# Data Preparation in the *MineCor* KDD Framework

Christian Ernst
*Ecole des Mines de St Etienne*
*CMP - Site Georges Charpak*
*Gardanne, France*
*ernst@emse.fr*

Alain Casali
*Laboratoire d'Informatique Fondamentale de Marseille (LIF),*
*CNRS UMR 6166, Aix Marseille Universités, IUT d'Aix en Provence*
*Aix en Provence, France*
*alain.casali@lif.univ-mrs.fr*

***Abstract*—Yield enhancement is a key issue in semiconductor manufacturing. Data mining tools can therefore be helpful, by extracting hidden links between numerous complex process control parameters. In order to highlight correlations between such parameters, we developed a complete Knowledge Discovery in Databases (KDD) model, called *MineCor*. Its mining heart uses a new method derived from association rules programming, based on lectic search and contingency vectors. After recalling these concepts, this paper focuses on data preprocessing and transformation functions, which have an important impact on final results. An overall presentation of these functions, of some significant experimental results and of associated performances are provided and finally discussed.**

***Keywords*-Data Mining, Semiconductor Manufacturing, Decision Correlation Rule, Data Preparation.**

## I. Introduction and motivation

In this Section, we first introduce why and how data mining techniques are useful to detect the main parameters, which have an impact on yield loss in semiconductor fabrication capabilities. Then, we present our approach, based on a complete KDD model. It determines the main correlated production parameters impacting the yield, and is based on important preliminary preparation tasks.

### *A. Data mining techniques in semiconductor fabs*

Data Mining specifies data models, which may be rules, anomalies or trends that are of interest. To improve quality in manufacturing areas, mining techniques extract knowledge to identify hidden patterns in the parameters that control production processes [1]. Unfortunately, there are no scalable models for associated applications, but only "implementation specific" mining algorithms. We focus hereafter on $(i)$ Fault detection and quality improvement, which examine what happened in the past to better predict and to improve the future system's performance; and $(ii)$ Decision support systems, which determine links between control parameters and product quality in the form of rules. We moreover concentrate on a particular area, semiconductor wafer manufacturing, where yield is the ratio of non-defective chips in a finished wafer to the number of input products.

In semiconductor manufacturing facilities, the volume and the complexity of the collected data are generally much more consequent than in other manufacturing fields: Fabrication processes include several hundred steps with regard to the produced chip. Each step uses various chemico-physical recipes, grouped into four phase units (photolithography, etch, implant and CMP).

Two techniques are used to improve the yield: Real-time and *post hoc*. The first approach monitors on-line measurements of process steps, and undertakes corrective action to ensure that the measures remain within desired limits. The *post hoc* approach compares the end result of the whole process with the desired specifications, analyzing the root causes of low yield for adjusting the process parameters to ensure future quality. Advanced Process Control (APC) considers both, by highlighting correlations between production parameters in order to rectify possible drifts of the associated process(es). This can be done for specific equipment and process steps in real-time: FDC (Fault Detection and Classification) tools and R2R (Run To Run) regulation loops are the most representative APC techniques. Correlations can also be discovered *post hoc*: This is the framework of our paper.

Both approaches first try to identify, which parameters are the root causes of a particular yield excursion. However, conventional methods such as SPC are here inaccurate, because they fail to extract underlying features from complex data. This is why data mining techniques become useful in semiconductor fabs. Associated models can be categorized into four types [2]: Classification, Clustering, Prediction and Association Rules. The most widely used is classification. At the contrary, association rules are not often used. Let us mention [3], where the authors present a modified *Apriori* algorithm used in LCD panel manufacturing in order to locate machines with low yield after process completion.

### *B. Our approach*

We present a whole KDD model based on specific rules. Within this framework, and in collaboration with STMicroelectronics and ATMEL, our work is focused on the detection of the main control parameters impacting the yield. The goal is to propose indicators to which special attention should be paid in order to construct, in a second step, yield enhancement models used in further production cycles. Let us emphasize that this second non-trivial problematic is excluded from the scope of our paper.

The realized *post hoc* analysis is based on CSV files of real valued measurements associated with production lots, extracted from large databases and covering the four fabrication units mentioned above. The main characteristic of these files is the huge number of columns (nature of the measurements) with regard to the number of rows (measures). We want to highlight correlations between the values of some columns and those of a target column: The yield. To detect these correlations, we use Decision Correlation Rules [4] but, before, we undertake important data preparation tasks to enhance the efficiency of the mining input database: This is the scope of our paper.

The paper is organized as follows: In Section II, the bases of Decision Correlation Rules and our mining algorithm are first recalled. In Section III, we expose the data preparation functions of the *MineCor* software. Experiments are detailed in Section IV. As a conclusion, we summarize our contributions and outline some research perspectives.

## II. RELATED WORK:THE MINECOR MINING MODEL

In this section, we first recall the definitions of correlation rules and lectic order. Then we introduce the LS Algorithm, which allows to browse the search space according to the lectic order, before presenting the LHS-CHI2 algorithm. Some points developed hereafter have soon been presented in [4]. But the given overview clarifies the approach.

### A. Decision correlation rules and lectic order

**Basic concepts**

An association rule [5] is an approximate implication $X \rightarrow Y$ between two sets of items. Two measures are used to extract significant rules: Support and confidence. Because the underlying semantics of an association rule are fairly poor, Wu et al. [6] introduce literalsets and compute positive and/or negative association rules such as $\neg X \rightarrow Y$. To generate the rules, the authors use the same platform by redefining the support of a literal: The number of transactions of the binary relation including $X$ and containing no 1-item of $Y$. Another approach is proposed by Brin *et al.* [7]: The extraction of correlation rules. The new platform is based on the Chi-Squared statistical measure, written $\chi^2$. We assume hereafter that the definitions of literalsets so as of the $\chi^2$ statistic are known.

When computing correlation rules, the memory usage required by levelwise algorithms is crucial. This is why Brin *et al.* compute only correlations between two values. Different criteria to evaluate whether a correlation rule is semantically correct have been proposed. The main is the Cochran criterion, and can be relaxed as follows: $MinPerc$ of the literalsets of a contingency table must have a support larger than $MinSup$, where $MinPerc$ (minimal percent) and $MinSup$ (minimal support) are thresholds.

**Decision correlation rules**

We want the computed correlation rules to include specific items, *e.g.* belonging to a target attribute. Let $r$ be a binary relation (a transaction database) over a set of items $\mathcal{R} = \mathcal{I} \cup T$. $\mathcal{I}$ represents the values (the items) of the binary relation used as analysis criteria, and $\mathcal{T}$ is a target attribute, which items may be null.

*Definition 1 (Decision Correlation Rules):* Let $X \subseteq \mathcal{R}$ be a pattern, and $MinCorr$ a given threshold ($\geq 0$). $X$ represents a valid Decision Correlation Rule if and only if: ($i$) $X$ contains a value of the target attribute $\mathcal{T}$; and ($ii$) $\chi^2(X) \geq MinCorr$.

Table I
RELATION EXAMPLE $r$.

| Tid | ItemSet | Target |
|---|---|---|
| 1 | BCF | $t_1$ |
| 2 | BCE | $t_1$ |
| 3 | BCF | $t_2$ |
| 4 | BC | - |
| 5 | BD | $t_1$ |
| 6 | B | - |
| 7 | ACF | $t_1$ |
| 8 | AC | - |
| 9 | AE | $t_1$ |
| 10 | F | $t_2$ |

*Example 1:* With the relation Example $r$ given in Table I, Table II shows the contingency table of pattern $BC$.

Table II
CONTINGENCY TABLE OF PATTERN $BC$.

| | $B$ | $\overline{B}$ | $\sum_{row}$ |
|---|---|---|---|
| $C$ | 4 | 2 | 6 |
| $\overline{C}$ | 2 | 2 | 4 |
| $\sum_{column}$ | 6 | 4 | 10 |

Continuing the example, $\chi^2(BC) \simeq 0.28$. which corresponds to a correlation rate of about 45%. If $MinCorr = 0.25$, the correlation rule materialized by the $BC$ pattern is valid, but the correlation rule represented by the $Bt_1$ pattern is not ($\chi^2(Bt_1) \simeq 0.1$).

**Lectic order and Lectic subset algorithm**

The lectic order [8], noted $<_{lec}$, permits to enumerate all the subsets of an itemset $\mathcal{I}$.

*Definition 2 (Lectic Order):* Let $\mathcal{I}$ be a set of items totally ordered and therefore comparable two by two via an order denoted by $\preceq$. If $X$ and $Y \subseteq \mathcal{I}$, then we have: $X <_{lec} Y \Leftrightarrow max_{\preceq}(X \backslash (X \cap Y)) \preceq max_{\preceq}(Y \backslash (X \cap Y))$.

The Lectic Subset Algorithm, noted LS [9], is one of its possible implementations.

## B. The LHS-Chi2 Algorithm

**Equivalence classes**

In [4], we adapted the concept of equivalence classes over literal patterns to our context:

*Definition 3 (Equivalence Class associated with a literal):* Let $Y\overline{Z}$ be a literal, and $[Y\overline{Z}]$ its associated equivalence class. This class contains the set of transaction identifiers of the relation including $Y$ and containing no value of $Z$.

*Example 2:* With our relation Example (see Table 1), we have $[B\overline{C}] = \{5, 6\}$.

**Contingency vectors**

The contingency vectors are another representation of the contingency tables:

*Definition 4 (Contingency Vector):* Let $\mathbb{P}(X)$ be the powerset lattice of $X$, and $X \subseteq \mathcal{R}$ a pattern. The contingency vector of $X$, denoted $CV(X)$, groups the set of the literalset equivalence classes belonging to $\mathbb{P}(X)$ ordered according to the lectic order.

*Example 3:* With our sample relation (see Table 1), we obtain $CV(BC) = \{[\overline{BC}], [B\overline{C}], [C\overline{B}], [BC]\} = \{\{9, 10\}, \{5, 6\}, \{7, 8\}, \{1, 2, 3, 4\}\}$.

Proposition 1 is the main result presented in [4]. It shows how to compute the $CV$ of the $X \cup A$ pattern given the $CV$ of $X$ and a set of identifiers of the relation containing $A$.

*Proposition 1:* Let $X \subseteq \mathcal{R}$ be a pattern and $A \in \mathcal{R}\backslash X$ a 1-item. The CV of the $X \cup A$ pattern can be computed given the CVs of $X$ and $A$ as follows:

$$CV(X \cup A) = (CV(X) \cap [\overline{A}]) \cup (CV(X) \cap [A])$$

*Example 4:* With the relation Example (see Table 1), we have $CV(B) = \{\{7, 8, 9, 10\}, \{1, 2, 3, 4, 5, 6\}\}$ and $CV(C) = \{\{5, 6, 9, 10\}, \{1, 2, 3, 4, 7, 8\}\}$. By applying Proposition 1 and ordering, we retrieve the result of Example 3: $CV(BC) = \{\{9, 10\}, \{5, 6\}, \{7, 8\}, \{1, 2, 3, 4\}\}$.

**The LHS-CHI2 Algorithm**

The Lectic Hybrid Subset-Chi2 Algorithm, or LHS-CHI2, adapts the LS Algorithm to our context, in the way that it includes contingency vectors so as five constraints in order to prune the search space [4]. The predicate *CtPerc* expresses the satisfiability of the Cochran criterion. The pseudo-code of the procedure CREATE_CV can be found in [4]. By convention, we consider that we have $CV(\emptyset) = \{Tid(R),\ \emptyset\}$. The positive border ($BD^+$) is initialized with $\{\emptyset\}$. The pseudo code of LHS-CHI2 is provided in Algorithm 1. The first call to LHS-CHI2 is carried out with $X = \emptyset$ and $Y = \mathcal{R}$.

*Example 5:* The Decision Correlation Rules computed by LHS-CHI2 on the Example of Table 1 and satisfying the minimal threshold constraints $MinSup = 0.2$, $MinPerc = 0.25$ and $MinCorr = 0.25$ are shown in Table III.

**Algorithm 1:** LHS-CHI2 Algorithm.

```
input : $X$ and $Y$ two patterns
output: { $Z \subseteq X$ such that $\chi^2(Z) \geq MinCorr$ }
1  if $Y = \emptyset$ and $\exists t \in \mathcal{T} : t \in X$ and $|X| \geq 2$ and $\chi^2(X) \geq MinCorr$ then
2      Output X, $\chi^2(X)$
3  end
4  $A := max(Y)$ ;
5  $Y := Y\backslash\{A\}$ ;
6  LHS-CHI2(X,Y) ;
7  $Z := X \cup \{A\}$ ;
8  if $\forall z \in Z, \exists W \in BD^+ : \{Z\backslash z\} \subseteq W$ then
9      CV(Z) := CREATE_CV(VC(X),Tid(A)) ;
10     if $|Z| \leq MaxCard$ and $CtPerc(CV(Z), MinPerc, MinSup)$ then
11         $BD^+ := max_{\subseteq}(BD^+ \cup Z)$ ;
12         LHS-CHI2(Z,Y) ;
13     end
14 end
```

Table III
RESULTS OF THE LHS-CHI2 ALGORITHM OVER TABLEI

| Decision Correlation Rule | $\chi^2$ Value |
|---|---|
| $At_1$ | 0.48 |
| $BCt_1$ | 0.28 |
| $BFt_1$ | 0.28 |

**Performance issues**

We show in [4] that levelwise algorithms require to store CTs 2.5 GB of memory at the 3rd level, and 1.3 TB at the 4th level. When our algorithm requires, in the worst case, $2|r| * (MaxCard + n + 1)$ Bytes in memory, where $n$ is the number of 1-items in the relation $r$. Which is much less than the mentioned volumes above. This is because we need only to keep the CVs stored in each branch of the search tree, the computation of a CV at each level using the CVs memorized at the upper levels. When a branch has been pruned, all the stored CVs in that branch are released. Moreover, computing a $CV$ is faster than computing a $CT$.

## III. DATA PREPARATION WITHIN *MineCor*

We developed a global KDD model including the LHS-Chi2 algorithm. The software, called *MineCor* (*Mine*r for *Cor*relations), is developed in C language. To carry out preprocessing and transformation in the form of a transaction database of the input files, we first performed column elimination and discretization stages [2], [10]. Description of these steps are our main contribution: seldom discussed nor presented in an industrial context, they have a huge impact on final results, and are summarized in Sections III-A and III-B. The output of the two steps is the source for the data mining phase. Which is followed by an interpretation of the results, resumed in Section III-C.

### *A. Preprocessing stage*

The first step of data cleaning is the preprocessing stage. Preprocessing consists in the reduction of the data structure [11] by eliminating columns and rows of low significance. This is for two reasons: ($i$) If each value of each column is considered as a single item, there will be a combinatorial explosion of the search space, and thus very large reponse times; ($ii$) We cannot expect this task to be performed by an expert, because manual cleaning of data is time consuming and subject to many errors. The defined order of following functions is important.

**Elimination of Concentrated Data and Outliers**

We eliminate first columns having small standard deviation (threshold $MinStd$): Since the values are almost the same, we consider that they do not have a significant impact on the result; but their presence pollutes the search space and reduces the response times. In the same way, we introduced the $MinDV$ threshold, which imposes a Minimal number of Distinct Values in each column. Attention is finally paid to inconsistent values, such as "outliers" in noisy columns. Detection is performed through another convenient threshold ($fStd$, a factor of $MinStd$), and elimination consists to force the detected values to null.

**Other Column Elimination**

The dysfunction of sensors, or the occurrence of a maintenance step may imply that some sensors can not transmit their values to the database. As a consequence, the associated columns contain many null/default values, and are thus deleted from the input file. Such cases are here detected using the $MaxNV$ (Maximum Null Values) threshold. Moreover, sometimes, several sensors measure the same information, resulting in identical columns in the source file. In this case, we keep only a single column. Finally, columns with no item having the support ($MinSup$ threshold) are also removed.

**Normalization**

Let $\mathcal{S}$ be the set of values to be discretized (the input column values as a numeric vector), and $Min_{\mathcal{S}}$ and $Max_{\mathcal{S}}$ be the smallest and the largest value of $\mathcal{S}$. Finally, and in order to manage the different values associated with each set $S$ in the same way, we normalize the values to keep them between 0 (kZero) and 1 (kOne). This is performed by replacing each value $v \in \mathcal{S}$ by $\frac{v-Min_{\mathcal{S}}}{Max_{\mathcal{S}}-Min_{\mathcal{S}}}$.

### *B. Discretization stage*

Discrete values deal with intervals of values, which are more concise to represent knowledge, so that they are easier to use and comprehend than continuous values. Many discretization algorithms have been proposed over the years in order to classify data into intervals, also called bins. Discretization can thus be performed [12]:

- In a supervised or unsupervised manner, depending on whether class information is at one's disposal;
- In a dynamic or static way: With a static approach, discretization is done before the classification task;
- Using splitting or merging techniques: In the latter case, the search space is examined bottom-up.

We represent continuous real valued columns by associating to each of their values an interval code (the one to which the value belongs). The intervals are created using either equal-width, equal-frequency and embedded means discretization, which are non supervised, static and splitting methods. In each approach, $NIC$ is an input parameter specifying the number of bins to create per column.

**Equal Width Discretization (EWD)**

Each interval has a length of $l = \frac{Max_{\mathcal{S}}-Min_{\mathcal{S}}}{NIC}$. The computed classes are $c_1 : [Min_{\mathcal{S}}, Min_{\mathcal{S}} + l[$, $c_2 : [Min_{\mathcal{S}} + l, Min_{\mathcal{S}} + 2l[$, .... The method is easy to compute and to interpret, but is not efficient in the case of asymmetric or discontinuous distributions.

**Equal Frequency Discretization (EFD)**

The goal is to obtain classes having, if possible, the same number of values. Because this configuration seldom appears, the problem becomes how to group close values into classes while respecting the above constraint. The Jenks' natural breaks classification schema gets the best class arrangement, after having generated each possible class combination [13]. It minimizes the in-class difference and maximizes the between-class difference using the Goodness of Variance Fit (GVF):

$$GVF = 1 - \frac{\sum_{j=1}^{NIC}\sum_{i=1}^{|[\mathcal{S}_i,\mathcal{S}_j]|}(\mathcal{S}_i - \overline{[\mathcal{S}_i,\mathcal{S}_j]})^2}{\sum_{i=1}^{|[\mathcal{S}]|}(S_i - \overline{\mathcal{S}})^2},$$

where $|[\mathcal{S}_i, \mathcal{S}_j]|$ is the cardinality of the interval $[\mathcal{S}_i, \mathcal{S}_j]$, and $\overline{\mathcal{S}}$ is the mean of the sorted set $\mathcal{S}$.

The main drawbacks of the method are, on one hand, that stability is not assumed when $NIC$ varies and, on the other hand, the high computational complexity of the class generation, which is $C_{d-1}^{NIC-1}$, where $d$ is the number of distinct values in the set $\mathcal{S}$. The associated computational cost becoming redhibitory, we use instead the Fisher's method of exact optimization [14] proposed for grouping $|[\mathcal{S}]|$ elements (distinct or not) into $NIC$ mutually exclusive and exhaustive subsets having maximum homogeneity, *i.e.,* minimizing the within-groups sum of squares. The obtained partition is guaranteed to be optimal, but not unique. Which is not important while the obtained gain of time is.

**Embedded Means Discretization (EMD)**
EMD is a divisive hierarchical clustering method, which starts with a single class/bin that contains all the initial values of the column to discretize ($\mathcal{S}$). The average divides the set into two groups used to construct two new classes. The averages of these 2 groups permit the splitting into 4 classes, and so on. This approach matches every kind of distribution, but the number of classes $NIC$ is here always a power of two, which may be an inconvenient.

**Algorithm 2:** REM Algorithm.

```
input : $Vec$ : initial vector, $Min, Max$ 2 borders,
        $ind$ : current index, $nbi$ : number of bins
output: $MVec$ : average vector
1 $Avg := ComputeAvg(Min, Max, Vec)$ ;
2 $MVec[Ind] := Avg$ ;
3 if $NIC > 1$ then
4     REM $(Vec, Min, Avg, ind - nbi/2, nbi/2)$ ;
5     REM $(Vec, Avg, Max, ind + nbi/2, nbi/2)$ ;
6 end
```

We compute the class borders ($MVec$) using the Recursive Embedded Means (REM) discretization algorithm, which pseudo code is provided in Algorithm 2. The first recursive call to REM is carried out with $Vec = \mathcal{S}$, $Min = kZero$, $Max = kOne$, $ind = NIC/2$ and $nbi = NIC/2$. The $ComputeAvg$ function returns the average of the $Vec$ values bordered by $Min$ and $Max$.

**String Valued Columns Discretization**
Because string valued columns often appear as parameter measures, we take them equally into account by applying them a rough discretization function presented in Algorithm 3. Associated Compute String Intervals (CSI) discretization method keeps the $NIC$ or $NIC - 1$ most present string literals of input string vector $\mathcal{S}$ as discretization "values". If the $bOth$ boolean is enabled, $OutVec[NIC - 1]$ represents any other value of $\mathcal{S}$ not equal to any of the first $NIC - 1$ values of $OutVec$. The column is removed for the mining step if it contains strictly less than $NIC$ values.

### *C. Interpretation stage*

Interpretation essentially consists in decoding the discretization stage with regard to the results, and to produce an intelligible output for the end-user. *MineCor* produces outputs in HTML and text formats.

*Example 6:* $BCt_1$ is a valid Decision Correlation Rule (cf Table III). Associated text output looks like [1.4, 2]; [2.8, 4.6]; [2.7, 7.4], where $[b_{min}, b_{max}]$ are real values representing items $B, C, t_1$ respectively.

## IV. Experimental Analysis

Some representative results of the LHS-Chi2 algorithm are presented below. As emphasized in Section I-B, the

**Algorithm 3:** CSI Algorithm.

```
input : $nbi$ : number of bins , $InVec$ : input string
        vector, $bOth$ : "others" boolean, $Minsup$ :
        threshold
output: $OutVec$ : output string vector
1 $OutVec := \emptyset$ ;
2 $nbDisVals := SortByPopularity(InVec, OutVec)$;
3 if $nbDisVals \geq NIC$ and $bOth$ and
  $Sup(OutVec[nbi - 1]) \leq MinSup$ then
4     $OutVec[nbi - 1] := kOthers$ ;
5 end
```

experiments were done on different CSV files of (essentially) real value measures supplied by STMicroelectronics (STM) and ATMEL (ATM). The files have one or more target columns, resulting from the concatenation of several measurement files. The characteristics of the datasets used can be found in Table 4. STM File1 and ATM File are representative of our *post hoc* approach (*cf.* Section I-A). STM File2 is more typical of a real-time approach (few parameters, large number of measures).

Table IV
Dataset Examples

| Name | Number of Columns | Number of Rows |
|---|---|---|
| STM File1 | 1 281 | 297 |
| STM File2 | 8 | 726 |
| ATM File | 749 | 213 |

All experiments were conducted on an HP Workstation (1.8 GHz processor with a 4 Gb RAM). Results are presented on Figures 1 through 7(b).

In [4], we compare the execution times of a classical Levelwise algorithm and LHS-Chi2. The experiments use the only EWD method. The response times of our method are between 30% and 70% better than Levelwise.

### *A. Impact of the Preprocessing Stage*

Figures 1 and 2 show respectively the number of items used in the mining stage and the number of decision correlation rules discovered after that stage for STM File2 (target1) when $MinSup$ (0.2), $MinCorr$ (0.24) and $MinPerc$ (1.2) are constant, while the number of bins ($NIC$) varies. We compare the three discretization methods.

EMD (and EFD) are better methods than EWD when the $NIC$ parameter remains low. With greater values of $NIC$ and also because of the large number of rows in the analyzed file, EWD returns more rules (even if not necessary all of interest). As it appears also on the following experiments, EWD is the best method the larger the thresholds, because of the more important number of columns kept and thus of

Figure 1. Number of items used *vs.* number of bins.

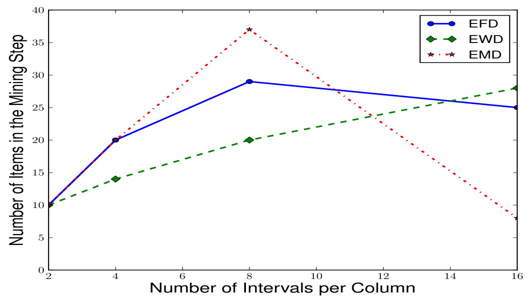
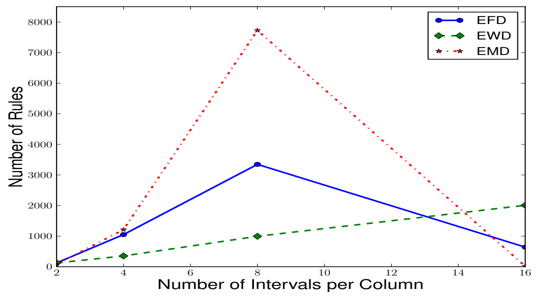


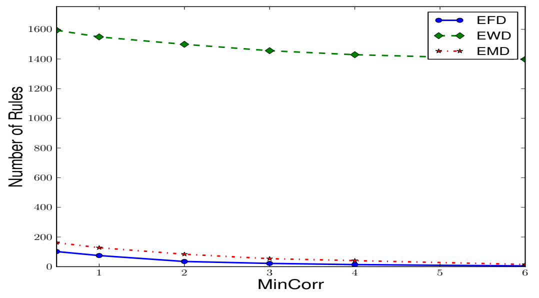
Figure 2. Number of generated rules *vs.* number of bins.



items produced after preprocessing, even if the execution times are more important.

Figures 3 and 4 show the number of extracted rules after mining when $MinPerc$ (0.34) and $MinSup$ (0.25 in Figure 3 and 0.27 in Figure 4) are fixed. The difference between the two experiments is that the $MinDV$, $MaxNV$ and $MinStd$ thresholds are smaller in Figure 4 than in Figure 3. What harmonizes the results, and shows also that the $MinCorr$ threshold has only few effect on mining. What means also that the preprocessing parameters impact the numbers of obtained items and thus of computed rules.

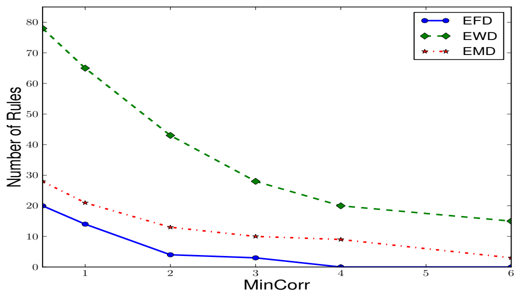
Figure 3. Number of generated rules with $MinDV = 58$, $MaxNV = 48$, $MinStd = 1.68$ (STM File1 - target1).



## B. Impact of the Discretization Stage

Figures 5(a) and 6(a) show the number of items kept after the discretization stage, which only depends on the $MinSup$ threshold, while the number of bins is constant. They illustrate that the smaller the threshold $MinSup$, the larger the number of items kept for the mining stage, whatever the discretization method. Figures 5(b) and 6(b) show the number of rules that are generated in both cases. While the number of partitions generated by the EFD method is larger than the one generated by the EWD method, the number of rules is smaller. Moreover, the execution time is shorter by a factor up to 2.5 (*cf.* Figure 5(c)). On the other hand, the EMD method provides better results when working on STM File2, which is a perticular case. These results outline that *MineCor* tries to provide the end-user with "best" quality rules: ($i$) Low in number, ($ii$) Significant, and ($iii$) Computed quickly.

Figure 4. Number of generated rules with $MinDV = 80$, $MaxNV = 50$, $MinStd = 1.28$ (STM File1 - target1).



## V. Conclusion and Future Work

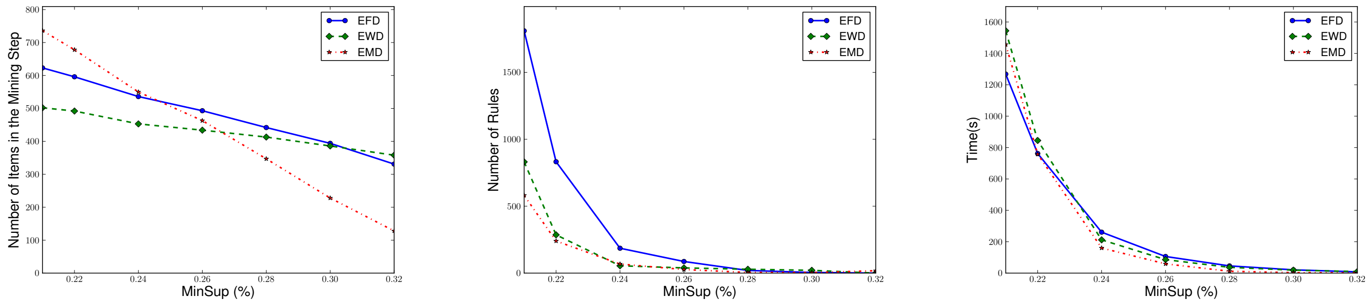
In this paper, we presented the *MineCor* software. Parameter measurement files given by semiconductor manufacturers are used, and produce values of parameters with most influence on the yield. To achieve this objective, we built a complete KDD model, based on Decision Correlation Rules. We show that the various thresholds used in our preprocessing stage have an impact on the number of kept columns of the input file, and thus, on the number of items used in further steps. Moreover, we implemented three methods at the discretization stage: ($i$) Equal Width, ($ii$) Equal Frequency, and ($iii$) Embedded Means. Experiments point out that, in most cases, the EFD method produces Decision Correlation Rules faster and of better quality.

Some new issues to our work are: ($i$) To compare our approach with classification methods; ($ii$) To optimize the processing stages upstream of the algorithm (aggregation of attributes, merging of intervals) while safeguarding the context in order to obtain a larger number of rules and/or more significant results; and ($iii$) To allow automatic threshold and parameter fixing depending on each input file column.

## References

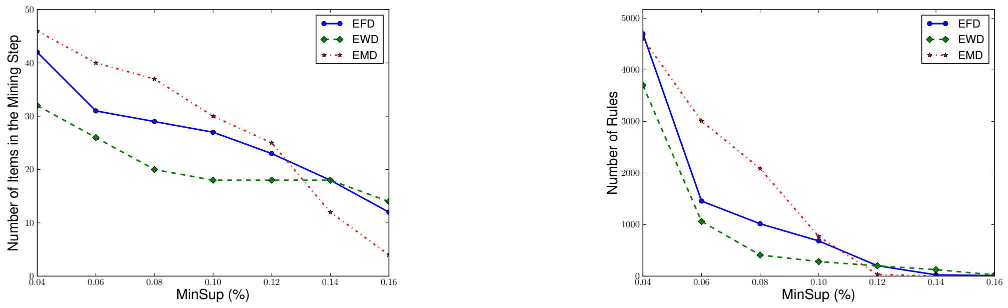
[1] A. Choudhary, J. Harding, and M. Tiwari, "Data mining in manufacturing: a review based on the kind of knowledge," *Journal of Intelligent Manufacturing*, vol. 20, no. 5, pp. 501–521, 2009.

Figure 5. Results with 4 intervals, $CtPerc = 0.34$, $MinCorr = 2.8$ (ATM File - target3).



(a) Number of items kept after the Preprocessing and Discretization stages (b) Number of generated Decision Correlation Rules (c) Execution Time

Figure 6. Results with 8 intervals, $CtPerc = 0.24$, $MinCorr = 1.6$ (STM File2 - target1).



(a) Number of items kept after the Preprocessing and Discretization stages

(b) Number of generated Decision Correlation Rules

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[3] C. Huang and R. Chen, "Application of new apriori algorithm mdnc to tft-lcd array manufacturing yield improvement," *International Journal of Computer Applications in Technology*, vol. 28, pp. 161–168, 2007.

[4] A. Casali and C. Ernst, "Extracting decision correlation rules," in *DEXA*, ser. Lecture Notes in Computer Science, S. S. Bhowmick, J. Küng, and R. Wagner, Eds., vol. 5690. Springer, 2009, pp. 689–703.

[5] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," in *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996, pp. 307–328.

[6] X. Wu, C. Zhang, and S. Zhang, "Efficient mining of both positive and negative association rules," *ACM Trans. Inf. Syst.*, vol. 22, no. 3, pp. 381–405, 2004.

[7] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *SIGMOD Conference*, 1997, pp. 265–276.

[8] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[9] M. Laporte, N. Novelli, R. Cicchetti, and L. Lakhal, "Computing full and iceberg datacubes using partitions," in *ISMIS*, 2002, pp. 244–254.

[10] D. Pyle, *Data Preparation for Data Mining*. Morgan Kaufmann, 1999.

[11] O. Stepankova, P. Aubrecht, Z. Kouba, and P. Miksovsky, "Preprocessing for data mining and decision support," in *Data Mining and Decision Support: Integration and Collaboration*, K. A. Publishers, Ed., 2003, pp. 107–117.

[12] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An enabling technique," *Data Min. Knowl. Discov.*, vol. 6, no. 4, pp. 393–423, 2002.

[13] G. Jenks, "The data model concept in statistical mapping," in *International Yearbook of Cartography*, vol. 7, 1967, pp. 186–190.

[14] W. Fisher, "On grouping for maximum homogeneity," in *Journal of the American Statistical Association*, vol. 53, 1958, pp. 789–798.