Directions

Big data mining is a novel area. The devices, systems, and algorithms are in their early stages of development. The challenges in big data mining are poles apart from the conventional system of data mining, which need to be addressed. Some of them are considered here-

Privacy: Data mining provides useful perceptions about users but it can also lead to intimidating the privacy of the user. Various information of users such as location, priorities, personalized services, etc. need to be well-preserved using the privacy-preserving techniques (PPT). Only a few algorithms in parallel mining are associated with privacy-preserving techniques. Algorithms for parallel mining need to be developed with PPT.

Scalability: Inadequate scalability is the main concern for distributing paradigms in data mining as every system has its own constraints for scaling. Some of the algorithms tend to show an abrupt increase in time and space with an increase in input data.

Security: Security is another issue for dynamic and complex data. The sensitive data of the user should be protected from any external interface. Retrieve, storage, and communication of data should be protected with security solutions. Presently most of the security algorithms are proposed for static and small sets of data.

Complex type of data: Most algorithms mine the utility information from sequential or transactional data. However, there is a variety of other complex data such as graphical data, time-series data, stream data, etc. in various applications. A foremost challenge is to advance the algorithms, which can compute information from such complex data.

Quality of end result: Result oriented interests are vital. The process of mining is useful if there is some end result with a specific importance. A framework needs to be there for the entire procedure of mining the data and then for characterizing the output also so that some kind of knowledge can be gained from it.

There are various promising fields and opportunities where mining with parallel computing is required. Some of the possible improvements that can be implemented in developing these algorithms and other future trends for research in this direction are discussed here:

Advanced framework: Parallel-distributed architecture for fast computing is required to enhance the performance of mining algorithms. For graceful scalability, progressive architecture is required in terms of software and hardware such as cloud-based computing or GPU based architecture.

Preprocessing: Efficient methods for preprocessing are required like sampling, compression, etc., which can reduce the input search space and so does the computation.

New applications: There are various new and innovative areas where the parallel framework can be applied for insights such as IOT, cloud computing, social media, etc.

Other issues: Further prominent issues are there such as privacy and security interests. Algorithms can be developed with privacy-preserving techniques and encrypted with security features. More efficient visualization techniques can be designed.

Conclusion

Traditional algorithms for itemset mining are inefficient for mining large datasets. Parallel and distributed frameworks have been developed to challenge the issues of mining with big data. Some of the parallel and distributed architectures along with various algorithms have been reviewed in this paper with their merits and demerits. Most of these algorithms are the parallel implementation of already existing algorithms for small datasets. Some vulnerable concerns have also been discussed to demonstrate the range of challenges in this area. Further research work can be done to enhance the utility mining algorithms with various features such as security, privacy, dynamic load balancing, etc.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for Mining Association Rules. *In: Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.

Yao, H., & Hamilton, H.J. (2006). Mining Itemset Utilities from Transaction Databases. *Data Knowledge Engineering*, *59*(3), 603-626.

Lin, Y.C., Wu, C.W., & Tseng, V.S. (2015). Mining High Utility Itemsets in Big Data. *In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham*, 649-661.

Chen, C.C., Tseng, C.Y., & Chen, M.S. (2018). Highly Scalable Sequential Pattern Mining Based on MapReduce Model on the Cloud. *In IEEE International Congress on Big Data*, 310-317.

Vo, B., Nguyen, H., Ho, T.B., & Le, B. (2009). Parallel method for mining high utility itemsets from vertically partitioned distributed databases. *In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, *Springer, Berlin, Heidelberg*, 251-260.

Ernemann, C., Hamscher, V., Schwiegelshohn, U., Yahyapour, R., & Streit, A. (2002). On advantages of grid computing for parallel job scheduling. *In 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, 39-39.

Dolbeau, R., Bihan, S., & Bodin, F. (2007). HMPP: A hybrid multi-core parallel programming environment. *In Workshop on general purpose processing on graphics processing units (GPGPU 2007)*, *28*, 1-5.

Zhang, N., Chen, Y.S., & Wang, J.L.(2010). Image Parallel Processing Based on GPU. *In IEEE 2nd International Conference on Advanced Computer Control*, 367-370.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Communication ACM*, *51*(1), 107-113.

Borthakur, D. (2007). *The hadoop distributed file system: Architecture and design*. Hadoop Project Website, *11*, 21.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., MaCauley, M., & Stoica, I. (2010). Resilient Distributed Datasets: A Fault-Tolerant abstraction for In-memory Cluster

Computing. *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*.

Subramanian, K., Kandhasamy, P., & Subramanian, S. (2013). A Novel Approach to Extract High Utility Itemsets from Distributed Databases. *Computing and Informatics*, *31*(6), 1597-1615.

Lin, J.C.W., Li, T., Fournier-Viger, P., Hong, T.P., Zhan, J., & Voznak, M. (2016). An Efficient Algorithm to Mine High Average-Utility Itemsets. *Advanced Engineering Informatics*, *30*(2), 233-243.

Chen, Y., & An, A. (2016). Approximate Parallel High Utility Itemset Mining. *Big Data Research*, *6*, 26-42.

Zihayat, M., Hut, Z.Z., An, A., & Hut, Y. (2016). Distributed and Parallel High Utility Sequential Pattern Mining. *In IEEE International Conference on Big Data (Big Data)*, 853-862.

Tamrakar, A. (2017). *High Utility Itemsets Identification in Big Data (Doctoral dissertation.* University of Nevada, Las Vegas).

Sethi, K.K., Ramesh, D., & Edla, D.R. (2018). P-FHM+: Parallel high utility itemset mining algorithm for big data processing. *Procedia computer science*, *132*, 918-927.

Nguyen, T.D., Nguyen, L.T., & Vo, B. (2018). A parallel algorithm for mining high utility itemsets. *In International Conference on Information Systems Architecture and Technology*, *Springer, Cham*, 286-295.

Sethi, K.K., Ramesh, D., & Sreenu, M. (2019). Parallel high average-utility itemset mining using better search space division approach. *In International Conference on Distributed Computing and Internet Technology*, *Springer, Cham*, 108-124.

Dalal, S., & Dahiya, V. (2018). Review of High Utility Itemset Mining Algorithms for Big Data. *In: Journal of Advanced Research in Dynamical and Control Systems– JARDCS, 10*(4),274-283.

Yun, U., Ryang, H., & Ryu, K.H. (2014). High Utility Itemset Mining with Techniques for Reducing Overestimated Utilities and Pruning Candidates. *Expert System Application*, *41*(8), 3861–3878.

Zihayat, M., Wu, C.W., An, A., & Tseng, V.S. (2015). Mining high utility sequential patterns from evolving data streams. *In Proceedings of the ASE Big Data & Social Informatics*, 1-6.

Sandeep, D., & Vandna, D. (2020). Big Data Mining: Current Status and Future Prospects. *International Journal of Advanced Science and Technology*, *29*(3), 4659- 4670.

Zaki, M.J. (2001). Parallel Sequence Mining on Shared-Memory Machines. *Journal of Parallel & Distributed Computing*, *61*(3), 401–426.

Yang, X.Y., Liu, Z., & Fu, Y. (2010). Mapreduce as a Programming Model for Association Rules Algorithm on Hadoop. *3rd International Conference on Information Sciences and Interaction Sciences*, 99–102.

Krishnamoorthy, S. (2015). Pruning Strategies for Mining High Utility Itemsets. *Expert Systems with Applications*, *42*(5), 2371-2381.

Dalal, S., & Dahiya, V. (2019). Various Research Opportunities in High Utility Itemset Mining. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(4), 2455-2461.

Tseng, V.S., Wu, C.W., Fournier-Viger, P., & Philip, S.Y. (2015). Efficient algorithms for mining top-k high utility itemsets. *IEEE Transactions on Knowledge and data engineering*, *28*(1), 54-67.

Apache Software Foundation. http://www.apache.org/

Hadoop. http://hadoop.apache.org/

IBM Quest Data Mining Project, Quest Synthetic Data Generation Code. (https://sourceforge.net/projects/ibmquestdatagen/)

Spark. http://spark.apache.org/