ISSN No. 0976-5697

Volume 2, No. 4, July-August 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Implementation of Association Rule Mining in the domain of Dyeing Unit and Related Issues

Saravanan .M.S*
Research Scholar in R & D Centre, Bharathiar University,
Coimbatore & Asst. Prof. in Dept. of I.T in Vel Tech
Dr. RR & Dr. SR Technical University,
Tamilnadu, India
saranenadu@yahoo.co.in

Rama Sree .R.J
Professor & Head in the Department of Computer Science,
Rashtriya Sanskrit Vidyapeetha,
Tirupati,
Andhra Pradesh, India.
rjramasree@yahoo.com

Abstract: In the past two decade the textile industry has grown in high to export the fabrics to other countries. To export high quality fabrics, particularly cotton cloths, the colouring process plays major role in the dyeing domain. The dyeing is a dynamic, complex and involved various discipline. The dyeing process is difficult to automate processes, because of their interdisciplinary and dynamic in nature. Moreover, it is also critical to keep a check on the automated processes to produce the expected results in the form of quality and timely dyeing processes. Delivering high quality colours are very difficult due to lot of constraints, such as poor dye mix powders, bad processing methods, etc. Hence, the dyeing organizations were requiring quality dyeing process methods to produce perfect shade. Therefore, first time the process mining algorithms were used in the previous my research paper. But in this paper the alternative process model that is association rule mining approach was used in the domain of dyeing unit to produce better understanding dyeing process model in the form of association rules to overcome the difficulties and limitations of process mining algorithms. The benefit of association rule mining is very easy to understand the dyer to process the colour with little difficulties. Hence, these systems can reduce the cost, improved operational efficiencies, pH test error reduction, improved shade quality, etc. Therefore, the main focus of this paper is to implement the association rule mining approach in the dyeing unit and discussed the related issues.

Keywords: Textiles, Fabrics, Dyeing, Automate, Association rule mining, Shades

I. INTRODUCTION

The HM can be used to construct a process model reflecting the control flow behaviour that has been observed and recorded in an event log. The DWS algorithm generates process models for a cluster of process instances representing distinct discriminant rules. These discriminant rules identify structural patterns that are found in the process model but not registered in the log. The DWS uses the HM to construct process models. Experiments with these algorithms did not provide us with clean and desired process models which could be analyzed to obtain insight into the dyeing processes. The resulting process models were spaghetti-like structures. It is identified that these algorithms are not suitable for mining less structured or unstructured processes of dyeing. Given the characteristics of the domain, it is imperative that any mining approach should focus on simplicity of results.

The limitations of the above mentioned plug-ins and the need for simple models for the dyeing domain led us to explore an approach that may give simple models as well throw light on behavioral patterns implicitly registered in the event logs. Nowadays, the application of machine learning algorithms has become a widely adopted means to extract knowledge from vast amounts of data [1]. The process of finding best method for process mining through the machine learning algorithms, association rules were found to have potential to gain knowledge about the process and/or to make understood knowledge explicit. They are simple to understand as compared to the confusing and complex dependency graphs from the HM and express behavioral frequent patterns in the log. Therefore this paper focuses on to *"Investigate the usefulness of Association Rules as an alternate process model representation"*.

The paper is organized as follows. First, Section 2 introduces the concept of association rules; Section 3 reviews the Apiori, FPGrowth, H-Mine and LinkRuleMiner association rule mining algorithms. Then, Section 4 explains the use of the Weka machine learning library to derive association rules. The experimental results and related issues of HM and DWS process mining algorithms and Apriori, FPGrowth, H-Mine and LRM association rule mining algorithms using Emerald dyeing unit, dyeing logs are highlighted in Section 5. Section 6 concludes this paper by summarizing the findings and observations of the experiments conducted with Weka.

II. INTRODUCTION TO ASSOCIATION RULES

Today Information and Communication Technologies are widely used in enterprises to maintain every record of their interactions with a client or prospect. These records can be seen as a learning opportunity [2] if this data gathering process leads to data analysis. This process of data analysis allows us to comb through the data for noticing patterns, devising rules, coming up with new ideas, figuring out the right questions and making predictions about the future. For instance, the records can be analyzed for learning patterns for various transactions, viz., a pattern revealing a customer's buying preferences, a reader's visit to a website's specific section etc. This analysis is the focus of data mining domain. Data mining, also known as Knowledge Discovery in Databases (KDD) is defined as [3]: "Data Mining is the process of discovering patterns in data".

There are various tasks that can be performed using data mining techniques, viz., classification, estimation, prediction, affinity grouping, clustering, and description and profiling tasks. But our focus is on affinity grouping or association rules. The task of affinity grouping or association rules is to determine which things go together [2]. Association rules tell us about the association between two or more items or elements or tasks in a database. The Market Basket Analysis (MBA) is the largest application for algorithms discovering these association rules. It is a modeling technique based upon the theory that if a person buys a certain group of items, he or she is more (or less) likely to buy another group of items [4].

The MBA is based on discovering purchasing habits of the customers and the association between different items that customers place in their "shopping baskets". A sample association rule is given: Bread, Milk=> Butter | 90%. The items on LHS {Bread, Milk} of an association rule are called antecedents and the items {Butter} on the RHS are called consequents. An association rule can have multiple antecedents and multiple consequents. The 90% factor in the above rule indicates that 90% of the customers who bought bread and milk also bought butter. This percentage indicates the *certainty* or the *confidence* of this association rule. The confidence factor is one of the measures of the interestingness of an association rule. Another measure is the *support*. Support indicates the usefulness of an association rule. For example, if the above rule has a support of 5% it means that 5% of all the transactions under analysis show that bread, milk and butter are purchased together. When the technique of association rules is applied to event logs, the user would like to retrieve associations and frequent patterns existing amongst the various events in event logs. This is described in detail in the coming sections, but before that some important concepts related to the association rules viz., support, confidence etc. are explained.

A. *Definitions: Association rules, Support &Confidence*

Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in [5]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control etc. Various association mining techniques and algorithms will be briefly introduced and compared later.

a. *Association Rules:*

Association rules are formally defined as statements of the form X=>Y where X and Y are disjoint itemsets i.e., $X \cap Y = \emptyset$, and Y is a non empty itemset. X and Y are sets of items from the transactional data. This rule holds in a transaction set D with confidence c if c% of transactions in D that contain X also contain Y. The rule X=>Y has supports in the transaction set D if s% of transactions in D contains $X \cup Y$. An association rule suggests a strong co-occurrence relationship between items in antecedent and consequent of the rule. They do not necessarily imply causality.

b. *Support:*

In many situations, association rules involving sets of items that appear frequently in a database or transaction log are of interest. This means that only the items with high support are interesting. In absolute terms, support of an item is the number of times this item appears in a log. For the association rule: a=>b, support can be understood as the joint probability of a, b. It indicates the coverage of a rule i.e. how often a rule is applicable to a given dataset. An itemset (set of items) satisfying a minimum support value is referred to as frequent itemset or large itemset. This minimum support value is called the minimum support threshold. Support can be calculated as:

$$\text{Support }(a, b) = \left(\frac{\#\text{Tuples}(a, b)}{\#\text{Tuples}}\right)$$

Equation 1. Support of an itemset

c. *Confidence:*

Confidence of an association rule X=>Y is the probability of finding Y in the transaction set D. In simple words, it indicates how frequently items in Y appear in transactions that contain X. It is also referred to as the accuracy of the association rule. For example an association rule involving items a and b: a =>b, the confidence is the conditional probability of 'b' given 'a', i.e. how much percentage of transactions in the database that has item 'a' also contains item 'b'. It is given as:

$$\text{Confidence }(a, b) = \left(\frac{\#\text{Tuples}(a, b)}{\#\text{Tuples}(a)}\right)$$

Equation 2. Confidence of an association rule

Confidence can also be derived from Equation 1:

$$\text{Confidence }(a, b) = \left(\frac{\text{Support}(a \cup b)}{\text{Support}(a)}\right)$$

Equation 3. Confidence of a rule can be derived using support count

III. ALGORITHMS FOR ASSOCIATION RULES

This section explains methods that generate association rules. All algorithms for association rule mining involves two steps:

a. *Find all Frequent Item sets:*

According to the definition of association rules, these itemsets will occur at least as frequently as a predetermined minimum support count.

b. *Generate Strong Association Rules from the Frequent Item sets:*

In this step, strong association rules are derived from the frequent itemsets generated in the first step. Strong rules are the rules that satisfy both a minimum support threshold and a minimum confidence threshold. They are preferred because it is not practical to do an exhaustive search for thousands of potential rules that can be generated from a database. Many of these rules will not be of interest and use because they may be unreliable due to low support or confidence values. Therefore it is common to generate only those rules that have a minimum specified support and confidence values. Association rules can be discovered using algorithms like the Apriori, Tertius, FPGrowth, H-Mine and newly proposed LinkRuleMiner, etc. Below it is given that a brief description of some of these algorithms.

A. *The Apriori Algorithm*

The Apriori algorithm was proposed by R. Agrawal and R. Srikant [6] in 1994 for mining frequent itemsets for

Boolean association rules. Boolean association rules are the rules involving associations between the presence/absence of items. A rule describing associations between quantitative items or attributes is called a quantitative association rule.

The Apriori algorithm uses a level-wise search to generate frequent itemsets traversing from frequent 1-itemsets (an itemset containing k items is referred to as a k-itemset) to the maximum size of frequent itemsets. This iterative search is continued till no new frequent itemsets can be generated.

B. Analysis of Apriori Algorithm

Apriori algorithm step by step approach, a combination of a candidate to collect items and scanning the database calculated candidates support the project and sets the rules strength. Although the algorithm will have many candidates which can not be established prior to deleting the project set to reduce the huge amount of computation, but still needs a lot of calculation, but it is also necessary to scan the database. Therefore, if the large-scale database system inside, the efficiency of the algorithm is still not good enough, so if we can further reduce the amount of computation or reduce the number of times of database scanning, can effectively improve mining efficiency [7]. Apriori algorithm to improve the effectiveness of the various studies is mainly towards reducing the amount of computation and by less the number of scanning the database to improve is required.

C. The FPGrowth Algorithm

The conventional frequent pattern mining method, *Apriori* discussed in the previous section. As analyzed, the major costs in *Apriori*-like methods are the generation of a huge number of candidates and the repeated scanning of large transaction databases to test those candidates. In short, *the candidate-generation-and-test operation is the bottleneck for* Apriori-*like methods*. Hence, there is a question that can *avoid candidate-generation and test in frequent pattern mining?* To attack this problem, the frequent pattern growth (*FPGrowth)*, a pattern growth method for frequent pattern mining is introduced. For this an effective data structure *FP-tree* is used. The efficient algorithm for mining frequent patterns from an *FP-tree* is verified for its correctness and also dealt that how to scale the method to mine large databases which cannot be held in main memory. Finally the experimental results and performance studies are reported.

One of the currently fastest and most popular algorithms for frequent item set mining is the FPGrowth algorithm [8], which can save considerable amounts of memory for storing the transactions. The basic idea of the FPGrowth algorithm can be described as a *recursive elimination* scheme: in a preprocessing step delete all items from the transactions that are not frequent individually, i.e., do not appear in a user-specified minimum number of transactions. Then select all transactions that contain the least frequent item (least frequent among those that are frequent) and delete this item from them [9]. Recurse to process the obtained reduced also known as *projected* database, remembering that the item sets found in the recursion share the deleted item as a prefix. On return, remove the processed item also from the database of all transactions and start over, i.e., process the second frequent item etc. In these processing steps the prefix tree, which is enhanced by links between the branches, is exploited to quickly find the transactions containing a given item and also to remove this item from the transactions after it has been processed. Based on the above properties the following algorithm is used for mining frequent patterns using FP-tree.

a. Analysis of FPGrowth Algorithm:

Let's now examine the efficiency of the algorithm. The FPGrowth mining process scans the FP-tree of DB once and generates a small pattern-base B_{ai} for each frequent item a_i, each consisting of the set of transformed prefix paths of ai. Frequent pattern mining is then recursively performed on the small pattern-base B_{ai} by constructing a conditional FP-tree for B_{ai}. A FP-tree is usually much smaller than the size of DB. Similarly, since the conditional FP-tree or a_i constructed on the pattern-base B_{ai}, it should be usually much smaller and never bigger than B_{ai}. Moreover, a pattern-base B_{ai} is usually much smaller than its original FP-tree, because it consists of the transformed prefix paths related to only one of the frequent items, a_i.

Thus, each subsequent mining process works on a set of usually much smaller pattern-bases and conditional FP-trees. Moreover, the mining operations consist of mainly prefix count adjustment, counting local frequent items, and pattern fragment concatenation. This is much less costly than generation and test of a very large number of candidate patterns. Thus the algorithm is efficient. From the algorithm and its reasoning, one can see that the FPGrowth mining process is a divide and conquer process, and the scale of shrinking is usually quite dramatic. If the shrinking factor is around 20~100 for constructing an FP-tree from a database, it is expected to be another hundreds of times reduction for constructing each conditional FP-tree from its already quite small conditional frequent pattern-base. Notice that even in the case that a database may generate a large number of frequent patterns, the size of the FP-tree is usually quite small. For example, for a frequent pattern of length 100, "a_1 …..,a_{100}", the FP-tree construction results in only one path of length 100 for it, such as "($a_1 \rightarrow \ldots \rightarrow a_{100}$)". The FPGrowth algorithm will still generate about 10^{30} frequent patterns (if time permits!!), such as "a_1, a_2, … a_1a_2, …, $a_1a_2a_3$, ..., $a_1 \ldots a_{100}$". However, the FP-tree contains only one frequent pattern path of 100 nodes, and there is no need to construct any conditional FP-tree in order to find all the patterns.

Therefore, the FPGrowth algorithm is much faster than Apriori because of no candidate generation, compresses the dataset and only limited number of passes over dataset. Moreover, with FP-tree, it can be derived some condensed expression of frequent patterns. In the case that it has only a long pattern $a_1 \ldots a_{100}$, hence, the user can only output the long pattern itself and omit all proper sub-patterns. The FPGrowth also has disadvantages such as the FP-Tree may not fit in memory and FP-Tree is expensive to build because of the following reasons (i) FP-Tree takes time to built, but once it is built, frequent itemset are read off easily (ii) Time is waster especially if support threshold is high as the only pruning that can be done is on single items (iii) Support can only be calculated once the entire dataset is added to the FP-Tree.

D. *The H-Mine Algorithm*

The existing association rule mining algorithms may still encounter some difficulties in different cases. First, a huge memory space is required to serve the mining, and the main memory consumption is usually hard to precisely predict. An *Apriori*-like algorithm generates a huge number of candidates for *long or dense* patterns. FPGrowth avoids candidate generation by compressing the transaction database into an *FP-tree* and pursuing partition-based mining recursively. However, if the database is *huge and sparse*, the *FP-tree* will be large and the space requirement for recursion is a challenge. Neither approach is superior in all cases.

It is hard to select an appropriate mining method on the fly if no algorithm fits all cases. Last, large applications need more scalability. Many existing methods are efficient when the data set is not very large. Otherwise, their core data structures (such as *FP-tree*) or the intermediate results for example the set of candidates in *Apriori* or the recursively generated conditional databases in *FPGrowth* may not fit into the main memory and can easily cause thrashing. This poses a new challenge: *can we work out a better method in that: (i) the main memory requirement is precisely predictable and moderate, even for very large databases; and (ii) it is efficient in most occasions (dense vs. sparse, huge vs. memory-based data sets)?*

Hence a data structure, H-struct, and a mining method, H-mine, to overcome these difficulties, with the following progress. First, a memory based, efficient pattern-growth algorithm, H-mine (Mem), is proposed for mining frequent patterns for the data sets that can fit into the main memory. A simple, memory based hyper-structure, H-struct, is designed for fast mining. Second, theoretically, H-mine (Mem) has a polynomial space complexity and is thus more space efficient than pattern-growth method such as *FPGrowth* when mining sparse data sets, and also more efficient than *Apriori* based methods which generate a large number of candidates. Therefore the implementation of the H-Mine algorithm is used to find the frequent patterns.

a. *Analysis of H-Mine Algorithm:*

When trying to parallel H-Mine one will soon realizes that there is a problem inherent to the algorithm itself which greatly increases the difficulties of parallelization efforts. The problem is H-struct link adjustment that is performed dynamically in the middle of mining process, i.e. when returning to the previous level of H-header after finished mining a queue. Therefore, the algorithm cannot proceed to mine the next queue before finishing the current queue to get information to adjust the links.

Link adjustment is absolutely required in the algorithm in order to mine all frequent patterns from data set completely else the mined patterns will be incomplete. Thus the mining process in each queue of H-Mine is dependent on its previous queue. This property makes parallelization is impossible unless it can take away the queue dependency from the algorithm. This property will overcome by the newly proposed association rule mining algorithm "LinkRuleMiner".

E. *The Link Rule Miner Algorithm*

LinkRuleMiner differs from H-Mine algorithm in two aspects: link structure and data processing order. LinkRuleMiner link structure is designed in such a way that it does not need any kinds of link adjustment and the data is processed in the reversed way compared to that of H-Mine algorithm. These two modifications has successfully removed any queue dependencies in LinkRuleMiner algorithm, and thus made it possible to do parallelization. The algorithm of LinkRuleMiner in general is similar to that of H-Mine, the complexity and performance is also about the same order. It also uses the data set in Table 1 to find all frequent patterns that satisfy minimum support of 2. Similar to H-Mine, the LinkRuleMiner algorithm can be divided in two phases, building data structures that involved two times scanning of data set and mining the data structures built in memory. The data set used for this discussed is shown in the table 1.

Table 1. Data set, Fitems: Computer, Google, Internet, Keyboard, Mouse.

Transaction ID	List of Items	Fitem Projection
1000	Google, Internet, Keyboard, Laptop, Mouse, Program	Google, Internet, Keyboard, Mouse
2000	Computer, Google, Internet, Keyboard, Web	Computer, Google, Internet, Keyboard
3000	Computer, Email, Internet, Keyboard, Mouse, Virus	Computer, Internet, Keyboard, Mouse
4000	Computer, Google, Internet, Network	Computer, Google, Internet

The data set is taken to find all frequent patterns that satisfy minimum support of 2.

a. *Building the Data Structures:*

i. Scan the data set once for all transactions and items to find all frequent items and store it in an array of H-header i.e. same as H-Mine, ordered alphabetically by item. The fitems are Computer, Google, Internet, Keyboard, Mouse and it is identified that the five frequent patterns, those are fitems itself in H-header.

ii. Scan the data set once more to project all transactions into fitems and store it in arrays of H-struct as shown in Figure 1. In this stage, the initial H-header and H-struct links are built. Note that differ to H-Mine, initial link structure in LinkRuleMiner are built completely for all queues not only for items in the first position. The rest of mining is then performed on these data structures.

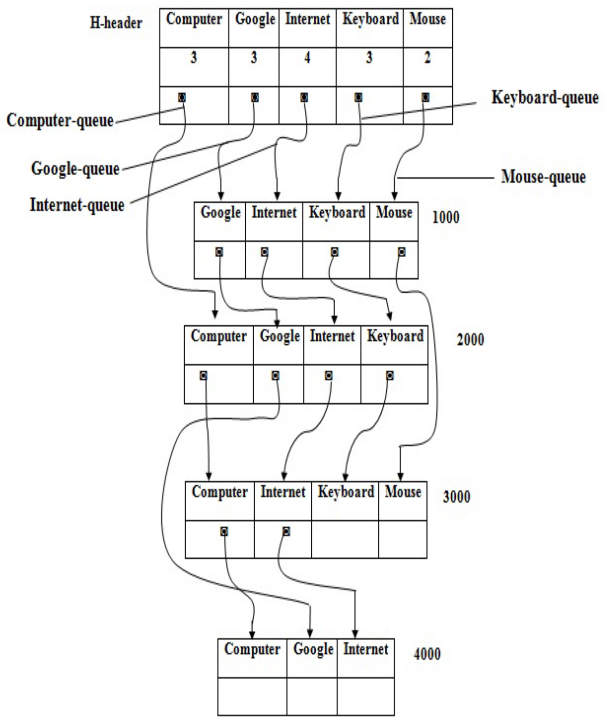


Figure 1. H-header and H-struct (LinkRuleMiner)

b. *Mining the Data Structures:*

The data mining process in LinkRuleMiner algorithm must be done in reversed alphabetical order of frequent items as stated in H-header: Computer-Google-Internet-Keyboard-Mouse, i.e. from backward, so it differ to H-Mine. First, this algorithm will mine all patterns containing 'Mouse' without Computer, Google, Internet, Keyboard then all patterns containing 'Keyboard' without Computer, Google, Internet and so on. Finding all patterns containing 'Mouse' without Computer, Google, Internet, Keyboard is not needed, because it is already found them just as stated in H-header, i.e. it has two patterns.

To find all patterns containing 'Keyboard' without Computer, Google, Internet from H-header follow Keyboard-queue, find all frequent items appearing after 'Keyboard' and store it in conditional H-header called HKeyboard-header as shown in Figure 3. Hence, it is identified that a frequent pattern 'KeyboardMouse' as stated in HKeyboard-header. To find all patterns containing 'Internet' without Computer, Google:

i. From H-header follow Internet-queue, find all frequent items appearing after 'Internet' and store it in conditional H-header called HInternet-header, shown in Figure 2. Hence, it is identified another two frequent patterns, 'InternetKeyboard' and 'InternetMouse' as stated in HInternet-header.
ii. Similarly, from HInternet-header follow InternetMouse-queue and find all frequent items appearing after 'Mouse' and found nothing. Therefore, go to the next queue 'InternetKeyboard', find all frequent items appearing after 'Keyboard' and store it in conditional H-header called HInternetKeyboard-header, shown in Figure 3. When counting the items, it is identified that to link all same items together and build new links in H-struct for the next level queues. Hence, found another frequent pattern, 'InternetKeyboardMouse' as stated in HInternetKeyboard-header. note that when moving from InternetMouse-queue to InternetKeyboard-queue does nothing to the H-struct link structure. There is no link adjustment between queues as required in H-Mine.
iii. Repeat this process recursively until there is no more level of queue can be built and then return to the previous H-header and continue to mine the next queue for all queues.

Note that, it must perform mining process in the reversed alphabetical order to get results. The reason is that in the step two above, it building new links in H-struct while counting items in each queue. This process alters link structure for queues in the next proper alphabetical order. Therefore, it must need to do mining process in reversed alphabetical order, such that link structure altered in the process will belong to the queues that are already mined.

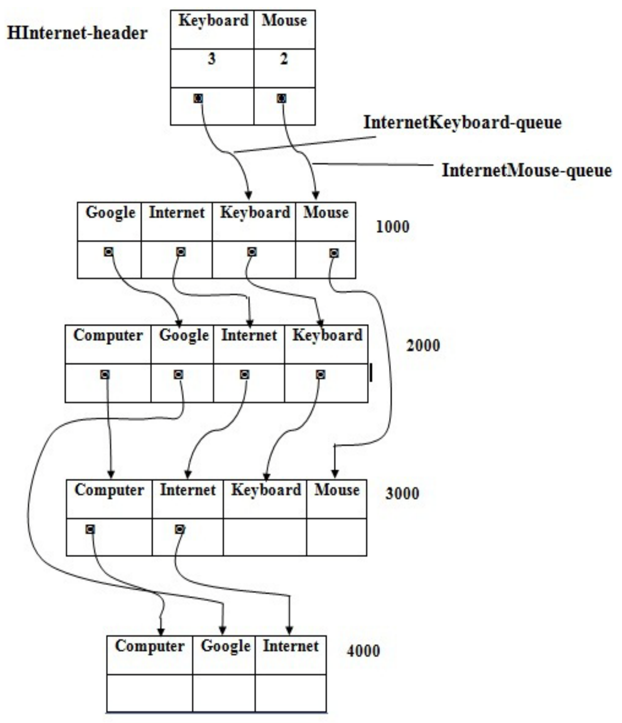


Figure 2. HInternet-header

c. *Analysis of Link Rule Miner Algorithm:*

The association rule mining algorithm H-mine can be scaled up to very large databases due to its small and precisely predictable run-time memory overhead and its database partition mining technique. Hmine partitions a large database into a set of relatively uniform-sized partitions, and mines each partition using H-mine. Since mining each partition in main memory is highly efficient, and many mined patterns can be shared among uniformly partitioned databases to reduce the effort of pattern matching in the additional database scan, the overall cost is far less than the proposed association rule mining algorithm LRM, which has been demonstrated in our performance study.

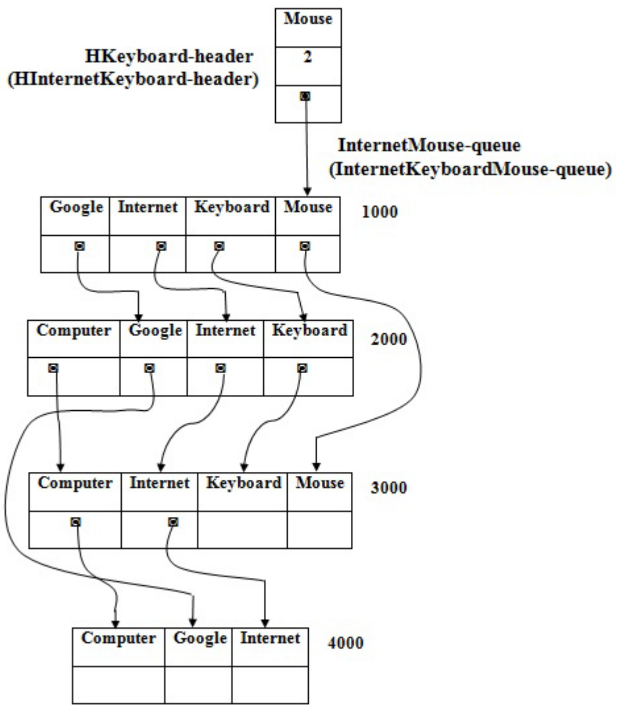

Mouse
2
HKeyboard-header
(HInternetKeyboard-header)
InternetMouse-queue
(InternetKeyboardMouse-queue)
Google Internet Keyboard Mouse
1000
Computer Google Internet Keyboard
2000
Computer Internet Keyboard Mouse
3000
Computer Google Internet
4000

Figure 3. HKeyboard-header/ HInternet Keyboard-header

The Speedup ratio of LRM, that is mining time is the time needed to mine H-header and H-struct after they are completely constructed in memory. Total time is the sum of H-header and H-struct creation time, mining time and results reducing time.

Based on the above analysis, one can see that LRM represents a new, highly efficient and scalable mining method. Its structure time consuming and space preserving mining methodology may have strong impact on the development of new, efficient and scalable association rule mining algorithm for mining dyeing process patterns from various data sets of Emerald dyeing unit's dyeing process and Jayabala dyeing unit's dyeing process.

IV. ASSOCIATION RULES AND WEKA LIBRARY

The Waikato Environment for Knowledge Analysis (Weka) is a comprehensive suite of Java class libraries that implement many state of the art machine learning and data mining algorithms. Weka is freely available on the World Wide Web and accompanies a new text on data mining [10] which documents and fully explains all the algorithms it contains. Applications written using the Weka class libraries can be run on any computer with a Web browsing capability; this allows users to apply machine learning techniques to their own data regardless of computer platform. Tools are provided for pre processing data, feeding it into a variety of learning schemes, and analyzing the resulting classifiers and their performance. An important resource for navigating through Weka is its on-line documentation, which is automatically generated from the source. The primary learning methods in Weka are "classifiers", and they induce a rule set or decision tree that models the data. Weka also includes algorithms for learning association rules and clustering data. All implementations have a uniform command line interface. A common evaluation module measures the relative performance of several learning algorithms over a given data set.

Tools for pre processing the data, or "filters," are another important resource. Like the learning schemes, filters have a standardized command line interface with a set of common command line options. The Weka software is written entirely in Java to facilitate the availability of data mining tools regardless of computer platform. The system is, in sum, a suite of Java packages, each documented to provide developers with state of the art facilities.

A. Input for Association Analysis Algorithms in Weka

To obtain association rules for tasks in an event log, this log is provided as input to Weka. The output of the Case Data Extraction (CDE) plug-in implemented in ProM is used as an input to Weka. The CDE converts the case data of an event log into a table. Case data refers to: *PI data attributes ATE data attributes*, *originators and event types.* A selection of a dyeing log and its CDE table obtained from the CDE plug-in. When the data is exported to a *comma separated values format (CSV)* file it can serve as an input to Weka. The CDE table can be exported to a CSV file using the Standard CSV Export plug-in available in ProM. In the next section, the user show some association rules generated from the logs for Emerald dyeing process and evaluate if they can be useful in understanding dyeing processes.

V. EXPERIMENTAL RESULTS AND RELATED ISSUES

The CSV files obtained from ProM can be used for discovering association rules in the Weka library. The library provides the Apriori, FPGrowth, H-Mine and LinkRuleMiner algorithms for association analysis. The purpose of this section is to analyze the usefulness of association rules in gaining insights into the less structured processes of dyeing. These rules would be evaluated on criteria like simplicity, ease of understanding and insights provided for the underlying process. It should be remembered that the motivation to explore these rules is to see how they fare on less structured processes in comparison to the mining plug-ins: HM and the DWS. This section illustrates some experiments with the Emerald dyeing process and compares the results of these experiments for three mining approaches: The HM, the DWS and the association analysis. This will give an indication of the potential of these rules to extract behavioral knowledge from the dyeing event logs. These experiments are illustrated below:

A. Illustration 1

The dyeing log used in this experiment is about the treatments, that the shades receive. The log has 683 PIs, 70 different ATEs and 12,294 total numbers of ATEs. The dyer used this log for experimenting with the HM process mining algorithm and the Apriori association rule mining algorithm provided in Weka. Parameter settings used for the Apriori algorithm are confidence= 0.9, support=0.7 and the desired number of rules =10. The HM process mining algorithm was experimented with default settings. The following figures 4 and 5 gives the process models generated by the HM process mining algorithm and Apriori association rule mining algorithm respectively.

As apparent, the process model generated by the HM is complex and confusing. Undoubtedly the model represents the disorder, unstructured and flexibility in the treatment process. From this process model it is difficult to trace out even the treatment route followed by a single shade. The complexity and hugeness of this model restricts the stakeholders from extracting any information about the underlying process except the fact that the process has very little structure. Moreover, the process model is constructed from the causal dependencies between the activities in the log and it does not provide any information about the behavioral patterns existing implicitly in the event log.

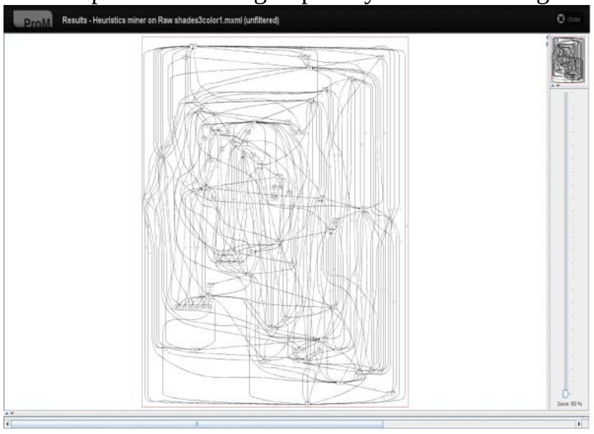
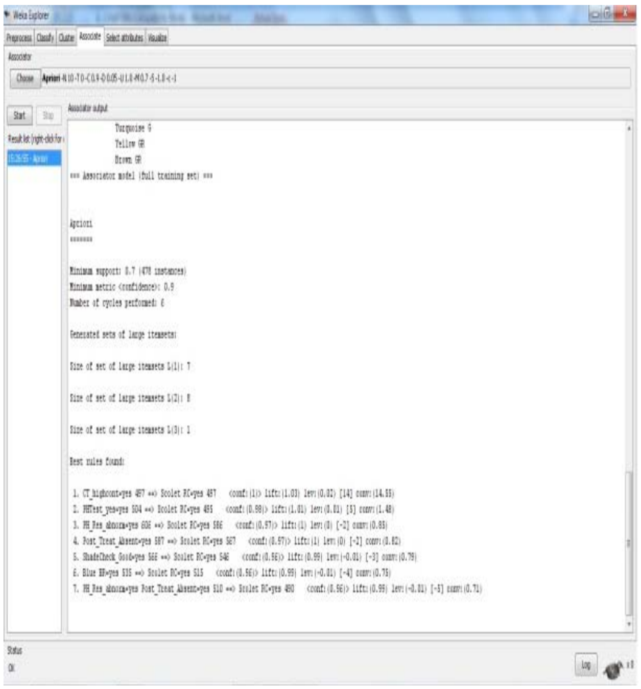

ProM
Results - Heuristics miner on Raw shades3color1.mxml (unfiltered)

Figure 4. The complex process model from the Heuristics Miner process mining algorithm from ProM tool with PIs = 683 and Event Logs =12,294

In contrast to this model, the Apriori association analysis algorithm presents the process model in form of association rules. These rules as already mentioned are the behavioural patterns existing in the log but registered implicitly. If the user analyze a rule, for example,

PH_Res_abnorm, Post_Treat_Absent -/-> Scolet RC
PH_Res_abnorm=yes, Post_Treat_Absent=yes 510 ==>
Scolet RC=yes 490 :(0.96),

says, the treatments *PH_Res_abnorm* and *Post_Treat_Absent* are always eventually followed by the treatment *Scolet RC*. In the rule, the number before the arrow is the number of instances for which the predecessor is true and the number after the arrow is the number of instance in which the consequent is true and the confidence is the ratio between the two [11].



Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.7 -S -1.0 -c -1

```
          Turquoise G
          Yellow GR
          Brown GR
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.7 (478 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 6

Generated sets of large itemsets:

Size of set of large itemsets L(1): 7

Size of set of large itemsets L(2): 8

Size of set of large itemsets L(3): 1

Best rules found:

 1. CT_highcont=yes 487 ==> Scolet RC=yes 487    <conf:(1)> lift:(1.03) lev:(0.02) [14] conv:(14.55)
 2. PHTest_yes=yes 504 ==> Scolet RC=yes 495    <conf:(0.98)> lift:(1.01) lev:(0.01) [5] conv:(1.48)
 3. PH_Res_abnorm=yes 606 ==> Scolet RC=yes 586    <conf:(0.97)> lift:(1) lev:(0) [-2] conv:(0.85)
 4. Post_Treat_Absent=yes 587 ==> Scolet RC=yes 567    <conf:(0.97)> lift:(1) lev:(0) [-2] conv:(0.82)
 5. ShadeCheck_Good=yes 566 ==> Scolet RC=yes 546    <conf:(0.96)> lift:(0.99) lev:(-0.01) [-3] conv:(0.79)
 6. Blue HF=yes 535 ==> Scolet RC=yes 515    <conf:(0.96)> lift:(0.99) lev:(-0.01) [-4] conv:(0.75)
 7. PH_Res_abnorm=yes Post_Treat_Absent=yes 510 ==> Scolet RC=yes 490    <conf:(0.96)> lift:(0.99) lev:(-0.01) [-5] conv:(0.71)
```

Figure 5. The simple process model by Apriori algorithm generated Association Rules from Weka tool with PIs = 683 and Event Logs =12,294 for the Figure 4

The Confidence of this rule is 0.96 indicating that in 96% of the log traces where the activities *PH_Res_abnorm* and *Post_Treat_Absent* occur, the activity *Scolet RC* also occurs. The stakeholders from the dyeing domain can benefit from such information because they can be well prepared to treat shades undergoing the treatments *PH_Res_abnorm* and *Post_Treat_Absent* (as they know the treatment *Scolet RC* will also be prescribed to these shades).

From this example it becomes apparent that the stakeholders (dyeing staff or researchers) can benefit better from association rules than the complex models as shown in Figure 5 as the rules are simple to understand and show a pattern present in the log. The rules as seen in the figure above are ranked according to their confidence. This gives an idea of the strength of different rules generated by the algorithm. Further, the rules do not exhibit problems like the unclear AND or XOR, dangling activities, missing dependencies which degrade the quality of the process model. This comparison between the HM and the association analysis technique leads to the fact that association analysis looks a promising approach for mining less structured processes. In the next section, the user compares the DWS with the association analysis algorithms and evaluate if the association rules fare better than the discriminant rules.

B. Illustration 2

For this experiment the same log as used for Illustration 1 is used. The DWS process mining algorithm and the Apriori association rule mining algorithms are used on this log. The Figure 6 and 4.9b gives the process models generated by these two algorithms respectively:

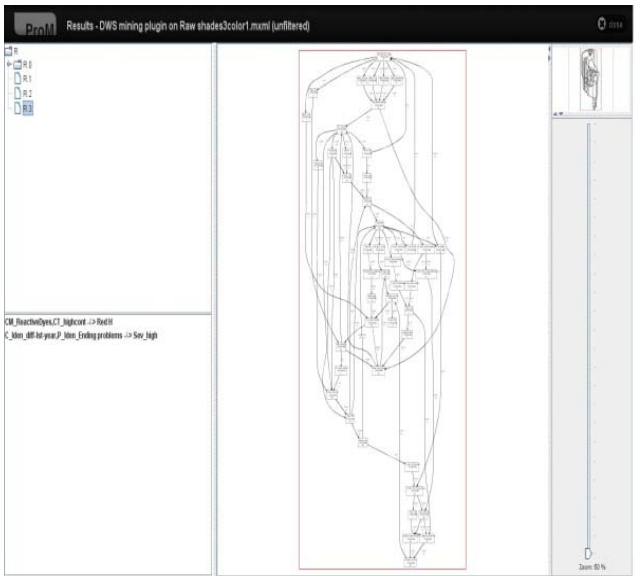

ProM
Results - DWS mining plugin on Raw shades3color1.mxml (unfiltered)
close
R
R.0
R.1
R.2
R.3
CM_ReactiveDyes,CT_highcont -/-> Red H
C_Iden_diff-Ist-year,P_Iden_Ending problems -/-> Sev_high
Zoom: 60 %

Figure 6. The complex process model with clusters from the DWS process mining algorithm from ProM tool with PIs = 683 and Event Logs =12,294

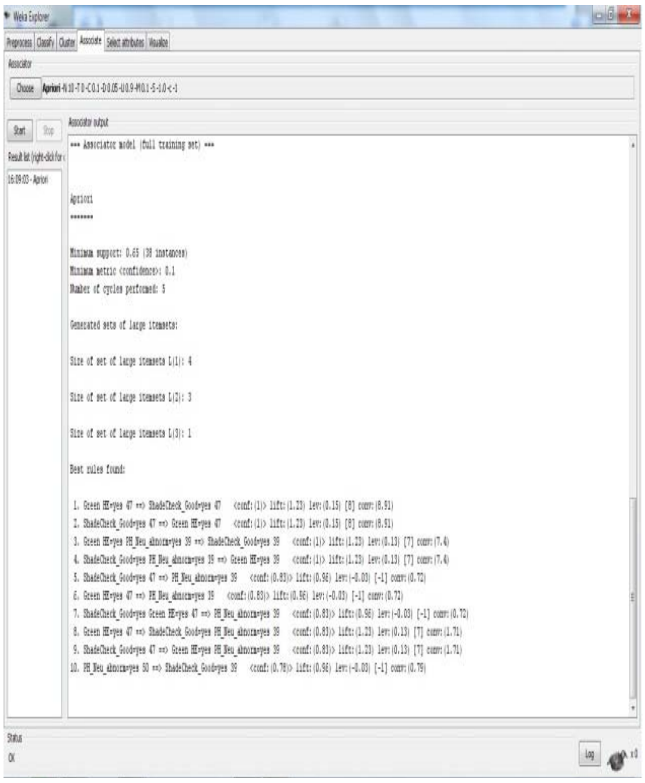
The discriminant rule in the process variant R.3 is:

C_Iden_diff_1st_year, P_Iden_Ending problems -/-> Sev_high

It is interpreted as the treatments *C_Iden_diff_1st_year, P_Iden_Ending problems;* and also the treatments *Sev_high* are given to 50% of the total shades whose information is recorded in the event log, but these three treatments together are not given even to 10% of the total shades. These percentages represent the parameters: sigma and gamma of the DWS approach. The above interpretation of this discriminant rule is not straightforward to understand as it involves the understanding of these frequency related parameters. Moreover, the emphasis of these rules is also on the execution order of the activities which is also seen as a limitation of the rule in case the 'timestamp' information is missing from the log and the ordering of activities is based on only on the date of execution. This can create problems when multiple activities are registered on same date but timestamp information is not present. In this case, these rules will show incorrect behavioral pattern as they depend on the ordering of activities. It is also found that the rules do not take into account the event type information associated with the activities.

In presence of multiple event types, it is not known which events or activity and event type are included in the rule. This may hinder the complete understanding of the situation because it may be possible, for example that the start of one treatment and the end of another treatment are both frequent i.e. it means that as many times one treatment finishes the same number of times the other treatment starts.

In contrast to the discriminant rules, the Apriori association rules are easy to understand. The relative importance of multiple discriminant rules for a process variant is not known. The confidence and predictive accuracy values are used for ranking the association rules in the Apriori algorithm. The DWS mining algorithm uses the HM algorithm for process discovery and as already said the HM is not an appropriate mining technique for dyeing processes. Also, the DWS analysis plug-in assumes that the mined process model is a dependency graph. This restriction makes the DWS plug-ins less flexible.



Apriori -N 10 -T 0 -C 0.1 -D 0.05 -U 0.9 -M 0.1 -S -1.0 -c -1

```
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.65 (38 instances)
Minimum metric <confidence>: 0.1
Number of cycles performed: 5

Generated sets of large itemsets:

Size of set of large itemsets L(1): 4

Size of set of large itemsets L(2): 3

Size of set of large itemsets L(3): 1

Best rules found:

 1. Green HE=yes 47 ==> ShadeCheck_Good=yes 47    <conf:(1)> lift:(1.23) lev:(0.15) [8] conv:(8.91)
 2. ShadeCheck_Good=yes 47 ==> Green HE=yes 47    <conf:(1)> lift:(1.23) lev:(0.15) [8] conv:(8.91)
 3. Green HE=yes PH_Neu_abnorm=yes 39 ==> ShadeCheck_Good=yes 39    <conf:(1)> lift:(1.23) lev:(0.13) [7] conv:(7.4)
 4. ShadeCheck_Good=yes PH_Neu_abnorm=yes 39 ==> Green HE=yes 39    <conf:(1)> lift:(1.23) lev:(0.13) [7] conv:(7.4)
 5. ShadeCheck_Good=yes 47 ==> PH_Neu_abnorm=yes 39    <conf:(0.83)> lift:(0.96) lev:(-0.03) [-1] conv:(0.72)
 6. Green HE=yes 47 ==> PH_Neu_abnorm=yes 39    <conf:(0.83)> lift:(0.96) lev:(-0.03) [-1] conv:(0.72)
 7. ShadeCheck_Good=yes Green HE=yes 47 ==> PH_Neu_abnorm=yes 39    <conf:(0.83)> lift:(0.96) lev:(-0.03) [-1] conv:(0.72)
 8. Green HE=yes 47 ==> ShadeCheck_Good=yes PH_Neu_abnorm=yes 39    <conf:(0.83)> lift:(1.23) lev:(0.13) [7] conv:(1.71)
 9. ShadeCheck_Good=yes 47 ==> Green HE=yes PH_Neu_abnorm=yes 39    <conf:(0.83)> lift:(1.23) lev:(0.13) [7] conv:(1.71)
10. PH_Neu_abnorm=yes 50 ==> ShadeCheck_Good=yes 39    <conf:(0.78)> lift:(0.96) lev:(-0.03) [-1] conv:(0.79)
```

Figure 7. The simple process model by Apriori algorithm generated Association Rules from Weka tool with PIs = 683 and Event Logs =12,294 for the Figure 6

The association rules however do not provide any visual process discovery, but they can be used as the basis for clustering. Later, any mining and analysis technique can be applied to these clusters. Moreover, the advantage of association rules is also reflected by the fact that they represent frequent patterns in the log. If the log contains complications, the association rules will depict the frequently occurring complications and this fact is strengthened by the statistical support provided by the confidence or predictive accuracy values. These rules in presence of adequate domain knowledge would definitely provide meaningful information about the underlying dyeing process thereby helping in improvements in the dyeing services.

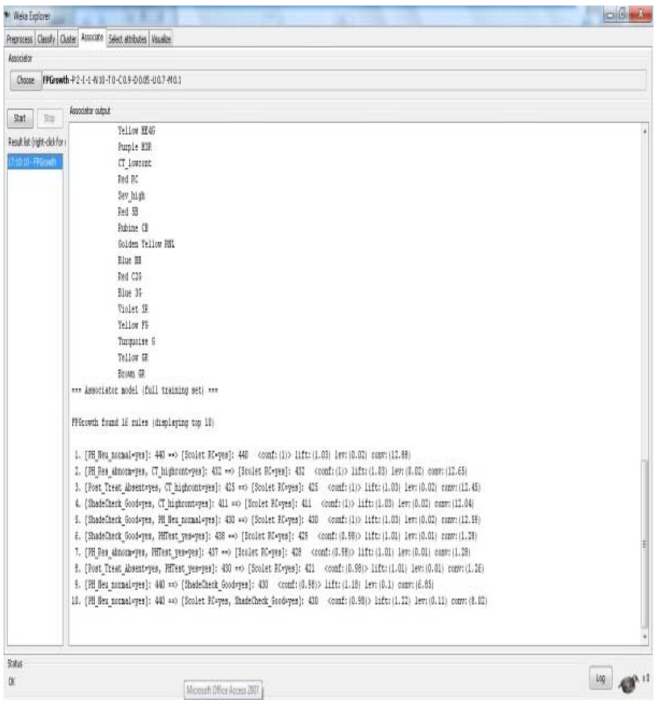
C. Illustration 3

For this experiment the same log as used for Illustration 1 and 2 is used. It has 683 process Instances (PIs), 70 different Event logs or ATEs and 12,294 total numbers of ATEs are used to illustrate the FPGrowth algorithm and H-Mine algorithm, it is shown in Figure 8 and 9 respectively.

The FPGrowth and H-Mine association rule mining algorithms uses the minimum support for the upper bound is set to 0.7 and the confidence set to 0.9. It is already identified that the Apriori algorithm produces 10 best association rules as shown in Figure 7. Hence, the same data is used with FPGrowth algorithm for mining the dyeing process, it produces 10 best association rules as shown in Figure 8.

Pattern growth can be viewed as successive computation of frequent 1-itemset of the database and conditional pattern bases and assembling them into longer patterns. Since computing frequent 1-itemsets is much less expensive than computing frequent 2-itemsets, the cost is substantially reduced. Second, since one transaction may contain many frequent itemsets. When there are many long transactions containing numerous frequent items the *FPGrowth* method constructs *FP-tree* which is a highly compact form of transaction database. Thus both the size and the cost of computation of conditional pattern bases, which corresponds roughly to the compact form of projected transaction databases, are substantially reduced.

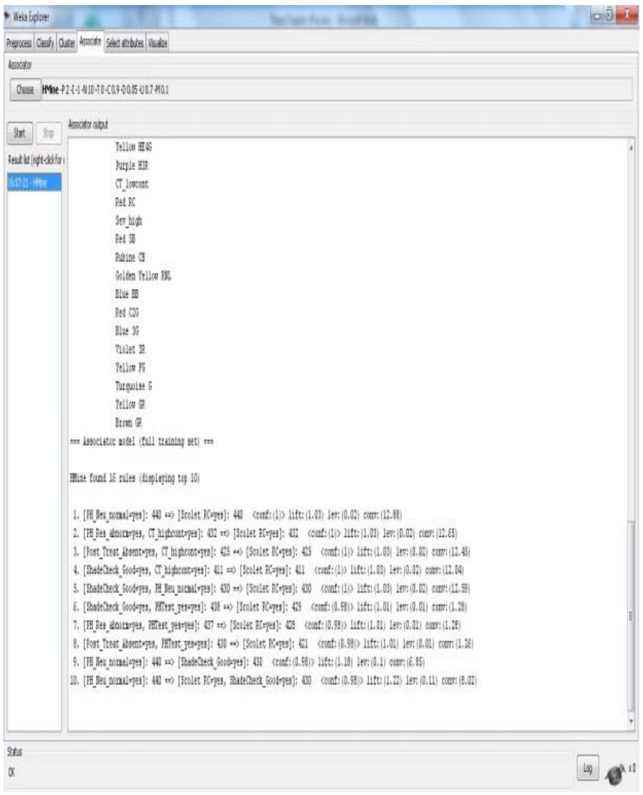
The H-Mine process mining algorithm is also used for the Emerald dyeing unit's dyeing process to depict the simple process model with the following parameter settings. The confidence value of the algorithm is set to 0.9, the minimum support of lower bound is set to 0.1 and the upper bound of the minimum support is set to 0.7. For these parameter settings the H-Mine algorithm yields the simple association rules. These rules were used to identify the various colour mix processes and treatment processes. The Figure 9 shows that the rule 1 specified that whenever the PH_neutral_normal occurs then must and should Scolet RC colour mix is used. Therefore the associations between these two treatments were used to understand the relationship between these two dyeing processes. Hence, it is necessary to obtain this type of analysis to identify the better dyeing process in future. It will help the dyeing expert or dyer to know the relationship between these two processes and it will help them to mix the dyes and to test various testings like pre PH and post PH treatments.



```
FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 0.7 -M 0.1
```

```
            Yellow HE4G
            Purple H3R
            CT_lowcont
            Red RC
            Sev_high
            Red 5B
            Rubine CB
            Golden Yellow RNL
            Blue BB
            Red C2G
            Blue 3G
            Violet 3R
            Yellow FG
            Turquoise G
            Yellow GR
            Brown GR
=== Associator model (full training set) ===

FPGrowth found 16 rules (displaying top 10)

 1. [PH_Neu_normal=yes]: 440 ==> [Scolet RC=yes]: 440    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.88)
 2. [PH_Bas_abnorm=yes, CT_highcont=yes]: 432 ==> [Scolet RC=yes]: 432    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.65)
 3. [Post_Treat_Absent=yes, CT_highcont=yes]: 425 ==> [Scolet RC=yes]: 425    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.45)
 4. [ShadeCheck_Good=yes, CT_highcont=yes]: 411 ==> [Scolet RC=yes]: 411    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.04)
 5. [ShadeCheck_Good=yes, PH_Neu_normal=yes]: 430 ==> [Scolet RC=yes]: 430    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.59)
 6. [ShadeCheck_Good=yes, PHTest_yes=yes]: 438 ==> [Scolet RC=yes]: 429    <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.28)
 7. [PH_Bas_abnorm=yes, PHTest_yes=yes]: 437 ==> [Scolet RC=yes]: 428    <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.28)
 8. [Post_Treat_Absent=yes, PHTest_yes=yes]: 430 ==> [Scolet RC=yes]: 421    <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.26)
 9. [PH_Neu_normal=yes]: 440 ==> [ShadeCheck_Good=yes]: 430    <conf:(0.98)> lift:(1.18) lev:(0.1) conv:(6.85)
10. [PH_Neu_normal=yes]: 440 ==> [Scolet RC=yes, ShadeCheck_Good=yes]: 430    <conf:(0.98)> lift:(1.22) lev:(0.11) conv:(8.02)
```

Figure 8. The simple process model by FPGrowth algorithm generated Association Rules from Weka tool with PIs = 683 and Event Logs =12,294



```
HMine -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 0.7 -M 0.1
```

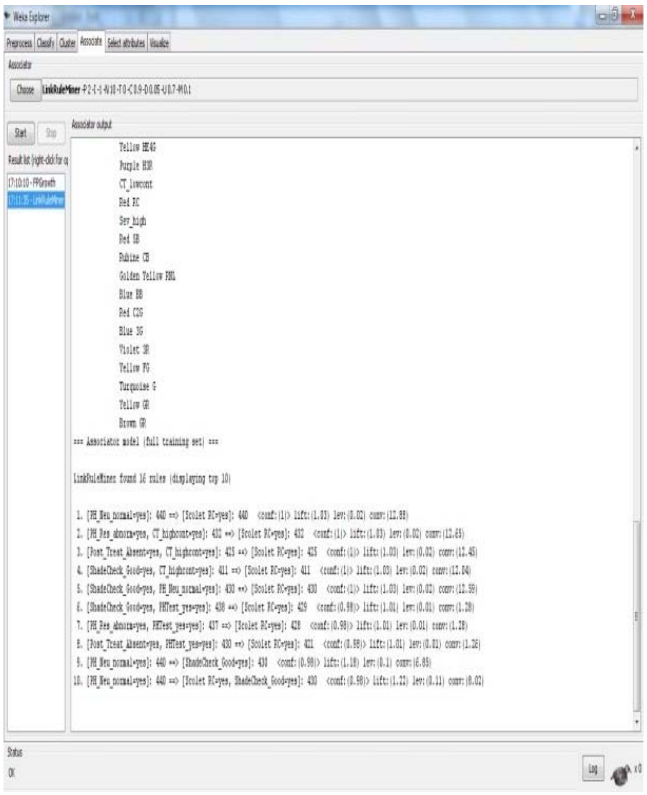
```
            Yellow HE4G
            Purple H3R
            CT_lowcont
            Red RC
            Sev_high
            Red 5B
            Rubine CB
            Golden Yellow RNL
            Blue BB
            Red C2G
            Blue 3G
            Violet 3R
            Yellow FG
            Turquoise G
            Yellow GR
            Brown GR
=== Associator model (full training set) ===

HMine found 16 rules (displaying top 10)

 1. [PH_Neu_normal=yes]: 440 ==> [Scolet RC=yes]: 440    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.88)
 2. [PH_Bas_abnorm=yes, CT_highcont=yes]: 432 ==> [Scolet RC=yes]: 432    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.65)
 3. [Post_Treat_Absent=yes, CT_highcont=yes]: 425 ==> [Scolet RC=yes]: 425    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.45)
 4. [ShadeCheck_Good=yes, CT_highcont=yes]: 411 ==> [Scolet RC=yes]: 411    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.04)
 5. [ShadeCheck_Good=yes, PH_Neu_normal=yes]: 430 ==> [Scolet RC=yes]: 430    <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.59)
 6. [ShadeCheck_Good=yes, PHTest_yes=yes]: 438 ==> [Scolet RC=yes]: 429    <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.28)
 7. [PH_Bas_abnorm=yes, PHTest_yes=yes]: 437 ==> [Scolet RC=yes]: 428    <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.28)
 8. [Post_Treat_Absent=yes, PHTest_yes=yes]: 430 ==> [Scolet RC=yes]: 421    <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.26)
 9. [PH_Neu_normal=yes]: 440 ==> [ShadeCheck_Good=yes]: 430    <conf:(0.98)> lift:(1.18) lev:(0.1) conv:(6.85)
10. [PH_Neu_normal=yes]: 440 ==> [Scolet RC=yes, ShadeCheck_Good=yes]: 430    <conf:(0.98)> lift:(1.22) lev:(0.11) conv:(8.02)
```

Figure 9: The simple process model by H-Mine algorithm generated Association Rules from Weka tool with PIs = 683 and Event Logs =12,294

D. Illustration 4

To illustrate the link structure of LRM, this section will help to understand the difference between the existing H-Mine process mining algorithm and proposed LRM process mining algorithm. The proposed LinkRuleMiner (LRM) algorithm produces the same 10 association rules as shown in Figure 10. The LRM algorithm has the effective results with the performance of less memory overhead and high speed. These LRM association rules would help us in obtaining simpler and understandable process models that give meaningful insights into the underlying process. The comparative study conducted between the existing and proposed association rule mining algorithms were discussed in the section 6.2.7. For this experiment the same log as used for Illustration 1, 2 and 3 is used. As it is know that it has 683 process Instances (PIs), 70 different Event logs and 12,294 total numbers of ATEs are used to illustrate the LinkRuleMiner algorithm. The minimum support for the upper bound is set to 0.7 and the confidence is set to 0.9 to illustrate this algorithm. Hence, the Figure 10 gives the process model generated by this algorithm.

The LinkRuleMiner algorithm modifies link structure and processing order of H-Mine algorithm. It has successfully removed link adjustment from H-Mine without any loss in performance and thus makes it possible to do parallelization on the algorithm efficiently. Therefore, the LinkRuleMiner algorithm that has no link adjustment made it possible to build parallel algorithm that requires no data exchange or any communications between each node in the middle of mining process. This allows us to build ideal parallel algorithm in shared disk environment and further to design parallel algorithm that exchanges data and communicates between each node in an efficient way in shared nothing environment.



LinkRuleMiner -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 0.7 -M 0.1

```
            Yellow HE4G
            Purple H3R
            CT_lowcont
            Red FC
            Sev_high
            Red SB
            Rubine CB
            Golden Yellow FNL
            Blue BB
            Red C2G
            Blue 3G
            Violet 3R
            Yellow FG
            Turquoise G
            Yellow GR
            Brown GR
=== Associator model (full training set) ===

LinkRuleMiner found 16 rules (displaying top 10)

 1. [PH_Neu_normal=yes]: 440 ==> [Scolet RC=yes]: 440   <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.88)
 2. [PH_Res_abnorm=yes, CT_highcont=yes]: 432 ==> [Scolet RC=yes]: 432   <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.65)
 3. [Post_Treat_Absent=yes, CT_highcont=yes]: 425 ==> [Scolet RC=yes]: 425   <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.45)
 4. [ShadeCheck_Good=yes, CT_highcont=yes]: 411 ==> [Scolet RC=yes]: 411   <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.04)
 5. [ShadeCheck_Good=yes, PH_Neu_normal=yes]: 430 ==> [Scolet RC=yes]: 430   <conf:(1)> lift:(1.03) lev:(0.02) conv:(12.59)
 6. [ShadeCheck_Good=yes, PHTest_yes=yes]: 438 ==> [Scolet RC=yes]: 429   <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.28)
 7. [PH_Res_abnorm=yes, PHTest_yes=yes]: 437 ==> [Scolet RC=yes]: 428   <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.28)
 8. [Post_Treat_Absent=yes, PHTest_yes=yes]: 430 ==> [Scolet RC=yes]: 421   <conf:(0.98)> lift:(1.01) lev:(0.01) conv:(1.26)
 9. [PH_Neu_normal=yes]: 440 ==> [ShadeCheck_Good=yes]: 431   <conf:(0.98)> lift:(1.18) lev:(0.1) conv:(6.85)
10. [PH_Neu_normal=yes]: 440 ==> [Scolet RC=yes, ShadeCheck_Good=yes]: 430   <conf:(0.98)> lift:(1.22) lev:(0.11) conv:(8.02)
```

Figure 10: The simple process model by LRM algorithm generated Association Rules from Weka tool with PIs = 683 and Event Logs =12,294

VI. CONCLUSION

The experimental results in the previous section favours the use of association rules for mining less structured processes like the dyeing processes. It should be noted that the above illustrations exhibit the use of data from Emerald dyeing process. The association rules would be evaluated on criteria of simplicity, ease of understanding and insights provided for the investigated process. It is clear from the above illustrations that the association rules are simple and easy to interpret, and they have the potential to provide knowledge in the form of behavioral or frequent patterns existing implicitly in the log. This information can be of importance to the dyeing personnel who can make anticipated preparations in terms of skills and equipments needed to deal with any emergency or similar situation for example, where A, B and C's absence implies execution of task C.

The behavioral pattern that is rules combined with the domain knowledge would be of great use to interpret and understand the various activities happening in any dyeing organization. The simplicity of these rules and the knowledge derived by them motivated us to further explore the domain of association rules and to get insights into how they can be of use in mining less structured processes. Currently the output of the CDE can be used for generating association rules from Weka but this process is not straightforward and consumes time as the CDE output has to be modified before it can be used in the Weka library for association rule generation.

In the illustration 1, it reviews the importance of association rule mining algorithm thru the experimentation with HM process mining algorithm and Apriori association rule mining algorithm. In the illustration 2, it identified that how clustering can be helpful to understand the complex process with ease using DWS process mining algorithm. Also this experimentation helps to know the cluster data to association rule mining data. In the illustration 3, it differentiates the association rule mining algorithms such as FPGrowth and H-Mine to understand the relationship between these two algorithms. In the illustration 4, the newly proposed association rule mining algorithm LinkRuleMiner is described with different parameter settings. Therefore, it is concluded that the dyer will benefit more from the association rule mining algorithms such as Apriori, FPGrowth, H-mine and LinkRuleMiner than the process mining algorithms such as Heuristic Miner (HM) and Disjunctive Workflow Schema (DWS) algorithms.

VII. REFERENCES

[1] A. Rozinat and W.M.P.van der Aalst. Decision mining in ProM. In S. Dustdar, J.L. Fiadeiro, A. Sheth, Business Process Management (Proceedings 4th International Conference, BPM 2006, Vienna, Austria, September 5-7, 2006) (Lecture Notes in Computer Science, Vol. 4102, pp. 420-425, Berlin: Springer.

[2] M. J. Berry, and G. Linoff. Data Mining Techniques: for Marketing, Sales, and Customer Support. John Wiley & Sons, Inc, 1997.

[3] I. H. Witten and E. Frank. Data Mining: Practical machine learning tools and techniques. 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[4] http://www.albionreserach.com/data_mining /market_basket.php

[5] Agrawal, R., Imielinski, T., and Swami, A. N. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207-216, 1993.

[6] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In Proc. of the 20th International Conference on Very Large Databases, Santiago, Chile, September 1994.

[7] Khellaf & Aomar. Mining Association Rules in Temporal Sequences, 2006, pp. 4-6.

[8] J. Han, H. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In: Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX). ACM Press, New York, NY, USA, 2000.

[9] T. Scheffer. Finding Association Rules That Trade Support Optimally against Confidence. In Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (September 03 - 05, 2001). L. D. Raedt and A. Siebes, Eds. Lecture Notes In Computer Science, vol. 2168. Springer-Verlag, London, pp. 424-435.

[10] Witten, I. H., and Frank E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, San Francisco, 1999.

[11] A.J.M.M. Weijters, W.M.P. van der Aalst, and A. K. Alves de Medeiros. Process Mining with the Heuristics Miner Algorithm. BETA Working Paper Series, WP166, Eindhoven University of Technology, Eindhoven, 2006.