Efficient Tree Based Distributed Data Mining Algorithms for mining Frequent Patterns

T.SathishKumar
Asst.Prof ,
Department of IT
VCET,Erode,India

V.Kavitha
Asst.Prof,
Department of IT
VCET,Erode,India.

Dr.T.Ravichandran
Principal,
HIT,
Coimbatore,India

ABSTRACT
Advancements in the field of wired and wireless network environments have paved route to the advent of many dynamic distributed computing environments. These environments have diverged computing resources and multiple heterogeneous sources of data. Most mining algorithms are designed to mine rules from monolithic non-distributed databases. Even algorithms exclusively designed to operate on distributed databases normally download the relevant data to a centralized location and then perform the data mining operations. This centralized approach does not work well in many of the distributed, ubiquitous, privacy sensitive data mining applications, which opened a new area of research Distributed Data Mining (DDM) under the data mining domain. Out of various methods employed to mine frequent Itemsets, tree based methodology proves some efficiency in distributed environment. So in this paper we study a set of tree based algorithms [DTFIM, PP, LFP and PP] to mine frequent pattern in distributed environment.

General Terms
Tree based frequent pattern mining.

Keywords
Tree based algorithms, Distributed data mining, Mining Frequent patterns.

1. INTRODUCTION
1.1 Overview and motivation
Advancements in the field of wired and wireless network environments have paved route to the advent of many dynamic distributed computing environments. These environments have diverged computing resources and multiple heterogeneous sources of data. Most mining algorithms are designed to mine rules from monolithic non-distributed databases. Even algorithms exclusively designed to operate on distributed databases normally download the relevant data to a centralized location and then perform the data mining operations. This centralized approach does not work well in many of the distributed, ubiquitous, privacy sensitive data mining applications, which opened a new area of research Distributed Data Mining (DDM) under the data mining domains.

In datamining one of the key area is finding frequent patterns which plays an essential role in finding interesting patterns from databases such as association rules, correlations, episodes, clusters, sequences, classification and many more. Association rule mining (ARM) one of the important areas in Market Basket Analysis, Agrawal, Imielinski, and Swami (1993)[1] first introduced the problem of mining association rules from transaction data . The need for ARM was to analyze the business transactional data, which is to identify the consumer behaviour in terms of product purchase / service received. In order to identify how often the items are bought together can be well described using association rules. For example an association rule "milk $\Rightarrow$ bread (80%)" states that eight out of ten consumers that bought milk also bought bread. We can use this to improvise the marketing decisions based on the current trends and the mined rules. A large number of FP mining algorithms focussing upon parallel and distributed environment have already been proposed. Out of this FP-tree based parallelisation algorithm has been proved to be more efficient, when compared to the other approaches.

As mentioned earlier most FP mining methods typically assume that the data is centralized, memory-resident, and static. Whereas, in actual environment, it is proven that methods of data mining using centralized concepts is less valid. Therefore, researchers focused on large-scale parallel and distributed FP mining algorithms [5,6,7,8,14,21,27] The objective of this paper is to identify and compare the efficiencies of those algorithms that employ tree based methodology in mining frequent from distributed databases.

2. MINING FREQUENT PATTERNS
2.1 Need for Mining Frequent Patterns
Mining Frequent patterns is a sub problem to the problem of mining association rules, which is defined by, for a given transaction database T, an association rule is an expression of the form $X \Rightarrow Y$, where X and Y are subsets of A and $X \Rightarrow Y$ holds with Confidence τ, if τ% of transactions in D that support X also support Y. The rule $X \Rightarrow Y$ has support σ in the transaction set T if σ% of transactions in T support $X \cup Y$. This means that a transaction of the database which contains X tends to contain Y. Given a set of transactions, T, the problem of mining association rules is to deliver all rules that have support and confidence greater than or equal to the user-specified minimum support and minimum confidence respectively.

The problem of mining association rules are then decomposed into two sub problems:

- Find all patterns whose support is greater than the user-specified minimum support, σ. Such patterns are called frequent patterns.
- Use the frequent patterns to generate the desired rules. The general idea is that if, say KLMN and KL are frequent patterns, then we can determine if the rule $KL \Rightarrow MN$ holds by checking the following inequality

$$\frac{s(\{K,L,M,N\})}{s(\{K,L\})} \geq \tau,$$

Where s(X) is the support of X in T.

Out of the two sup problems mentioned above much research works have been focused on the first sub problem, as the database is accessed in this part of the computation.

2.2 Frequent Itemsets

Let T be the transaction database and σ be the user-specified minimum support. An itemset X ⊆ A is said to be frequent itemset in T with respect to σ, if

$S(X)_r \geq \sigma$

Consider the following set of transactions in a bookshop, in the first transaction, purchases are made of books on Compiler Architecture, Distributed Databases, Theory of Computations, Client Server and System Software; let the subjects be denoted by CA, DD, TOC, CS and SS, respectively. Thus the set of transactions are described as follows :

t_1 := { SS, CA, TOC, CS }

t_2 := { CA, DD, CS }

t_3 := { SS, CA, TOC, CS }

t_4 := { SS, CA, DD, CS }

t_5 := { SS, CA, DD, TOC, CS }

t_6 := { CA, DD, TOC}

if support count is assumed to be σ = 50%, then { SS, CA, TOC} is a frequent set as it is supported by at least 3 out of 6 transactions. It can be found that any subset of this set is also frequent set. On the other hand {SS, CA, DD} is not a frequent itemset and hence, no set which properly contains this set is a frequent set.

Interesting Properties of frequent sets for a given transaction, T:

- **Downward Closure Property** Any subset of a frequent set is a frequent set.
- **Upward Closure Property** Any superset of an infrequent set is an in frequent set.

2.3 Basic Mining Methodologies

2.3.1 Apriori

- Major idea (Agrawal and Srikant 1994)[2]
  - A subset of a frequent itemset must be frequent.
- Core of Apriori algorithm
  - Use frequent (k – 1)-itemsets to generate candidate frequent k-itemsets.
  - Use database scan and pattern matching to collect counts for the candidate itemsets.
- Methodology
  - Mining frequent itemsets in a large transaction database using scalable methods proves to be highly challenging because each transaction databases contains large number of distinct single items also the combination of these single items form large itemsets. Agrawal and Srikant (1994) [2] observed an interesting *downward closure* property, called Apriori, among frequent *k* itemsets: *A k-itemset is frequent only if all of its sub-itemsets are frequent.* The essence of the Apriori (Agrawal and Srikant 1994)[2] and its alternative (Mannila et al. 1994)[21] is that to find frequent 1-itemsets by scanning the database once, then using frequent 1-itemsets the frequent 2-itemsets are generated after k+1 time of scanning the database frequent k-itemsets are generated.
- Improvements or extensions
  - Hashing (Park et al. 1995)[25],
  - Partitioning (Savasere et al. 1995)[26],
  - Sampling approach (Toivonen 1996)[28],
  - Incremental mining (Cheung et al. 1996)[7],…etc.
- Two-nontrivial costs
  - Generating a huge number of candidate sets.
    - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets
  - Repeatedly scanning the database and checking the candidates by pattern matching.
    - Needs (*n +1*) scans, *n* is the length of the longest pattern
- Real reason of Apriori Algorithm's failure
  - It lacks of good database processing method**.**

ALGORITHM Apriori

Input: D, σ
Output: F(D, σ)

1: *C1 := {{i} | i ∈ I}*
2: *k :=* 1
3: **while** *Ck* ≠ {} **do**
4: // Compute the supports of all candidate itemsets
5: **for all** transactions (*tid* , *I*) ∈ *D* **do**
6: **for all** candidate itemsets X ∈ Ck **do**
7: **if** *X* ⊆ *I* **then**
8: X.support++
9: **end if**
10: **end for**
11: **end for**
12: // Extract all frequent itemsets
13: Fk := {X | X.support ≥ σ}
14: // Generate new candidate itemsets
15: **for all** X, Y ∈ Fk,X[i] = Y [i] for 1 ≤ i ≤ k − 1, and X[k] < Y [k] do
16: I = X ∪{Y [k]}
17: **if** ∀ J ⊂ I, |J| = k : J ∈ Fk then
18: Ck+1 := Ck+1 ∪ I
19: **end if**
20: **end for**
21: k++
22: **end while**

2.3.2 ECLAT

- Major idea (Zaki (1997))[18].
  - Mining frequent itemsets using vertical data format
- Core of Eclat algorithm
  - The frequent itemsets are determined using simple tid-list intersections in a depth-first graph.
- Methodology
  - Transaction Id set (TID_set) for each frequent item is generated during the first scan of the database. Éclat too presumes Apriori property as such it extracts (K+1)-itemsets from the previously generated K-itemsets, with a depth-first computation order similar to FP-growth (Han et al. 2000)[12]. The (k+1) itemsets are generated from the K – itemsets by intersecting their TID_sets. This process is repeated until no new frequent itemsets are generated.
- Improvements or extensions
  - Eclat Z, (Laszlo Szathmary et al (2008)) [19]
  - UEclat (Laila A. Abd-Elmegid et al(2010)) [18]
  - Eclat VJ (Minho Kim et al (2003)) [22]
- ECLAT is very efficient for large itemsets but less efficient for small ones.

ALGORITHM Eclat

Input: D, σ, I ⊆ I
Output: F[I](D, σ)
1: *F[I]* := {}
2: **for all** *i 2 I* occurring in D **do**
3: *F[I] := F[I]*$\cup$ {I $\cup$ {i}}
4: // Create D ^{i}
5: $D^{i} := \{\}$
6: **for all** j $\in$ I occurring in D such that j > i do
7: C := cover({i})$\cap$ cover({j})
8: **if** |C| $\geq \sigma$**then**
9: $D^{i} := D^{i} \cup \{(j,C)\}$
10: **end if**
11: **end for**
12: // Depth-first recursion
13: Compute $F[I \cup \{i\}](D^{i}, \sigma)$
14: $F[I] := F[I] \cup F[I \cup \{i\}]$
15: **end for**

2.3.3 FP-Growth

- Major idea (Han et al. (2000))[12]
  - a combination of the vertical and horizontal database layout to store the database in main memory.
- Core of FP-Growth algorithm
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones.
  - No candidate generation: only need sub-database test.
- Methodology
  FP-growth employs *divide-and-conquer* method. A list of frequent itemsets in descending order is generated during the first scan of the database. By using this list , a compressed data representation in tree form named Frequent Pattern tree, or *FP-tree is generated*, which maintains the itemset association information. By assuming the frequent length-1 pattern as the initial suffix pattern the FP-tree is mined, constructing its conditional pattern base, which is a sub database that contains co-occurring prefix paths with suffix pattern in the FP-tree, next the conditional FP-tree is generated and mining is recursively performed on that tree. On concatenating the suffix patterns and the frequent patterns the resultant pattern growth is achieved.
- Improvements or extensions
  - H-Mine, by Pei et al. (2001)[16]
  - Pattern-growth mining, by Liu et al. (2002; 2003)[18]
  - Prefix-tree-structure, by Grahne and Zhu (2003)[11]
- Two Nontrivial cost
  - Generating a huge number of candidate sets.
  - Repeatedly scanning the database and checking the candidates by pattern matching.

ALGORITHM FP-Growth

Input: D, σ, I $\subseteq$ I
Output: F[I](D, σ)
1: F *[I]* := {}
2: **for all** i $\in$ I occurring in D do
3: F [I] := F[I] $\cup$ {I $\cup$ {i}}
4: // Create D^{i}
5: D^{i} := {}
6: H := {}
7: **for all** j $\in$ I occurring in D such that j > i do
8: **if** *support*(I $\cup${i, j}) $\geq \sigma$ then
9: H := H $\cup$ {j}
10: **end if**
11: **end for**
12: **for all** (tid ,X) $\in$ D with i $\in$ X do
13: D^{i} := D^{i} [{(tid,X $\cap$ H)}
14: **end for**
15: // Depth-first recursion
16: Compute F[I $\cup${i}](D^{i}, σ)
17: F[I] := F[I] $\cup$ F[I $\cup${i}]

Analysing all the three basic mining methodologies of frequent pattern mining, the number of candidate sets generated for mining Frequent patterns was very large in Apriori as compared to FP-growth & Eclat, where as the FP-growth algorithm too generates much candidate sets, which is not the case in éclat , but it proves efficient only in very large databases. Altogether the FP-tree based mining method proves to be more reliable as it was proven in later extensions and improvement works.

3 MINING FREQUENT PATTERNS FROM DISTRIBUTED & DYNAMIC DATABASES USING TREE DATA STRUCTURE

Recent advancements in the areas like networking and wireless technologies have led a new highly demandable source of data named Distributed and Dynamic databases. A more real time example of such a scenario can be best found in the repositories of multinational corporations that contain disjoint databases that are demographically separated. Also each such database is kept updated continuously for every corresponding transaction. The

differentials in the frequency of updating and its credentials are unique to each individual site. This environment provides a vast scope for researchers to research; one such area of research is to mine frequent patterns from distributed and dynamic databases.

Algorithms [5],[26],[21] developed over last decade for mining Frequent patterns trough parallel and distributed mining mainly finds its roots as Apriori based, and inherits the same performance bottlenecks of Apriori technique mentioned above. To overcome this bottleneck, parallelization of efficient FP-tree based FP-growth mining can be found in [6],[8],[15],[29],[9],[13], . The algorithms FP-Forest [13] , Parallel FP-tree (PFP-tree) [15] , Load Balancing FP-tree (LFP-tree) [29] , Multiple Local Parallel Trees (MLPT)[30] and Parallel Pattern Tree (PP-tree) [27] all employs the concept of dividing large databases into smaller parts that can be handled by individual node by using its available resources. The then individual nodes construct local FP-trees by mining its respective database twice. While mining each individual nodes transmit and share their respective frequent itemset count with other nodes. In distributed environment the major factor to be considered are load sharing and minimizing the number of transactions between the nodes. In respect of this we are analyzing the following algorithms.

3.1 Distributed Trie Based Frequent item set Mining

This algorithm DTFIM[4] , uses a trie based methodology in order to find out the frequent Item sets in the each site. All the K- frequent item set (C_k)is calculated at each site separately. Finally all the local K item sets are pruned in, by transferring the trie between the sites. the final K item set calculated in global is considered., only if the k item set satisfies the global support count.

In this DTFIM algorithm the number of transactions is minimized by transferring only the local C_k item sets between the sites. The problem is that the infrequent item set in a local site may be missed while calculating the global frequent item set but the difference in the distributed calculation of first pass of C_k item set is negligible compared to the speed obtained by data transferring between the sites and the speed up of the algorithm

3.2 Revised DTFIM

This algorithm is an optimization of the previous algorithm. This algorithm optimises by stopping the calculation of infrequent item sets. After every pass in a local site the trie is exchanged between the sites. If a frequent item set is infrequent in other sites then the corresponding item is removed from the frequent item set thereby reducing the time in the calculation of frequent item sets.

In this algorithm it tries to speed up the computation by reducing the frequent item set calculation from the first pass onwards, but this may have high impact on the final frequent item set . Since the item which is infrequent at an site may be frequent in other sites, All the local sites may not be synchronized, an local site may have to wait for the answer of other sites before it enters into the other pass. Even though it speedup the computation time , the time taken to complete the process may grow large if the synchronization is not good or when there is a slow site.

3.3 PP Tree

Tanbeer et.al [2010] [27]proposes an efficient method for finding the frequent pattern using parallel and distributed algorithm using PP Tree. This is an advanced application of FP tree in distributed environment. The large database is partitioned into non overlapping blocks and each block is assigned to individual node in a distributed environment there by load is shared equally in all the nodes. The Frequent patterns are constructed in five phases First phase each individual node scans the database once and generates a local PP through this it counts the support of each item distinctly. Second phase calculates the global count of each item. Third phase the local PP tree is reconstructed according to the global count of each item. Fourth phase potential global frequent item set is generated at each local site. Fifth phase the actual global frequent patterns are constructed.

All the nodes are considered to be identical, so the load sharing is achieved by partitioning equally. The frequent pattern identified will be same as that are generated homogenous algorithm. Since the all local PP tree is send to the master processor all the data items are included in frequent pattern generation. The data transfer overhead involved in this algorithm will be more compared to the DTIFM.

3.4 Load balancing Frequent pattern tree (LFP)

Kun-Ming Yu et.al [17] proposes an algorithm which shares the load between the nodes in the distributed environment. The database is equally partitioned and distributed to the nodes. A local header table is generated at each node (Slave node SN) which scan the database once and counts the occurrence of each item. The entire local table from the slave nodes are transmitted to the master node (MN). The master node prepares a global table. The global table has the number of occurrences of the all data items.

The global table is transmitted to all the nodes. FP tree is constructed in each slave node based upon the local database and the global databases. Each node of FP tree consists of item name, count and link. The item name is the item of the node represents, count represents the number of transaction occurred in the corresponding path. And link links to the next node. The depth and width of FP tree is calculated in each node. To avoid large database transfer SN preserves the data item in which loading is large. The load degree is decided at each node. Depending upon the load degree each SN is assigned a data item to mine. Then the FP tree is exchanged between the SN. Each node calculates frequent item sets in its FP tree. Finally MN collects all the frequent patterns.

3.5 Parallel FP-Growth (PFP)

Haoyuan Li et al (2008)[14] in their work proposed an algorithm to parallelize the FP-Growth algorithm on distributed machines. At the core this algorithm partitions computation in such a way that each individual machine executes an independent group of mining tasks, through this it eliminates the computational dependencies each machine and thereby reducing the communication between the machines. This algorithm targets to overcome and reduce the challenges such as Storage Overheads, Complexity in distributing the computation & Communication complexities in the FP-growth algorithm. The algorithm achieves this through five phases, during the first phase *Sharding* , the database is divided into equal parts and distributed into different machines, the second phase *Parallel Counting* ,counts the support values of all items that appear in

the database and stores it in list of frequent items, the third phase *Grouping Items*, divides the list of frequent items into groups, then each group is identified uniquely using unique id., the fourth phase *Parallel FP-Growth*, is the core phase which is sub divide into two sup-phases *Mapper & Reducer* phase, the Mapper phase identifies the group-dependent transactions using a mapper algorithm, followed by the reducer phase that generates FP-growth on the dependent transactions finally the *aggregating phase* aggregates the results arrived at the previous phase.

4 CONCLUSION & FUTURE WORK

Mining frequent patterns creates a boon for market research analysis to predict the customer behavioural patterns, since the source of data have been distributed and dynamic in nature the frequent pattern mining in such an environment proves to be a highly demanding researchable area. Existing algorithms concentrate more on distributing database to multiple machines by portioning it and mining in parallel. But in real world, the data source itself is distributed which requires a different approach rather than that of DDM. In our future research we propose an effective mining method to mine frequent patterns from the distributed data sources, with special emphasize on load sharing between different data source hosts.

5 REFERENCES

[1] R. Agrawal, T. Imielinski and A. N. Swami, 1993. "Mining association rules between sets of items in large databases", in *ACM SIGMOD Int. Conf. on Management of Data* pp. 207-16.

[2] R. Agrawal and R. Srikant, 1994. "Fast algorithms for mining association rules,", in *VLDB* pp. 487-99.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo 1996.”Fast discovery of association rules”, in Advances in KnowledgeDiscovery and Data Mining pages 307–328.

[4] E. Ansari, G.H. Dastghaibifard, M. Keshtkaran,” 2008. 19-21 March, 2008. DTFIM: Distributed Trie-based Frequent Itemset Mining”, Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol I IMECS.

[5] M.Z Ashrafi, D. Taniar and K. Smith, 2004. "ODAM: An optimized distributed association rule mining algorithm", IEEE Distributed Systems Online 1541-4922, 5 (3).

[6] G. Buehrer, S. Parthasarathy, S. Tatikonda, T. Kurc and J. Saltz,2007. "Toward terabyte pattern mining an architecture-conscious solution," in PPoPP, p. 2-12.

[7] D. W. Cheung, J. Han, V. T. Ng, and C. Y.Wong. 1996. “Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique”. In proceedings of 12th ICDE.

[8] S. Cong, J. Han, J. Hoeflinger and D. Padua, 2005. "A sampling-based framework for parallel data mining," in PPoPP , pp. 255-65

[9] D. Chen, C. Lai, W. Hu, W.G. Chen, Y. Zhang and W. Zheng, 2006. "Tree partition based parallel frequent pattern mining on shared memory systems," in IEEE Parallel and Distributed Processing Symposium.

[10] David Wai-Lok Cheung , Jiawei Han , Vincent Ng , C. Y. Wong, February 26-March 01. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique, Proceedings of the Twelfth International Conference on Data Engineering, p.106-114.

[11] G. Grahne and J. Zhu , May 2003. ”High performance mining of maximal frequent itemsets”, In *SIAM’03 Workshop on High Performance Data Mining: Pervasive and Data Stream Mining*.

[12] Han, J., Pei, J., and Yin, Y. 2000. Mining frequent patterns without candidate generation. In Proc. 2000 ACMSIGMOD Int. Conf. Management of Data (SIGMOD’00), Dallas, TX, pp. 1–12

[13] J. Hu and X. Yang-Li, 2008. "A fast parallel association rules mining algorithm based on FP-Forest," in 5th Int. Symposium on Neural Networks , pp. 40-9.

[14] Haoyuan Li,Yi Wang,Dong Zhang, Ming Zhang,Edward Chang 2008.”Pfp: parallel fp-growth for query recommendation Proceedings of the 2008 ACM conference on Recommender systems Pages: 107-114.

[15] A. Javed and A. Khokhar, 2004. "Frequent pattern mining on message passing multiprocessor systems," Distributed and Parallel Databases , vol. 16, pp. 321-34.

[16] Jian Pei,Jiawei Han ,Hongjun Lu ,Shojiro Nishio ,Shiwei Tang ,Dongquing Yang,” H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases”, First IEEE International Conference on Data Mining (ICDM'01)

[17] Kun-Ming Yu, Jiayi Zhou, and Wei Chen Hsiao , 2007. ”Load Balancing Approach Parallel Algorithm for Frequent Pattern Mining” V. Malyshkin (Ed.): PaCT 2007, LNCS 4671, pp. 623–631, 2007 Springer-Verlag Berlin Heidelberg .

[18] Laila A. Abd-Elmegid Mohamed E. El-Sharkawi Laila M. El-Fangary & Yehia K. Helmy May 2010. “Vertical Mining of Frequent Patterns from Uncertain Data” journal on Computer and Information Science Vol. 3, No. 2.

[19] Laszlo Szathmary, Petko Valtchev, Amedeo Napoli, and Robert Godin 2008. “An Efficient Hybrid Algorithm for Mining Frequent Closures and Generators “ CLA 2008, pp. 47–58, ISBN 978–80–274–2111–7, Palack´y University, Olomouc.

[20] J. Liu, Y. Pan, K. Wang, and J. Han, 2002. ” Mining frequent item sets by opportunistic projection”. In *SIGKDD*.

[21] Mannila, H.; Toivonen, H.; and Verkamo, 1994. A. I. Efficient algorithms for discovering associationrules. In AAAI Workshop on Knowledge Discovery in Databases (KDD 94) , 181 - 192.

[22] Minho Kim, Gye Hyung Kim and R.S. Ramakrishna 2003 .“A Virtual Join Algorithm for Fast Association Rule Mining “Intelligent Data Engineering and Automated Learning, Volume 2690/2003, 796-800, DOI: 10.1007/978-3-540-45080-1_108

[23] Mohammed J. Zaki. May/June 2000. Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering, 12(3):372-390.

[24] S. Orlando, P. Palmerini, R. Perego and F. Silvestri, 2003."An efficient parallel and distributed algorithm for counting frequent sets," in VECPAR , pp. 421-35.

[25] J.S. Park, M.-S. Chen, and P.S. Yu 1995. ”An effective hash based algorithm for mining association rules”, In Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, volume 24(2) of SIGMOD Record, pages 175–186. ACM Press.

[26] Savasere, E. Omiecinski and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," in VLDB , 1995, pp. 432-44.

[27] Tanbeer SK, Ahmed CF, Jeong B. 2009.” Parallel and Distributed Algorithms for Frequent Pattern Mining in Large Databases” IETE Tech Rev;26:55-65

[28] H. Toivonen, 1996. ”Sampling large databases for association rules”,in T.M. Vijayaraman, A.P. Buchmann, C. Mohan, and N.L. Sarda, editors, Proceedings 22nd International Conference on Very Large Data Bases, pages 134–145. Morgan Kaufmann.

[29] K.-M. Yu, J. Zhou and W. C. Hsiao, 2007. "Load balancing approach parallel algorithm for frequent pattern mining," in PaCT , pp. 623-31.

[30] O.R.Zaoane, M. El-Hajj and P. Lu, 2001. "Fast parallel association rule mining without candidacy generation," in IEEE Int. Conf. on Data Mining , pp. 665-8.