International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume: 3 | Issue: 2 | Jan-Feb 2019 Available Online: www.ijtsrd.com e-ISSN: 2456 - 6470

A Smart Association Rule Bit Vector Matrix for Mining Behavioral Patterns from Wireless Sensor Network

Uppada Venkateswara Rao[1], Dr. Somaraju Mouli[2]

[1]Final M.Tech Student, [2]Assistant professor, HOD
[1,2]Department of CSE, Sarada Institute of Science,
[1,2]Technology and Management (SISTAM), Srikakulam, Andhra Pradesh, India

ABSTRACT
Now a day's wireless sensor network interesting research area for discovering behavioral patterns wireless sensor network can be used for predicting the source of future events. By knowing the source of future event, we can detect the faulty nodes easily from the network. Behavioral patterns also can identify a set of temporally correlated sensors. This knowledge can be helpful to overcome the undesirable effects (e.g., missed reading) of the unreliable wireless communications. It may be also useful in resource management process by deciding which nodes can be switched safely to a sleep mode without affecting the coverage of the network. Association rule mining is the one of the most useful technique for finding behavioral patterns from wireless sensor network. Data mining techniques have recent years received a great deal of attention to extract interesting behavioral patterns from sensors data stream. One of the techniques for data mining is tree structure for mining behavioral patterns from wireless sensor network. By implementing the tree structure will face the problem of time taking for finding frequent patterns. By overcome that problem we are implementing associated correlated bit vector matrix for finding behavioral patterns of nodes in a wireless sensor network. By implementing this concept

1. INTRODUCTION

Wireless Sensor Network generates a large amount of data in the form of data stream and mining these streams to extract useful knowledge is a highly challenging task. In literature study, existing mechanism use sensor association rules measured in terms of frequency of patterns occurrence. Among the enormous number of rules generated, most of those are not valuable to reproduce true association among data objects. Moreover, mining associated sensor patterns from sensor stream data is essential for real-time applications, but it is not addressed in literature papers. In this proposed work, a new type of sensor behavioural pattern called associated sensor patterns to capture substantial temporal correlations in sensor data simultaneously is introduced to address the above-said problem. In this paper we are proposed an efficient associated correlated bit vector matrix for find the frequent item sets and associate correlated frequent pattern sets. The Data Mining in WSN are used to extract useful data from the huge amount of unwanted dataset. The need of mining to get knowledgeable data and discovers the behavioural patterns. As there are many Association techniques in data mining to find out the Frequent Patterns as per . The Association rule can apply on static data and stream data. The frequent patterns are those items, Sequences or substructure which reprise from the available dataset by providing the user specified frequencies. Whenever you want to find out the frequently occurred data apply association rules which will find out the frequent patterns from the dataset. Mining play main role to mine frequent item set in many data mining tasks. Over data streams, the frequent item set mining is mine the approximation set of frequent item sets in transaction with given support and threshold. It should support the flexible determine between mining accuracy and processing time. When the user-specified minimum support threshold is small, it should be time efficient. To propose an efficient algorithm the objective is generates frequent patterns in a very less time. Frequent patterns are very meaningful in data streams such as in network monitoring, frequent patterns relate an indicator for network attack to excessive traffic. In sales transactions, frequent patterns correspond to the top selling products with their relationships in a market. If we consider that the data stream consist of transactions, each items being a set of items, then the problem definition of mining frequent patterns can be written as given a set of transaction and finds all patterns with frequency above a threshold. Wireless sensor networks (WSNs) are successfully deployed in diverse monitoring and detection applications. In these applications, WSNs generate a large amount of data in the form of streams. Such data stream from WSN can be mined to extract knowledge in real time about the sensed environment (e mining certain behaviors and the network itself, and this presents new challenges for data mining techniques. Data mining techniques, which are well established in the traditional database systems , have recently received a great deal of attention as promising tools to extract interesting knowledge from sensor data streams (SDSs). Using knowledge discovery in WSN, one particular interest is to find behavioral patterns of sensor nodes, which are evolved from meta-data describing sensor behaviors. Data mining techniques, well established in the traditional database systems, recently became a popular tool in extracting interesting knowledge from sensor data streams (SDSs). Using knowledge discovery in WSNs, one particular interest is to find behavioral patterns of sensor nodes evolved from meta-data describing sensor behaviors. The application of fine grain monitoring of physical environments can be highly benefitted from discovering behavioral patterns (i.e., associated patterns) in WSNs. These behavioral patterns can also be used to predict the cause of future events which is used to detect faulty nodes, if any, in the network. For example, possibility of a node failure can be identified using behavioral pattern mining by predicting the occurrence of an event from a particular node, but no such event reported in subsequent iteration. As behavioral patterns reveal a chain of related events, source of the next event can be identified. For e.g. in an industry, fault in a particular process may trigger fault in other processes. In addition, behavioral patterns can also use to identify a set of temporally correlated sensors, thus improving operational aspects in WSNs.

Wireless sensor network is a collection of nodes organized into a cooperative network. It consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location predefined as central node called sink in multi-hop mode of transmission. Not only the unreliable wireless communication, the node of wireless sensor networks have to work with limited resources such as limited energy, limited processing capacity, limited storage, limited memory ,limited communication capacity, etc. So, wireless sensor networks suffers from lot of problems such as lost messages, delay in data delivery, loss of data, data redundancy, etc. resulting in poor quality of service.

2. PROPOSED SYSTEM

Mining play main role to mine frequent item set in many data mining tasks. Over data streams, the frequent item set mining is mine the approximation set of frequent item sets in transaction with given support and threshold. In this paper we are proposed an efficient correlated association rule mining for mining behavioral patterns from wireless sensor network. For mining association correlated patterns can be done by performing the two steps. In the first step we are finding frequent patterns of wireless sensor networks and second step is to test whether they are associated correlated patterns or not based on the confidence of each pattern in a transaction dataset. By performing those two operations we are implementing an efficient correlated bit vector matrix for finding behavioral patterns from a wireless sensor network. The implementation procedure of associated correlated bit vector matrix is as follows.

Associated Correlated Bit Vector Matrix:

Generation of Bit Vector Matrix:

In this module we can retrieve the transactional data set of sensor items from the data base. Take
the each transaction and generate bit vector matrix. The implementation of bit vector matrix is as follows.

1. Read each transaction from the data base (D) and get each item of sensor node id Si.
2. Read all the individual items of sensor nodes until the length of all transactional dataset is completed.
3. After completion of reading process we can sort the all node ids.
4. Find all frequent length of item sets(Ti) from the data base D

```
If Ti is not null
For each transaction (Ti) from database
For each item (Ii) in database D
If item (Ii) contains Transaction Item sets (Ti)
BV = 1
Else
BV=0
End for.
End for.
End if.
```

If all the maximum frequent item sets are empty, the maximum length of frequent item set is one.
Input: the bit vector matrix, Minimum support value
Output: Maximum Frequent patterns of sensor nodes

Process:

```
For each column in the bit vector matrix
Calculate number of value one in the current row
End for
Return max[n]
Sort (Max[n])
For each one in the max[n]
Calculate number of columns with the same number of ones
If number> minimum support value
Generate maximum number of candidate item sets from transaction
For each item set in candidate item sets
Calculate support (item set)
If(support(item set)>minimum support count)
Item set is frequent
End if
End for
End if
If maximum item sets is not null
Break;
End if
End for.
```

All the frequent item sets could be extracted from all the maximum frequent item sets according to the nonempty subsets of frequent item sets being still frequent. And the support of each frequent item set could be calculated, all the strong association rules can be mined from all the frequent item sets.

Associated Correlated Frequent Patterns:

By calculating confidence of two sensor, such as s1 , s2is as follows.

$$\rho(s1,s2) = P(s1,s2)-P(s1)P(s2)/P(s1,s2)+P(s1)P(s2)$$

Suppose we are calculating more than two nodes of confidence we are using the following formula.

$$P=P(s1,s2....sn)-P(s1)P(s2)....P(sn)/P(s1,s2....sn)+P(s1)P(s2)....P(sn)$$

3. REQUIREMENT ANALYSIS

A Software Requirements Specification (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

Functional Requirements:

In software engineering, a functional requirement defines a function of a software system or its

Component A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). How a system implements functional requirements is detailed in the system design. In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

Non Functional Requirements:

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviour. This should be contrasted with functional requirements that define specific behaviour or functions In general, functional requirements define what a system is supposed to do whereas nonfunctional requirements define how a system is supposed to be. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements," and "non-behavioural requirements." Qualities, that is, non-functional requirements, can be divided into two main categories:

1. Execution qualities, such as security and usability, which are observable at run time.
2. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

Software Requirements For Present Project:

1. **Operating System**: Any Operating System
2. **Run-Time**: OS Compatible JVM

Hardware Requirements:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Hardware Requirements for Present Project:

1. VDU: Monitor/ LCD TFT / Projector
2. Input Devices: Keyboard and Mouse
3. RAM: 512 MB
4. Processor: P4 or above
5. Storage: 10 to 100 MB of HDD space.

4. SYSTEM DESIGN

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. The UML has become the standard language used in Object-oriented analysis and design [citation needed]. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

Unified Modeling Language (UML) :

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of that is intended to provide a standard way to visualize the design of a system.The Unified Modeling Language (UML) offers a way to visualize a system's architectural blueprints in a diagram (see image), including elements such as:

- ➢ Any activity
- ➢ Individual component of the system
- ➢ And how they can interact with the other components
- ➢ How the system will run
- ➢ How entities interact with others (components and interfaces)
- ➢ External user interface

UML 2 has many types of diagrams which are divided into two categories. Some types represent Structural information and the rest represent general types of behavior, including a few that represent different aspects of interactions. These diagrams can be categorized hierarchically as shown in the following class diagram.

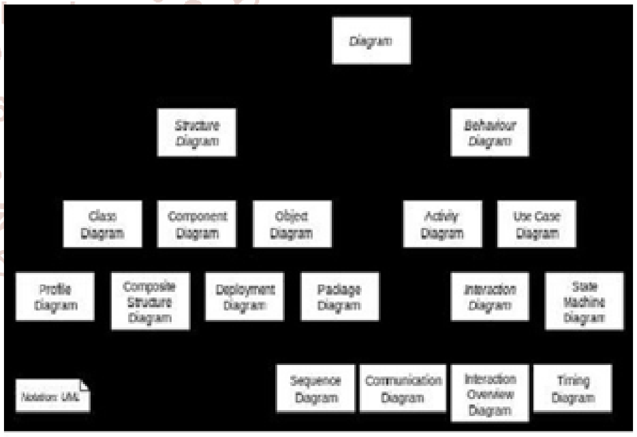


Fig 1 Block Diagram of various UML diagrams

Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that Shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

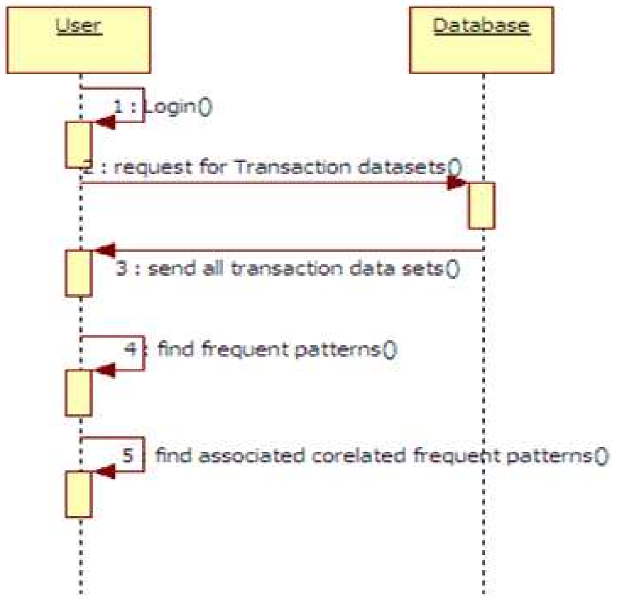

User
Database
1 : Login()
2 : request for Transaction datasets()
3 : send all transaction data sets()
4 find frequent patterns()
5 find associated corelated frequent patterns()

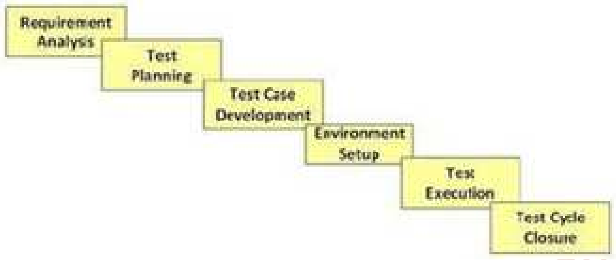
Fig 2 sequence diagram

5. TESTING

Software testing can also be stated as the process of validating and verifying that a software
Program/application/product:

1. meets the business and technical requirements that guided its design and development;
2. Works as expected; and
3. Can be implemented with the same characteristics.

Software Test Life Cycle:



Requirement Analysis

During this phase, test team studies the requirements from a testing point of view to identify the testable requirements. The QA team interacts with various stakeholders to understand the requirements in detail. Requirements could be either functional or nonfunctional automation feasibility for the given testing project is also done in this stage

Activities:

- ➢ Identify types of tests to be performed.
- ➢ Gather details about testing priorities and focus.
- ➢ Prepare Requirement Traceability Matrix (RTM).
- ➢ Identify test environment details where testing is supposed to be Carried out

Test Planning

This phase is also called Test Strategy phase. Typically, in this stage, a Senior QA manager will
determine effort and cost estimates for the project and would prepare and finalize the Test Plan

Activities:

- ➢ Preparation of test plan/strategy document for various types of Testing
- ➢ Test tool selection
- ➢ Test effort estimation
- ➢ Resource planning and determining roles and responsibilities.
- ➢ Training requirement

6. TEST EXECUTION

During this phase test team will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be
Performed.

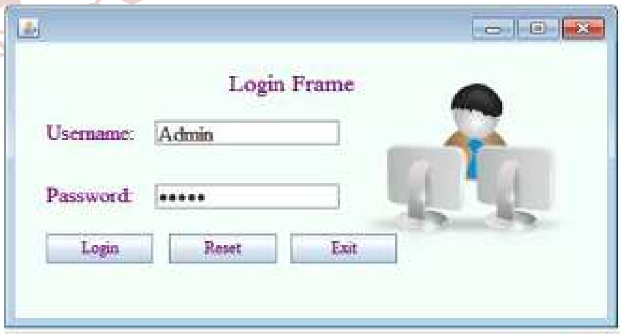
Activities:

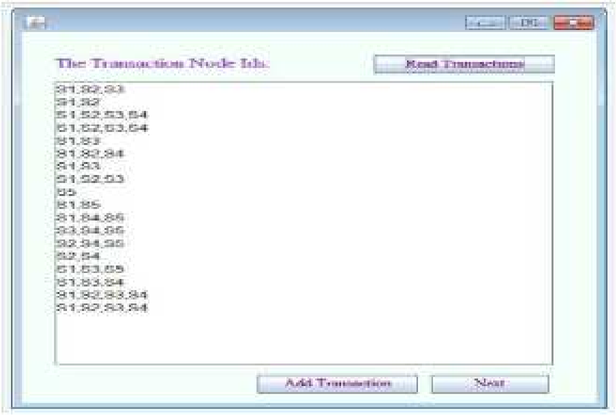
- ➢ Execute tests as per plan
- ➢ Document test results, and log defects for failed cases
- ➢ Map defects to test cases in RTM
- ➢ Retest the defect fixes
- ➢ Track the defects to closure

Grey box testing

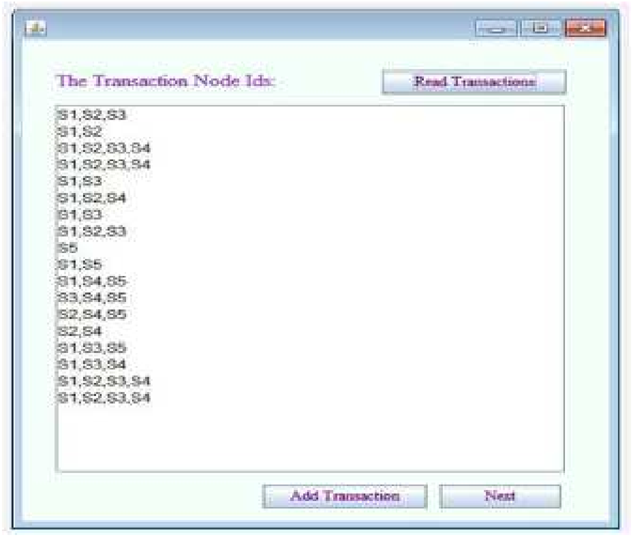
Grey box testing (American spelling: gray box testing) involves having knowledge of internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. Manipulating input data and formatting output do not qualify as greybox, because the input and output are clearly outside of the "black-box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test. However, modifying a data repository does qualify as grey box, as the user would not normally be able to change the data outside of the system under test. Grey box testing may also include reverse engineering to determine, for instance, boundary values or error.

7. RESULTS

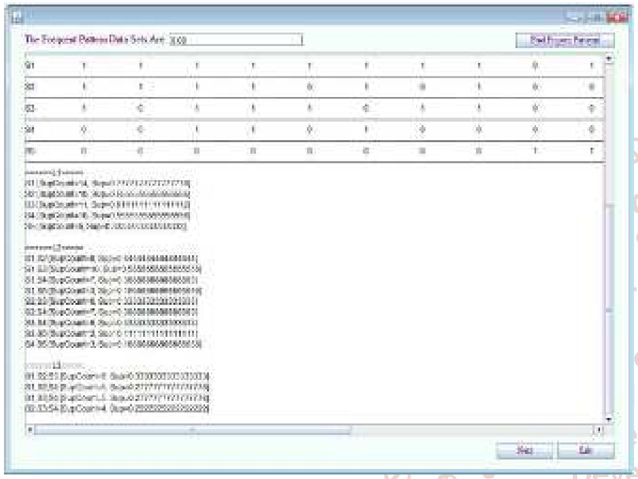




```
S1,S2,S3
S1,S2
S1,S2,S3,S4
S1,S2,S3,S4
S1,S3
S1,S2,S4
S1,S3
S1,S2,S3
S5
S1,S5
S1,S4,S5
S3,S4,S5
S2,S4,S5
S2,S4
S1,S3,S5
S1,S3,S4
S1,S2,S3,S4
S1,S2,S3,S4
```

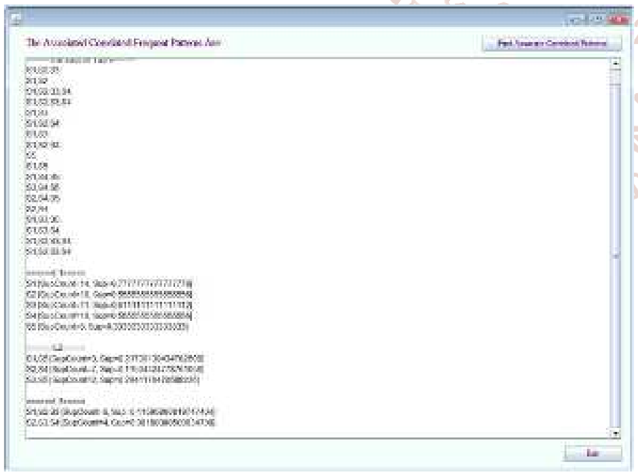


S1,S2,S3
S1,S2
S1,S2,S3,S4
S1,S2,S3,S4
S1,S3
S1,S2,S4
S1,S3
S1,S2,S3
S5
S1,S5
S1,S4,S5
S3,S4,S5
S2,S4,S5
S2,S4
S1,S3,S5
S1,S3,S4
S1,S2,S3,S4
S1,S2,S3,S4



S1	1	1	1	1	1	1	1	1	0	1
S2	1	1	1	1	0	1	0	1	0	0
S3	1	0	1	1	1	0	1	1	0	0
S4	0	0	1	1	0	1	0	0	0	0
S5	0	0	0	0	0	0	0	0	1	1

[illegible]


The Associated Correlated Frequent Patterns Are:
Find Associated Correlated Patterns
Exit

[illegible]

8. CONCLUSION

In this paper we are proposed an efficient association rule mining finding associated correlated frequent behavioral patterns from wireless sensor network. Our proposed associated correlated bit vector matrix for mining behavioral frequent patterns of wireless sensor network data. By implementing this process we can scan the entire data once and mine many properties is suitable for interactive mining. An extensive analysis of associated correlated bit vector matrix is finding associated frequent patterns mining and out performs the existing algorithm based on execution time and memory usage.

9. REFERENCES

[1] M. M. Rashid, I. Gondal and J. Kamruzzaman, (2013) Mining associated sensor patterns for data stream of wireless sensor networks, □ in Proc. 8th ACM Workshop Perform. Monitoring Meas. Heterogeneous Wireless Wired Netw., , pp. 91–98

[2] Jiinlong, XuConglfu, CbenWeidong, Pan Yunhe," Survey of the Study on Frequent Pattern Mining in Data Streams", 2004 IEEE International Conference on Systems, Man and Cybernetics.

[3] R. Agrawal and R. Srikant, "Fast algorithms for Mining Association Rules", 20th International Conference on Very Large Data Base, pp. 487–499, May1994.

[4] Md. Mamunur Rashid, IqbalGondal and JoarderKamruzzaman, "Share-Frequent Sensor Patterns Mining from Wireless Sensor Network Data", IEEE Transaction Parallel Distribution System, 2014.

[5] Imielienskin T. and Swami A. Agrawal R., "Mining Association Rules Between set of items in large databases," in Management of Data, 1993, p. 9.

[6] H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri R. Agrawal, "Fast Discovery of Association Rules," in Advances in Knowledge Discovery and Data Mining, 1996, pp. 307-328.

[7] M. Chen, and P.S. Yu J.S. Park, "An Effective Hash Based Algorithm for Mining Association Rules," in ACM SIGMOD Int'l Conf. Management of Data, May, 1995.

[8] R. Motwani, J.D. Ullman, and S. Tsur S. Brin, "Dynamic Itemset Counting And Implication Rules For Market Basket Data," ACM SIGMOD, International Conference on Management of Data, vol.26, no. 2, pp. 55–264, 1997.