ISSN: 2366-1313

The Improved Efficiency of Data Mining Via Hierarchical Analysis for High Data Density in Big Data Servers

RAMESH BOLLI, Research Scholar, Department of CSE, J.S University, Shikohabad, U.P.

Dr. SURIBABU POTNURI, Professor, Supervisor, Department of CSE, J.S. University, Shikohabad, U.P.

Abstract: Big data is a term for massive data sets having large, more varied and complex structure with the difficulties of storing, analyzing and visualizing for further processes or results. The process of research into massive amounts of data to reveal hidden patterns and secret correlations named as big data analytics. These useful information for companies or organizations with the help of gaining richer and deeper insights and getting an advantage over the competition. For this reason, big data implementations need to be analyzed and executed as accurately as possible. Utility itemset mining is an emerging field that identifies multiple relationships between various database items, which may be utilized as the foundation for a variety of information management and decision-making systems. This thesis is primarily concerned with developing data mining technique(s) for extracting patterns of utility- based itemsets from large datasets. The research community has extensively researched frequent itemset mining to mine such patterns from a transaction database, where a transaction represents the set of items purchased together by a user. The Frequent Itemset Mining (FIM) method assumes that all items in a transaction database are of similar relevance. Customers, however, buy items in varying amounts, and products yield varying revenues. FIM may produce patterns with a high frequency but low profitability or interest. The FIM rules are prevalent, although they aren't always interesting. The rules that deviate from expectations are the most interesting. This paper also presents the implementation of K-means based on the new similarity measure using MapReduce framework. We introduced novel similarity measure based K-means algorithm for clustering big data using MapReduce. Inter and Intra-cluster density has been used to validate the results of the new similarity measure. The results of this new similarity measure helps to obtain good clustering for big data. This paper presents a smart application for traffic jam and congestion avoidance in the smart city using parallel k-means algorithm. In this work, we made use of in-memory computing to get superior performance of parallel K-means algorithm which helps us to build a real-time application for smart traffic system in the smart city.

ISSN: 2366-1313

1. INTRODUCTION

Data mining is an analytical method for extracting hidden information, trends, and patterns from a variety of data sources. It is a set of specific methods and techniques for extracting patterns from large amounts of data, as shown in figure 1. According to Williams, "Data mining is a technology designed with the objective of enabling data exploration, data analysis, as well as data visualization of large databases at a high level of abstraction without any specific hypothesis in mind" (William et al.,1998). Furthermore, the larger the size of the data, the higher the opportunity value and the more effective the data-mining tool should be in order to extract useful information. Data mining has a wide range of applications in today's world, including data analysis in the health sector, computational biology, cyber-crime detection, web mining, sentiment analysis, decision making, weather

forecasting, and so on (Rodriguez et al., 2016).

Figure1: The process of data mining

Data mining is not a stand-alone procedure; rather, it is an integral part of the "Knowledge Discovery from Data" (KDD) process (Han & Kamber, 2006). The KDD process consists of data collection, data integration, data transformation, and visualization. Figure 1.2 depicts the process of KDD. Data mining systems are designed and equipped with a wide range of approaches that meet the needs of users to a considerable extent.

1.1 **Utility Itemset Mining**

Some basic definitions are as follows:

"Utility of an Item"

The utility of an item or itemset is defined as the product of internal utility and external utility.

For example, the utility of item {M} in T1 is 6. In T1, the utility for an itemset {M, N} is equal to the total of the utilities of these items, which is 14.

"Utility of an Item in the Dataset"

The utility of an itemset in the database is defined as the sum of all the utilities of that itemset from all the transactions having the said itemset.

For example, the utility for an itemset {M, N} is the sum of utilities of {M, N} from transactions T1 and T3, which equals 14+36 = 50.

1.2 Utility Itemset Mining for Big Data

"Lots of information and a lot of noise. To hear the signal between the noises is the key."

There has been a dramatic shift in intelligent

ISSN: 2366-1313

data processing with the expanded capacity of modern applications to produce and store huge quantities of data. Almost everything we are doing today leaves a digital trace, which can be analyzed and used by data scientists. Activities like playing an online game or browsing a shopping website generate and collect the data. Our smart phones collect data every second on what and how we use it. Big data is an immense amount of data with a high velocity and variety. Big data mining has a wide spectrum of uses in various sectors like health care, computational biology, detection of cyber-crime, web mining, analysis of sentiments, decision making, weather forecasting, etc. Over 100 million genomes are expected to be sequenced as part of genomic studies by 2025. Efforts from both big pharma and national population genomics programs are already yielding massive amounts of data, which are only expected to grow in the future. This data, if properly analyzed and interpreted, has the potential to bring precision medicine into a new golden era, as represented in figure 2. Several government agencies have already initiated their projects to mine the hidden facts in this era of big data, seeing the huge potential that can be used as the major decision factors for various policies in the fields of health, education, sports, etc. (Wu et al., 2013), (Tsai et al., 2015), (Lin et al., 2015). Furthermore, the larger the data set, the greater the opportunity potential and the more effective the data-mining tool in extracting meaningful information.

Figure 2 : Genomics with big data leads towards precision public health

1.3 Characteristics of Big Data

Why these large data sets are called “Big Data”? The answer can be given from the definition of distinctive Vs, which was introduced by Gartner analyst Doug Laney in 2001 (Laney, 2001). These characteristics are discussed here in brief.

Volume

Volume refers to the enormous amounts of data generated every second from various sources. Sensors acquire gigabytes of data every day as a result of device downsizing and the Internet of Things (IoT). Every hour, over 30,000 hours of video are uploaded to YouTube and 6,500 tweets are tweeted every minute. The amount of data available is increasing at an alarming rate.

Variety

Variety refers to a large number of data sources and forms, both structured and unstructured. Data is generated from heterogeneous sources and in various forms. It

ISSN: 2366-1313

could take the form of tweets, emails, forms, images, videos, audio files, transactions, and so on. There is no defined data structure. It might be structured as tables, semi- structured as XML, or unstructured as images.

Velocity

The pace at which the massive amounts of data are generated, acquired, and processed to meet the demands determines the true power of the data. Compared to the other factors, velocity is critical; it's pointless to pay big bucks and have to stand in line for data. Thus, one of the most essential characteristics of big data is the capacity to supply data on demand at a faster pace.

2. LITERATURE REVIEW

Several pattern-mining techniques have been suggested in the literature. Itemsets, sequences, and graphs are some of the patterns (Agrawal & Srikant, 1995), (Elseidy et al., 2014), which have been the focus of interest. This chapter reviews studies on the most famous category, high utility itemsets (Yao et al., 2006). Mining High Utility Itemsets (HUIs) can be considered as a refinement of the problem of frequent itemset mining where the database is composed of items having a factor associated with them, indicating their significance. This general formulation of the problem allows for the modeling of a broad spectrum of applications, such as identifying all itemsets in a transaction database that generate a high profit, discovering sets of web pages on which people spend a significant period of time, or locating all sequential patterns, as in classical frequent pattern mining. This chapter presents an in-depth review of the topic of High Utility Itemset Mining (HUIM) and serves as an overview as well as a guideline to recent achievements and research prospects. Section 2.2 and section 2.3 discuss the core concepts of association rule mining, itemset mining, frequent itemset mining and high utility itemset mining. A review of various high utility itemset mining techniques for big data is presented in section 2.4. Various loopholes and research opportunities in the sphere of HUI mining are also discussed in this section. The chapter concludes with the identification of research gaps.

(Agrawal et al., 1993) introduced the thought of itemsets and association rule mining. The method of finding itemsets in a database is known as “itemset mining”. It could be a group of often-occurring items, a rare set of items, items having negative correlations, or items with a specific interest aspect. The Apriori technique has been proposed as a method for extracting frequently occurring itemsets from a transactional database. Many data structures and techniques (Agrawal & Srikant, 1994), (Zaki, 2000), (Per et al., 2001) have since been proposed to mine frequent and high-utility itemsets from a transaction database. These data structures are intended to cut down on overall execution time, candidate itemsets investigated throughout the search

ISSN: 2366-1313

process, and memory requirements.

3. PROPOSEDTECHNIQUE(S) OF HIGH UTILITY ITEMSET MINING FOR BIG DATA

3.1 Proposed Technique - Absolute High Utility Itemset Miner (AHUIM)

This section introduces the proposed technique, Absolute High Utility Itemset Miner, or AHUIM, which extends the state-of-the-art "Efficient high-utility Itemset Mining" (EFIM) method. The proposed approach works in a distributed context by modeling EFIM in a parallel framework. Enormous datasets or big data can be processed quickly and efficiently by employing the "divide and conquer" practice. AHUIM makes use of the Apache Spark framework, with RDD as the data storage format. The data is partitioned across multiple nodes, each of which handles the task in its own way. The technique is discussed and assessed in the following sub-sections.

3.1.1 Phases of AHUIM

The phases of the technique AHUIM are discussed as follows:

The transactional data is distributed evenly in the form of data blocks among the working nodes of the cluster. Different working nodes carry out the task simultaneously. This is accomplished using the *flatmap* and *reduce by key* functions. A value is flat mapped to zero or more key-value pairs using the *flatmap* function.

An itemset F is taken as the current itemset, which is initially empty. For each item, a local utility list is created based on the quantity and profit factor. By creating pairs of *<item, utility>*, each node processes the data that has been allotted to it. The initial local utilities or TWU values of 1-itemsets are used to compute the following items of F (the items that should be considered in extension of F) by comparing them with the minimum utility threshold. These items of F are then arranged with their sorted TWU values. And the items that are not a part of the following items of F (i.e. the unpromising items based on the TWU values) are discarded here only as they cannot be a part of any larger HUI set. The dataset is then reviewed by arranging all the remaining items of a transaction in order of their rising values of TWU. Any transaction with no outstanding items is also eliminated here. As a result, the entire dataset is a revised version of the original, with transactions containing items with increasing TWU values.

The RDD functionality provided by the Spark system is used to read this updated dataset from the main memory. The algorithm then calculates the sub tree utility and succeeding items for all the items in F. The sub tree utility is used to prune the unnecessary nodes, which saves the visiting time of these nodes. The sub tree utility is calculated at the first level of the tree using the 1-HTWUIS to find the succeeding items. The search space is made up of these items and their respective sub trees,

ISSN: 2366-1313

which are evenly distributed throughout the nodes.

Based on the search space, transaction data for each node is also obtained. Each node then generates HUI from the allocated search space using the **Search** procedure.

Algorithm 1: Absolute_HUIM

Input: DS - Transactional dataset, I - List of items, Th - Minimum utility threshold value

Output: Itemsets with high utility

F = empty itemset

Compute locU(F, I_i) ∀ items I_i ∈ I by DS scan

Compute fi(F) = {I_i | I_i ∈ I ∧ locU(F, I_i) ≥ Th}

For fi (F), arrange according to increasing TWU values

Prune items I_i ∉ fi (F)

Remove Null transactions

Sort the remaining transactions by ~

Compute the SU (F, I_i) for every item I ∈ fi (F) by DS scan

Succeeding_items for F, si (F) = {I_i ∈ fi (F) ∧ SU (F, I_i) ≥ Th}

Return **Search** (F, DS, si (F), fi (F), Th)

3.2 Overall Flow of AHUIM

The proposed method AHUIM works in a distributed framework. It takes the transactional dataset and the minimum utility threshold as input. The dataset is partitioned equally among the working nodes of the cluster. The nodes compute the local utility of the items, which is initially equal to the TWU values of the items, using the flatMap and reduce By Key functions. Any item with a TWU value less than the threshold is dropped here. If there is any empty transaction after the deletion of items, it is also clipped from the database.

Transactions are now sorted, and items are arranged accordingly. The database is now a modified version of the original dataset. Further, sub tree utility is calculated from the modified dataset using the TWUs of 1-HTWUIs.

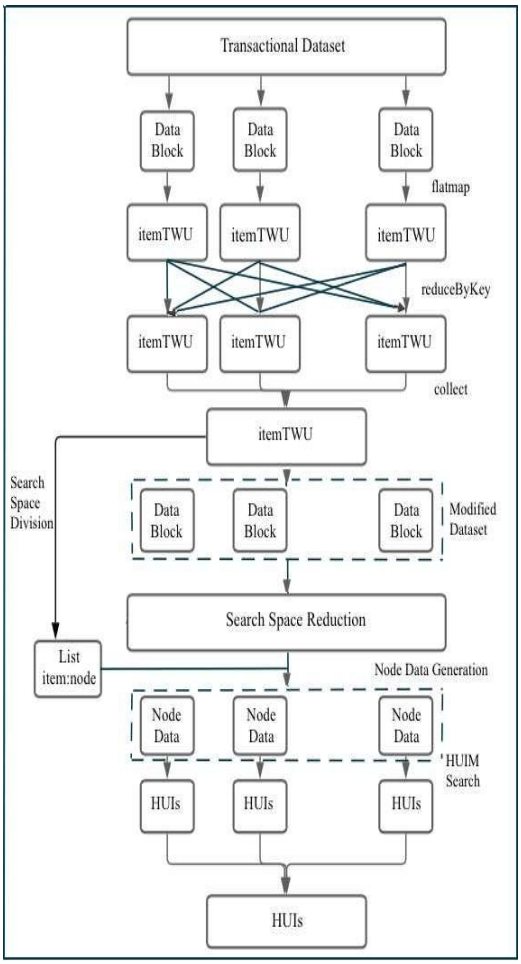

Figure3: Flow graph of AHUIM

The sub tree utility is used to prune the unnecessary nodes, which saves the visiting time of these nodes. The algorithm then calculates the succeeding items for all the items to be explored. The search space is made up of these items and their respective sub trees.

4. PERFORMANCE COMPARISON OF PROPOSED TECHNIQUE(S) WITH EXISTING HIGH UTILITY ITEMSET MINING TECHNIQUES

Utility Itemset (HUI) mining is a fascinating

ISSN: 2366-1313

topic in data analytics. It seeks itemsets with the lowest or highest utility offered by the user with respect to the purpose. In utility mining, each object is connected to a factor that reflects its importance, such as price, number, size, interestingness, profit, or any other piece of information depending on the desire of the user. The task is to identify the itemsets with the specified (low or high) utility value. High utility itemset mining is a key activity in the realm of big data, with several applications in a wide range of industries.

There have been very few high utility itemset-mining techniques (HUIM) developed for huge datasets. Among these are DTWU-Mining, EFIM-Par, PHUI-Miner, BigHUSP, and P-FHM+. In this research work, two techniques for enhancing mining efficiency are proposed: Absolute High Utility Itemset Miner (AHUIM) and Enhanced Absolute High Utility Itemset Miner (EAHUIM) for mining high utilityitemsets from big data. In this chapter, the performance of the proposed techniques AHUIM and EAHUIM is evaluated against that of other standard techniques.

4.1 Performance Comparison of AHUIM with Existing Approaches

The technique AHUIM is compared with other state-of-the-art HUIM techniques in this section. It is first compared with a non-distributed approach using small datasets to analyze the difference between the distributed and non-distributed frameworks. AHUIM is then compared with distributed approaches on large datasets.

4.1.1 AHUIM vs. Non-Distributed Algorithm

The proposed AHUIM technique is compared with the Two-Phase algorithm (Liu et al., 2005), which is the most widely used algorithm for non-distributed systems. Two-Phase is a complete algorithm, which uses a candidate generation approach to mine the HUIs. The 1-itemset is used to generate a candidate 2-itemset, which is then used to obtain a candidate 3-itemset, and so on until the longest-itemset is discovered. The candidate-itemsets are then assessed to find the high utility itemsets. The comparison of AHUIM with Two-Phase is done on the parameters of execution time, scalability, accuracy, and stability. The experiments are performed on real datasets.

4.2 Experimental Setup

Two-Phase is a non-distributed algorithm widely used for small datasets. It is used to evaluate the performance of AHUIM. Both the algorithms have been written in Python using the Spyder4 IDE. A Spark cluster with one master node and six working nodes is built for the distributed system using Apache Spark 3.0. The system used for execution has 32 GB of RAM and two Intel® Xeon® CPU E5-2620 processors with six cores each running at 2.00 GHz. The operating system is Windows 10.

ISSN: 2366-1313

Datasets

The experiments are carried out on three real-world datasets: Chess, Connect, and Mushroom. The datasets are available on the UCI repository. SPMF, a data-mining library, keeps these datasets in a format that is suitable for data mining techniques. Chess is a dataset with 3196 transactions and 75 unique items representing different game movements. Connect is also a dataset for a game with 67557 transactions and 129 different items. Mushroom is a sparse dataset with 8416 transactions and 119 unique items for various mushroom types. The attributes of the dataset are listed in table 1. The #Transactions, #Items, and #AverageItems are the total number of transactions in the dataset, the number of unique items, and the average number of items per transaction, respectively. The UCI repository contains detailed information about these datasets.

Table1: Various datasets for experiments (Source: SPMFLibrary)

Dataset	#Transactions	#Items	#AverageItems
Chess	3196	75	37
Connect	67557	129	43
Mushroom	8416	119	23

Performance Evaluation

In this section, the technique AHUIM is assessed on small datasets for various parameters. For every threshold value, experiments are conducted five times, and the average values are considered.

Comparison of Execution Time

The techniques AHUIM and Two-Phase are compared on three datasets: Chess, Connect, and Mushroom for the running time. Since the technique AHUIM works in distributed mode with more than one working node and Two-Phase is a non-distributed technique, AHUIM performs much better than Two-Phase.

Table2: Execution Time (seconds) of AHUIM and Two-Phase for different thresholds

Dataset	AHUIM			Two-Phase		
	2 %	3 %	4 %	2 %	3 %	4 %
Chess	1.844	1.78	1.32	15.68	13.57	11.54
Connect	6.65	5.44	5.2	45.64	43.68	41.46
Mushroom	4.238	4.62	3.98	38.86	34.36	29.46

The execution times of the techniques AHUIM and Two-Phase can be seen in table 2. Figure 4 (a) represents the execution time results for Chess dataset; figure 4.(b) Represents the execution time results for Connect, and figure 4(c) represents the result of execution time for the Mushroom dataset.

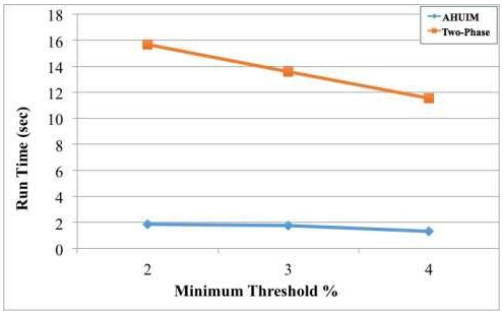

Figure 4(a): Execution time of AHUIM and Two-Phase for Chess

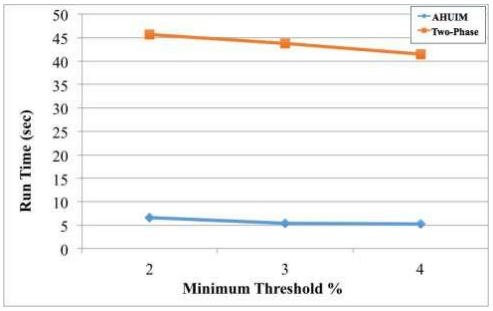

Figure 4(b): Execution time of AHUIM and Two-Phase for Connect

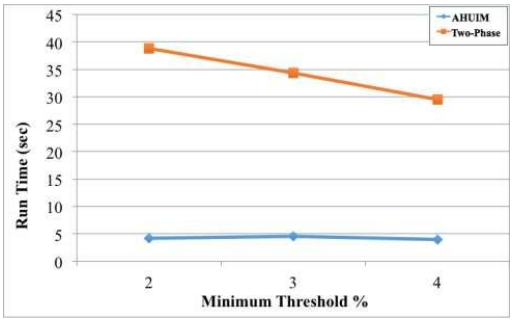

Table 3: Execution time for various sizes of Chess datasets

Figure4(c): Execution time of AHUIM and Two-Phase for Mushroom

Dataset	Technique	
	AHUIM	Two-Phase
Chess	1.844	15.68
Chess10x	13.423	197.86
Chess20x	24.54	486.92
Chess30x	52.546	893.53

4.3 Comparison of Scalability

The property of an algorithm to preserve performance as the workload grows is referred to as "scalability." To investigate the scalability of the techniques, the data size is increased and the performance of the techniques is noted. It can be seen from table3 (considering the Chess dataset) that AHUIM performs well with the increased data size and the execution time surges approximately linearly (Threshold = 2%).

4.4 AHUIM vs. Distributed Algorithms

In this section, AHUIM is compared with standard distributed techniques like PHUI-Miner (Chen, 2015) and EFIM-Par (Tamrakar, 2017). PHUI-Miner was introduced in 2016 for Big Data. It is a distributed approach with improved methods for load balancing. Sampling and compression techniques are used to minimize the search space. Although PHUI-Miner and other variants mine data

from big datasets, they only provide approximate results due to a trade-off between accuracy and efficiency. EFIM-Par was proposed in 2017 for mining big data with effective pruning strategies of "sub tree utilities" and "local utilities". During the literature survey, it is found that both of these techniques are better than other benchmark algorithms. AHUIM is competed for execution time, scalability, memory usage, stability and accuracy.

5. Traffic Jams Detection and Congestion Avoidance in Smart City Using Parallel K-Means Clustering Algorithm

5.1 Parallel K-Means Clustering Algorithm

In this section, we present how to adapt K-means in the parallel environment for big data. Here, K-Means algorithm is used for traffic control applications in smart city. The k-means algorithm takes the input data set D and parameter K, and then divides a data set D of n objects into k groups. This partition depends upon the similarity measure so that the resulting intra cluster similarity is high but the inter cluster similarity is low. Cluster similarity is measured regarding the mean value of the objects in a cluster, which can be showed as the clusters mean. The k- means procedure works as follows. First, it randomly chooses k objects, each of which initially defined as a cluster mean or center. For each of the remaining objects, an object is moved to the cluster to which it is the most similar, based on the similarity measure which is the distance between the item and the cluster average. It then calculates the new mean for each cluster. This process repeats until no change in the mean values in the clusters.

To get the advantages of the high performance of in-memory computing in the traffic field in smart city we have proposed K-means model whose work flow is represented in Figure 5. According to the pervious diagram, the data is spitted into m segment and sent to the in-memory computing cluster. At the same time the k number of clusters and their centers initial values delivered to the in-memory computing cluster. After that, each point in the data set is classified as one of the output clusters states. This output will be sorted and shuffled. The system will check whether the number of iterations is reached or the convergence condition is satisfied if yes then output is obtained otherwise it will go for next iteration.

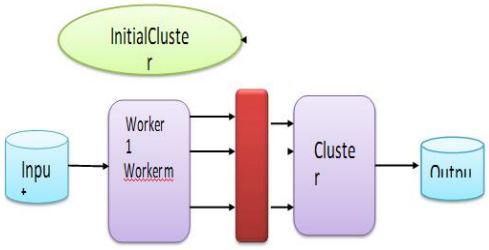

FIGURE5: Proposed Traffic Management Model.

5.2 Experimental Results and Discussions

This analysis is based on the real traffic data for the Aarhus city in the CityPulse project [107]. A collection of datasets of vehicle traffic, observed between two points for a set duration of time over a period of 6 months (449 observation points in total).In this study K-Means clustering algorithm (k = 3) have been applied to all location in order to build a real model of the traffic condition all over the day in order to help the people in this city to get use of the smart traffic issue. In this study there are three classes red indicates heavy traffic area, green indicates traffic free area and yellow indicates the area with moderate traffic but a high chance of getting jammed. The clusters are monitored continuously and the results are updated for every 5 min and the summary data is collected for every 2 h. The use of these results aims to make the transportation system inside the Arhus city smarter in order to help people to save time and effort for searching about alternative ways in the time of traffic jam happen as the model presents a full view of the traffic condition for the hall city every five minutes. A sample of results obtained from the model for certain location on the Aarhus city from 11:30 am to 7:30 pm is as it is shown in Figure 6 the traffic conditions from 11:30 am to 1:30 pm was moderate only one small traffic jam happened. From 1:30 pm to 3:30 pm was so heavy a lot of traffic jams was recorded which means that the people have to choose another route in this duration of day. After that, from 3:30 pm to 5:30 pm the traffic condition in this location becomes good. Finally, from 5:30 pm to 7:30 pm the road is free and traffic condition was excellent in this location.

By this way the proposed model give the traffic management good option to do good control of the traffic conditions in the city. They could give the people in the city a complete picture of the traffic conditions inside the city. Using this model, help people to choose optimal route of their journey at any time. Traffic condition in different locations in the Aaruthu city from 1:30 pm to 3:30 pm is given in Figure 6.3. The duration from 1:30 pm to 3:30 pm is one of traffic jams times in the most of cities. By using the proposed model alternative routes will be available to the people to choose best route at this time. As

it is shown in Figure 6.3, traffic condition from different six locations inside the Aaruthu city from 1:30 pm to 3:30 pm is presented. It is clear that the location (a) is the heaviest road. It contains the maximum number of vehicles at this duration so it is suggested to the people to avoid this road at this time. Locations b and c have a lot of traffic jam at this time of day.

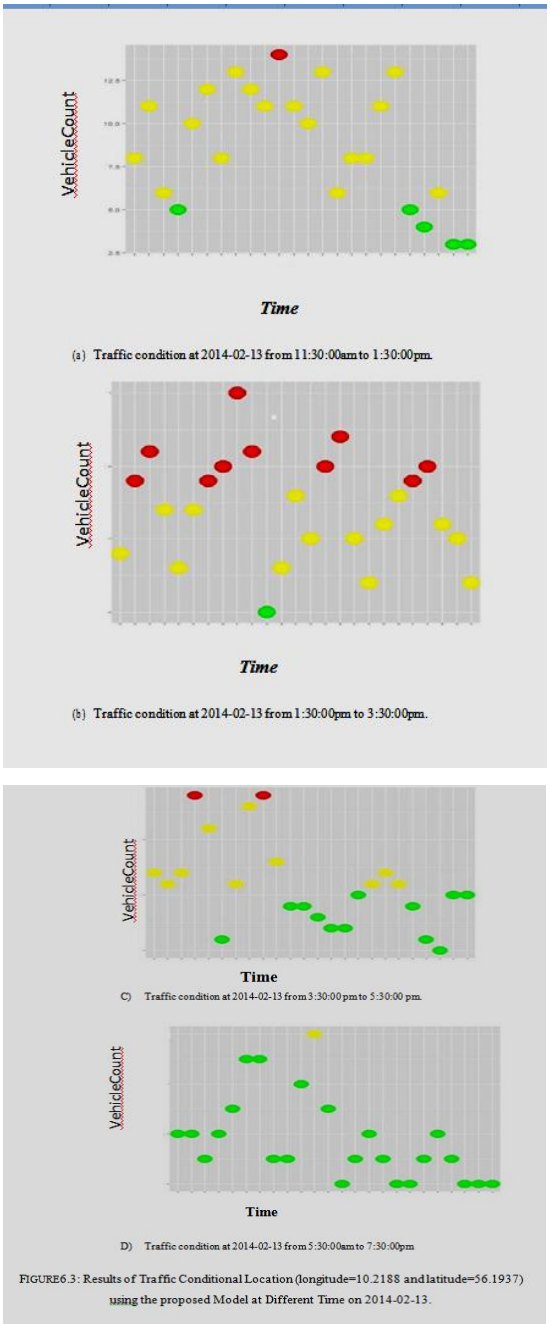

FIGURE6.3: Results of Traffic Conditional Location (longitude=10.2188 and latitude=56.1937) using the proposed Model at Different Time on 2014-02-13.

CONCLUSION

The main focus of this thesis is on developing method(s) for extracting patterns of itemsets from large datasets. Apache Spark, which is regarded as the most capable platform for parallel processing, is used for analysis. A spark cluster is being employed with master and slave nodes. A novel technique, Absolute High Utility Itemset Miner (AHUIM), is proposed by amending the structure of the traditional EFIM technique. The technique works on a distributed framework to process the huge datasets, as stand-alone systems are not capable of processing these datasets. The AHUIM technique works by distributing the data to various nodes, and these nodes then mine the itemsets simultaneously. The Map-Reduce framework is used for this purpose. To reduce the search space, two novel space pruning strategies are proposed, called absolute local utility and absolute sub tree utility. The performance of the technique is compared to other state-of-the-art techniques such as Two-Phase, EFIM- Par, and PHUI-Miner. AHUIM outperforms these techniques in terms of execution time, memory requirements, and scalability. The accuracy of the technique is compared to ground truth data, and it is found that AHUIM efficiently finds the itemsets with good accuracy.

REFERENCES

1) Caesar Wu, Rajkumar Buyya, and Kotagiri Ramamohanarao. Big data analytics= ma- chine learning+ cloud computing. arXiv preprint arXiv:1601.03115, 2016.
2) Rajkumar Buyya, Rodrigo N Calheiros, and Amir Vahid Dastjerdi. Big Data: Principles and Paradigms. Morgan Kaufmann, 2016.
3) Hsinchun Chen, Roger HL Chiang, and Veda C Storey. Business intelligence and analyt- ics: From big data to big impact. MIS quarterly, 36(4), 2012.
4) Peter Lake and Robert Drake. Information systems management in the big data era. Springer, 2014.
5) Timothy Paul Smith. How Big is Big and how Small is Small: The Sizes of Everything and why. OUP Oxford, 2013.
6) Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of big data on cloud computing: Review and open research issues. Information Systems, 47:98–115, 2015.
7) Gil Press. A very short history of big data. Forbes Tech Magazine, May, 9, 2013.
8) Fremont Rider et al. scholar and the future of the research library. 1944.
9) Frank J Ohlhorst. Big data analytics: turning big data into big money. John Wiley & Sons, 2012.
10) Aggarwal, C. C., Bhuiyan, M. A. & Al Hasan, M. (2014). Frequent pattern mining.
11) Springer. https://doi.org/10.1007 /978-3-319-07821-2
12) Aggarwal, C.C. (2016). An Introduction to recommender systems. in Recommender Systems, (pp. 1-28). Springer, Cham, Switzerland. https:// doi.org/ 10.1007/ 978-3-319-29659-3_1
13) Agrawal R., Imielinski T. & Swami A. (1993). Mining association rules between sets of items in large databases. In International Conference of Management of Data (pp. 207-216). Washington, DC, http://www.rakesh.agrawal-family.com/ papers/ sigmod93assoc.pdf
14) Agrawal R. & R. Srikant. (1994). Fast Algorithm for Mining Association Rules. In International conference on Very Large Data Bases (pp. 487-499), Santiago, Chile. http:// www.cse.msu.edu/ ~cse960/ Papers/ MiningAssoc-AgrawalAS-VLDB94.pdf
15) Agrawal, R. & Srikant, R. (1995). Mining sequential patterns. In Eleventh International Conference on Data Engineering, (pp. 3–14).

https://doi.org/10.1109/ICDE.1995.380415.

16) Agrawal, R., Gehrke, J., Gunopulos, D. & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In ACM SIGMOD International Conference on Management of Data, vol 27 (2), pp. 94–105. https://doi.org/10.1145/276305.276314

17) Ahmed C, Tanveer C, Jeong B & Lee Y. (2009). Efficient tree structures for high utility pattern mining in incremental databases. IEEE Transactions on Knowledge and Data Engineering 21(12), 1708-1721, https:// doi.org/ 10.1109/TKDE.2009.46