Discovering correlated parameters in Semiconductor Manufacturing processes: a Data Mining approach

Christian Ernst, Alain Casali

Discovering Correlated Parameters in Semiconductor Manufacturing Processes: A Data Mining Approach

Alain Casali and Christian Ernst

***Abstract*—Data mining tools are nowadays becoming more and more popular in the semiconductor manufacturing industry, and especially in yield-oriented enhancement techniques. This is because conventional approaches fail to extract hidden relationships between numerous complex process control parameters. In order to highlight correlations between such parameters, we propose in this paper a complete knowledge discovery in databases (KDD) model. The mining heart of the model uses a new method derived from association rules programming, and is based on two concepts: decision correlation rules and contingency vectors. The first concept results from a cross fertilization between correlation and decision rules. It enables relevant links to be highlighted between sets of values of a relation and the values of sets of targets belonging to the same relation. Decision correlation rules are built on the twofold basis of the chi-squared measure and of the support of the extracted values. Due to the very nature of the problem, levelwise algorithms only allow extraction of results with long execution times and huge memory occupation. To offset these two problems, we propose an algorithm based both on the lectic order and contingency vectors, an alternate representation of contingency tables. This algorithm is the basis of our KDD model software, called *MineCor*. An overall presentation of its other functions, of some significant experimental results, and of associated performances are provided and discussed.**

***Index Terms*—Chi^2 correlation statistic, data mining, decision rule, semiconductor manufacturing.**

I. Introduction and Motivation

IN THIS SECTION, we first introduce why and how data mining techniques are useful to enhance semiconductor fabrication capabilities. Discussion is set on how to detect the main parameters which have an impact on yield loss rather than on how to improve final yield. Then we present our approach that determines the main correlated production parameters impacting the yield.

A. Data Mining Techniques in the Manufacturing Industry

Data mining [1] allows us to extract data in terms of models which may be rules, concepts, patterns, anomalies, or trends that are useful and intelligible for the end-user. Nowadays, databases or data warehouses of significant size implicitly contain a large amount of relevant information. Their extraction presents an interest in various domains such as marketing, design, medical research [2], telecommunication networks [3], dynamic restructuring of websites [3], manufacturing sciences [4], and so on.

Data mining models can be categorized into four types [1]: classification, clustering, prediction, and association rules. Such approaches have been widely carried out in manufacturing areas [4]. Data mining extracts knowledge to identify hidden patterns in the parameters that control manufacturing processes or to determine and to improve product quality. Unfortunately, there is no standard scalable model for manufacturing applications. The models used are a collection of "implementation specific" data mining algorithms. Associated applications can be roughly divided into six categories.

1) Customer relationships: the objective is to develop the relationship with the customers in order to maximize profits.
2) Engineering design: based on historical data, the goal is to optimize design specification by matching the temporal data of a new product with the knowledge base.
3) Manufacturing systems: in such environments, the need and importance of data is ever present for statistical process control (SPC) purposes; SPC consists in effective statistical methods for monitoring a process through the use of control charts, by enabling the use of objective criteria for distinguishing background variation from events of significance.
4) (Equipment) maintenance: since databases contain information to improve processes, they also contain the reasons for machine failures.
5) Fault detection and quality improvement: examining what happened in the past is used to better understand the process, and therefore to predict and to improve the future system's performance. Virtual metrology [5] is here one of the most novel tools.
6) Decision support systems: the goal is to determine links between control parameters and product quality, essentially in the form of (decision) rules.

We focus hereafter on the last two points, which deal with quality, and thus implicitly with product yield. Moreover, we

Manuscript received September 2, 2010; revised March 21, 2011; accepted July 31, 2011. This work was initially supported by Research Project "Rousset 2003–2008," financed by the Communauté du Pays d'Aix, Conseil Général des Bouches du Rhône, and Conseil Régional Provence Alpes Côte d'Azur.

A. Casali is with the Laboratoire d'Informatique Fondamentale de Marseille, Aix Marseille University, IUT (Technical Institute) of Aix-en-Provence, Aix-en-Provence 13625, France (e-mail: alain.casali@lif.univ-mrs.fr).

C. Ernst is with the Ecole des Mines de Saint-Etienne, Gardanne 13541, France (e-mail: ernst@emse.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSM.2011.2171375

concentrate on a particular application area, semiconductor wafer manufacturing. Yield is here defined as the ratio of nondefective chips in a finished wafer to the number of input products. However, we do not directly focus on yield enhancement, but on front-end issues, as shown in the next paragraph.

B. Enabling Quality in Semiconductor Fabs

In semiconductor manufacturing facilities, the volume and the complexity of the collected data are generally much more consequent than in other manufacturing fields due to the very nature of the domain. Fabrication processes include several hundred steps with regard to the produced "chip." Each of these steps uses various chemico-physical recipes, divided into four main phase units (photolithography, etch, implant, and chemical mechanical polishing).

Two approaches are used to improve the yield: real-time and *post hoc*. The first approach monitors on-line measurements of specific process steps, and undertakes corrective action to ensure that the parameter being measured remains within the 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), an extension of SPC, considers both aspects by highlighting correlations between production parameters in order to rectify possible shifts of the associated process(es).

This can be done for specific equipment and process steps in real-time: fault detection and classification (FDC) tools, and run to run (R2R) feedback and feedforward regulation loops are most representative APC techniques. FDC detects monitored key parameters which tend to drift. After identifying an abnormal status of a tool or a process running on it, the goal is to classify the detected failure. Associated data are finally checked by conventional SPC tools such as univariate or multivariate statistical methods, or by knowledge based procedures. R2R is an increasingly used process control method where process recipes are modified during the fabrication chain to diminish process drifts. The recipe contains all the equipment parameters required for a given process. A R2R control loop is able to center a process on a given value, by acting on defined parameters to reduce the process variability.

Correlations can also be discovered *post hoc*, i.e., after the whole fabrication process has been completed. This is the framework of our paper.

Both approaches try to identify the parameters causing particular yield excursion. By automatically deriving correlations between variability in process parameters including yield, model-based analysis can then reduce the time required to determine the yield loss causes. Let us emphasize that this second nontrivial problematic is excluded from the scope of this paper.

However, in manufacturing plants, conventional methods are inaccurate to improve yield, because they fail to extract underlying features from complex data [6], [7]. These methods include SPC and derived techniques such as FDC, design of experiments, or spatial mapping analysis. Component specification changes, mean process shift and variance reduction are well-known SPC techniques, while control charts aim at monitoring processes in order to detect abnormal drifts but cannot point out which parameters impact them.

When studying data mining techniques in semiconductor fabrication, the most widely used method is classification, even if it generally focuses on very specific process stages, such as cleaning [8], or photolithography [9]. The aim of classification is to build a classifier by induction from a set of pre-classified instances (the sensor measures). The classifier is then used to categorize "unlabeled" instances. Decision tree induction is the most representative approach in the field [10].

Clustering methods correspond to a particular classification of values into clusters. Among the relevant hierarchical algorithms that search to minimize a formal objective function, the most widely used is K-means clustering [11]. K-means remains also the simplest and most commonly used nonhierarchical algorithm employing a squared error criterion [12].

Prediction systems search to perform automatic discovery of significant parameters having an impact on the yield. Genetic programming [13] or neural networks [8] are therefore employed. Input data are first grouped into categorical classes. Field engineers can then build the relationship between the low yield lots and the in-line measurements at specific stages and, by the way, use these measurements to predict the future line yield.

Finally, only association rules are not often used to try to enhance yield. In [14], the authors used a modified *a priori* algorithm in LCD panel manufacturing to locate machines with low yield after completion of processes, and thus to improve the yield rate. In [15], correlations are also analyzed between combinations of used tools and defective products. Other relevant approaches do not directly deal with the semiconductor area [16].

C. Our Approach

We present hereafter a whole KDD model based on specific association rules. Within this framework, and in collaboration with STMicroelectronics (STM) and ATMEL (ATM), this paper is focused on the detection of the main control parameters impacting the yield. The goal is not to directly enhance the yield, but to propose indicators to which special attention should be paid in further production cycles through, for example, the construction of yield enhancement models.

Our *post hoc* analysis is based on comma-separated values (CSV) files of real valued measurements associated with production lots. These data have themselves been extracted in a previous step from very large manufacturer databases, covering the four fabrication units mentioned at the beginning of Section I-B. The main characteristic of the CSV 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: a particular column of the file, the yield. To detect these correlations, we introduce the concept of decision correlation rules, a restriction of correlation rules containing a value of one target column. In order to compute these rules:

1) We use the lectic order [17] to browse the powerset lattice (the search space).

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

2) We propose the concept of contingency vector: a new approach to contingency tables.
3) We show how to build the contingency vector of a pattern with a cardinality i with the contingency vector of one of its subsets with a cardinality $i-1$ (which is impossible with contingency tables).
4) We take advantage of the lectic order, the contingency vectors and the recursing mechanisms of construction to propose the LHS-CHI2 algorithm.

This paper is organized as follows. In Section II, the bases of association and correlation rules, and of the lectic order are recalled. Section III describes the concepts used for mining decisional correlation rules and our algorithm. In Section IV, we expose the other functions of the software—called *MineCor*—developed for mining decisional correlation rules. Experiments are detailed in Section V. As a conclusion, we summarize our contributions and outline some research perspectives.

II. RELATED WORK

In this section, we recall the definitions of association rules, correlation rules [18], and lectic order [17]. Then, we introduce the LS algorithm [19]. It allows the browsing of the search space according to the lectic order.

A. Statement of the Problem

An association rule is an approximate implication $X \rightarrow Y$ between two sets of items X and Y. Two measures are used to extract such rules: 1) support: the proportion of transactions (rows) containing X and Y, and 2) confidence: the ratio between the support of Y and the support of X (the degree of truth of the implication).

Example 1: The relation example r of Table I illustrates the introduced concepts. The BC pattern has a support equal to 4, and the rule $B \rightarrow C$ has a confidence equal to 2/3. This means that two thirds of the transactions including pattern B also contain pattern C.

Agrawal *et al.* [20] introduced levelwise algorithms for the computation of association rules in reasonable response times. Because the underlying semantics of an association rule are fairly poor, Wu *et al.* [21] introduced literal sets and proposed the computation of positive and/or negative association rules such as $\neg X \rightarrow Y$.

A literal is a pattern $X\overline{Y}$ in which X is also called the positive part and $\overline{Y}$ the negative part. To compute such rules, the authors still use the support-confidence platform by redefining the support of a literal: the number of transactions of the binary relation including X and containing no 1-item (item of cardinality 1) of Y.

Example 2: With the relation example r, the literalset $B\overline{C}$ has a support equal to 2. As a consequence, the association rule $B \rightarrow \overline{C}$ has a confidence equal to 1/3. This means that one third of the transactions containing pattern B does not include pattern C.

Brin *et al.* [18] proposed the extraction of correlation rules. The platform is no longer based on the support nor the confidence of the rules, but on the chi-squared statistical measure, written χ^2. The use of χ^2 is well-suited for several reasons: 1) it is a more significant measure in a statistical way than an association rule; 2) the measure takes into account not only the presence but also the absence of the items; and 3) the measure is nondirectional, and can thus highlight more complex existing links than a "*simple*" implication.

The crucial problem, when computing correlation rules, is the memory usage required by levelwise algorithms. For a pattern X, the computation of the χ^2 function is based on a table including $2^{|X|}$ cells. Thus, at level i, C_n^i candidates (where n is the number of values of r) have to be generated and stored, in the worst case scenario, as well as the associated contingency tables. With cells encoded over 2 Bytes, corresponding storage space requires 2.5 GB of memory at the third level, and 1.3 TB at the fourth level. This is why Brin *et al.* [18] computed only correlations between two values of a binary relation. Using an end-user threshold *MinCor*, Grahne *et al.* [22] showed that the "$\chi^2(X) \geq MinCor$" constraint is monotone. Consequently, the resulting set of rules is a convex space [23], which can be represented by its minimal border [24], noted L. In this paper, the author proposed a levelwise algorithm to compute L, and used an approximation to compute the χ^2 value of any pattern belonging to that convex space.

B. Correlation Rules

Let r be a binary relation (a transaction database) over a set of items $\mathcal{R} = \mathcal{I} \cup T$. In our approach, $\mathcal{I}$ represents the values (the items) of the binary relation used as analysis criteria, and $\mathcal{T}$ is a target attribute. For a given transaction, the target attribute does not necessarily have a value. The computation of the value for the χ^2 function for an item $X \subseteq \mathcal{R}$ is based on its contingency table. In order to simplify the notation, we first introduce the lattice of the literalsets associated with a pattern $X \subseteq \mathcal{R}$. This set contains all the literalsets that can be built up given X, and with a cardinality $|X|$.

Definition 1 (Literalset Lattice): Let $X \subseteq \mathcal{R}$ be a pattern. We denote by $\mathbb{P}(X)$ the literalset lattice associated with X: $\mathbb{P}(X) = \{Y\overline{Z}$ such that $X = Y \cup Z$ and $Y \cap Z = \emptyset\} = \{Y\overline{Z}$ such that $Y \subseteq X$ and $Z = X \backslash Y\}$.

Example 3: The literalset lattice associated with $X = \{A, B, C\}$ contains the following elements: $\{ABC, AB\overline{C}, AC\overline{B}, BC\overline{A}, A\overline{BC}, B\overline{AC}, C\overline{AB}, \overline{ABC}\}$.

Definition 2 (Contingency Table): For a given pattern X, its contingency table, noted $CT(X)$, contains exactly $2^{|X|}$ cells. Each cell stores the support of a literalset $Y\overline{Z}$ belonging to the literalset lattice associated with X.

TABLE II
CONTINGENCY TABLE OF PATTERN BC

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

Example 4: With the relation example r given in Table I, Table II shows the contingency table of pattern BC.

For each cell $Y\overline{Z}$ of $CT(X)$, we compute its expectation value: the theoretical frequency in case of independence of the 1-items included in $Y\overline{Z}$ [see (1)]

$$E(Y\overline{Z}) = |r| * \prod_{y \in Y} \frac{Supp(y)}{|r|} * \prod_{z \in Z} \frac{Supp(\overline{z})}{|r|}. \quad (1)$$

Formula (2) finally computes the value of the χ^2 function for a pattern X

$$\chi^2(X) = \sum_{Y\overline{Z} \in \mathbb{P}(X)} \frac{(Supp(Y\overline{Z}) - E(Y\overline{Z}))^2}{E(Y\overline{Z})}. \quad (2)$$

Brin *et al.* [18] showed that there is a single degree of freedom between the items. A table giving the centile values with regard to the χ^2 value for X can be used in order to obtain the correlation rate for X [25].

Example 5: Continuing our example, $\chi^2(BC) \simeq 0.28$, which corresponds to a correlation rate of about 45%.

Unlike association rules, a correlation rule is not represented by an implication but by the patterns for which the value of the χ^2 function is larger than a threshold.

Definition 3 (Correlation Rule): Let $MinCor$ be a threshold (≥ 0), and $X \subseteq \mathcal{R}$ a pattern. If the value for the χ^2 function for X is larger than or equal to $MinCor$, then this pattern represents a valid correlation rule.

Many authors have proposed additional constraints to evaluate whether a correlation rule is semantically valid [26]. Generally, the Cochran criteria are used: 1) all literalsets of a contingency table must have an expectation value not equal to zero (which never happens in our context), and 2) 80% of them must have a support larger than 5% of the whole population. This last criterion has been generalized by Brin *et al.* [18] as follows: $MinPerc$ of the literalsets of a contingency table must have a support larger than $MinSup$, where $MinPerc$ and $MinSup$ are also thresholds.

Example 6: Let $MinCor = 0.25$, then the correlation rule materialized by the BC pattern is valid ($\chi^2(BC) \simeq 0.28$). However, the correlation rule represented by the Bt_1 pattern is not valid ($\chi^2(Bt_1) \simeq 0.1$).

C. Lectic Order

The lectic order, noted $<_{lec}$, enumerates all the subsets of an itemset $\mathcal{I}$. This order allows the closed lattice of a binary relation to be computed [17], or to serve as a basis for the computation of the partition cube [19]: a lossless reduction of the data cube.

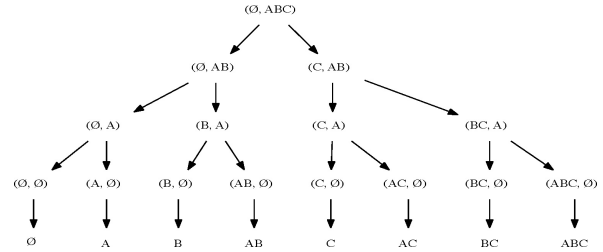

Fig. 1. Execution tree of LS for $\mathcal{I} = \{A, B, C\}$.

Definition 4 (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 $X <_{lec} Y \Leftrightarrow max_{\preceq}(X \backslash (X \cap Y)) \preceq max_{\preceq}(Y \backslash (X \cap Y))$.

Example 7: Let us consider the set $\mathcal{I} = \{A, B, C\}$, totally ordered according to the lexicographic order. The enumeration of the subsets of $\mathcal{I}$, according to the lectic order, produces the following result: $\emptyset <_{lec} A <_{lec} B <_{lec} AB <_{lec} C <_{lec} AC <_{lec} BC <_{lec} ABC$.

In order to enumerate all the subsets of $\mathcal{I}$ according to the lectic order, the lectic subset algorithm, noted LS [19], [27], is used. It is a simplified version of Algorithm 2 (limited to lines 1–7, 12). The associated execution tree is a balanced tree, based on a double recursive call. Being given a node of the tree (representing a pattern $X \subseteq \mathcal{I}$), the left subtree generates subpatterns of X not containing $max_{\preceq}(X)$, whereas the right subtree leads to subpatterns of X containing $max_{\preceq}(X)$.

Example 8: Fig. 1 shows the execution tree of the LS Algorithm for $\mathcal{I} = \{A, B, C\}$.

Proposition 1 expresses the fact that the lectic order is compatible with the antimonotone constraints. Consequently, we can modify the LS algorithm to take into account a conjunction of antimonotone constraints.

Proposition 1: Let be $X,\ Y \subseteq \mathcal{I}$ two itemsets. If $X \subset Y$, then $X <_{lec} Y$ [17].

III. LHS-CHI2 ALGORITHM

In this section, we introduce the contingency vectors: another representation of the contingency tables. We show that, for a given pattern $X \cup A$ ($X \subseteq \mathcal{R}$, $A \in \mathcal{R} \backslash X$), the computation of its contingency vector is possible using the contingency vector of X and the list of the row identifiers of the relation containing A. Then, we present the concept of the decision correlation rule: a restriction of correlation rules, in such a way that only the rules containing a value of the target attribute are kept. Finally, in order to compute these rules, we describe the LHS-CHI2 Algorithm.

A. Contingency Vectors

A literal $Y\overline{Z}$, belonging to the literalset lattice associated with a pattern X, is represented in a computer with vectors of $|X|$ bits. For a 1-item $x \in X$, the value of the bit vector has a value of 1 if $x \in Y$ (the 1-item belongs to the positive part of the literal), and 0 otherwise. Thus, comparing two literals $Y_1\overline{Z_1}$ and $Y_2\overline{Z_2}$ belonging to the literalset lattice associated with pattern X, consists in comparing each integer corresponding

to the binary value of the associated bit vector. The comparison is equivalent to extending the definition of the lectic order to the literalset one.

This order allows the total ordering of the whole literalset lattice associated with pattern X.

Definition 5 (Lectic Order for a Literalset): Let $X \subseteq \mathcal{R}$ be a pattern, $Y_1\overline{Z_1}$ and $Y_2\overline{Z_2}$ two elements of the literalset lattice associated with the X pattern. The definition of the lectic order is extended over the literalsets as follows: $Y_1\overline{Z_1} <_{lec} Y_2\overline{Z_2}$ if and only if $Y_1 <_{lec} Y_2$.

Example 9: The literalset lattice associated with the pattern $X = \{A, B, C\}$ according to the lectic order is the following: $\overline{ABC} <_{lec} A\overline{BC} <_{lec} B\overline{AC} <_{lec} AB\overline{C} <_{lec} C\overline{AB} <_{lec} AC\overline{B} <_{lec} BC\overline{A} <_{lec} ABC$.

Definition 6 (Equivalence Class Associated with a Literal): Let $Y\overline{Z}$ be a literal. Let us denote by $[Y\overline{Z}]$ the equivalence class associated with the literal $Y\overline{Z}$. This class contains the set of transaction identifiers of the relation including Y and containing no value of Z (i.e., $[Y\overline{Z}] = \{i \in Tid(r)$ such that $Y \subseteq Tid(i)$ and $Z \cap Tid(i) = \emptyset\}$).

Example 10: With our relation example (see Table I), we have $[B\overline{C}] = \{5, 6\}$.

Proposition 2: Let $X \subseteq \mathcal{R}$ be a pattern. The union of the equivalence classes $[Y\overline{Z}]$ of the literalset lattice associated with X is a partition [28] of the identifiers of relation r. In other words

$$\bigcup_{Y\overline{Z} \in \mathbb{P}(X)} [Y\overline{Z}] = Tid(r).$$

Definition 7 (Contingency Vector): Let $X \subseteq \mathcal{R}$ be 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.

Proposition 2 ensures that each transaction identifier belongs only to one single equivalence class. Consequently, for a given pattern X, its CV is an exact representation of its contingency table. To derive the contingency table from a contingency vector, it is sufficient to compute the cardinality of each of its equivalence classes. If the literalsets, related to the equivalence classes of a CV, are ordered according to the lectic order, it is possible to know the literal relative to a position i of a contingency vector ($i \in [0; |X|-1]$). This is because the literal and the integer i have the same binary coding.

Example 11: With our sample relation (see Table I), the contingency vector associated with the BC pattern is the following: $CV(BC) = \{[\overline{BC}], [B\overline{C}], [C\overline{B}], [BC]\} = \{\{9, 10\}, \{5, 6\}, \{7, 8\}, \{1, 2, 3, 4\}\}$.

Theorem 1 is the main result of our paper. It shows how to compute the CV of the $X \cup A$ pattern given the CV of X and the set of identifiers of the relation containing pattern A.

Theorem 1: Let $X \subseteq \mathcal{R}$ be a pattern and $A \in \mathcal{R} \backslash X$ a 1-item. The contingency vector of $X \cup A$ can be computed given the contingency vectors of X and A as follows:

$$CV(X \cup A) = (CV(X) \cap [\overline{A}]) \cup (CV(X) \cap [A]). \quad (3)$$

Example 12: With the relation example (see Table I), 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 Theorem 1, the contingency vector of BC is the following: $CV(BC) = \{\{9, 10\}, \{5, 6\}, \{7, 8\}, \{1, 2, 3, 4\}\}$. Thus, we retrieve the result of Example 11.

Algorithm 1 CREATE_CV Algorithm

Input: $CV(X)$ contingency vector of X, $Tid(A)$
Output: contingency vector of $X \cup A$ sorted according to the lectic order

1: $CV(Z) := \{\emptyset\}$
2: **for all** Equivalence classes $[Y\overline{Z}] \in \mathbb{P}(X)$ according to the lectic order **do**
3: $\quad CV(Z) := CV(Z) \cup ([Y\overline{Z}] \cap (Tid(r) \backslash (Tid(A))) \cup ([Y\overline{Z}] \cap Tid(A))$
4: **end for**
5: **return** $CV(Z)$

Algorithm 1 is used, given the CV of a pattern X and the set of the transaction identifiers containing a 1-item A, to build the CV of the $X \cup A$ pattern sorted according to the lectic order over the literalset lattice $\mathbb{P}(X \cup A)$. Line 3 is an adaptation of Theorem 1 to our context.

The computation of a CV needs one database scan, and the following transition to the associated CT another one (overheads are ignored). This leads to a complexity of $2 * |r|$ or $\mathcal{O}(|r|)$, whatever the number of cells in the CT. A classical computation of a CT at level i also needs one database scan; but here, in the worst case, each of the CT cells is involved in one operation, which globally forces $2^i * |r|$ operations. Because 2^i is generally much smaller in comparison to $|r|$, the complexity is also of $\mathcal{O}(|r|)$. But when going into detail, the difference between the two methods is $2^{i-1} * |r|$ operations.

B. Decision Correlation Rules

Definition 8 (Decision Correlation Rules): Let $X \subseteq \mathcal{R}$ be a pattern, and $MinCor$ a given threshold. X represents a valid decision correlation rule if and only if:

1) X contains a value of the target attribute $\mathcal{T}$;
2) $\chi^2(X) \geq MinCor$.

Example 13: With our relation example (see Table I), if $MinCor = 0.25$, the decision correlation rule materialized by the BCt_1 pattern is a valid rule because:

1) $t_1 \in \mathcal{T}$ and $t_1 \in BCt_1$;
2) $\chi^2(BCt_1) \simeq 0.28$ ($\geq MinCor$).

The lectic hybrid subset-Chi2 algorithm, or LHS-CHI2, permits to extract the whole set of decision correlation rules for a relation r satisfying the threshold constraint $MinCor$ for the χ^2 function. This algorithm is an adaptation of the LS Algorithm to our context, by taking into account contingency vectors. Moreover, we added several monotone and antimonotone constraints in order to prune the search space [22].

1) A value of the target attribute must be present in the extracted pattern (monotone constraint).
2) As the χ^2 computation has no significance for a 1-item, we only examine patterns of cardinality larger than or equal to two (monotone constraint).

3) Since the χ^2 function is an increasing function, we impose a maximum cardinality, noted *MaxCard*, on the number of 1-items belonging to the patterns to examine (antimonotone constraint).
4) All literalsets of a *CT* must have an expectation value not equal to zero (antimonotone constraint).
5) Because the obtained rules must have a semantics on the relation, at least *MinPerc* of the cells of a *CT* must have a support larger than or equal to *MinSup*. This constraint is expressed in our algorithm by the *CtPerc* predicate, which parameters are the contingency vector, *MinPerc* and *MinSup* (antimonotone constraint).

Laporte *et al.* [19] modified the LS Algorithm in order to compute "iceberg" data cubes. The authors included an antimonotone constraint threshold, evaluated before the second recursive call of the LS Algorithm; they used a pruning step with the negative border [29] in order to only examine the most "*interesting*" cuboids (patterns in our context). In the same spirit, we modify LS in order to take into account the five constraints above, and to compute the χ^2. The result is an algorithm requiring, in the worst case, $|\mathcal{R}|+MaxCard+1$ *CVs* in memory. We need $|\mathcal{R}|$ *CVs* for the 1-items, the height of our tree is bounded by *MaxCard*, and we need an additional *CV* for the current node computation. This value has to be compared to the number of contingency tables to be computed at each level using a levelwise algorithm (see the end of Section II-A).

Proposition 1 justifies the inclusion of these constraints into our algorithm. However, we do not carry out pruning using the negative border. Instead, we use the positive border [29] relating to predicate *CtPerc*. The use of the positive border is justified on the basis of the experiments carried out by Flouvat *et al.* [30]. The authors showed that the positive border is of highly reduced cardinality in comparison with the negative one. As a consequence, the satisfiability tests of the antimonotone constraints are faster when the positive border is used. In our context, we make sure that the *Z* pattern, used as a parameter within the second recursive call of the algorithm, has all its direct subsets included in one of the elements of the positive border (line 8). Let us emphasize that this test is carried out in the *AprioriGen* function [20] during the generation of the candidates of level $i+1$ using the frequent i-itemsets. If pattern *Z* is a candidate, then we compute its contingency vector by making sure that the literalsets relating to the classes of equivalence are sorted according to the lectic order (line 9) by calling Algorithm 1. If the pattern satisfies the antimonotone constraints (line 10), we update the positive border (line 11), and carry out the second recursive call of the algorithm (line 12). The monotone constraints are evaluated on the leaves of the execution tree (line 1). By convention, we have $CV(\emptyset) = \{Tid(R),\ \emptyset\}$. The positive border is initialized with $\{\emptyset\}$. The pseudo code of LHS-CHI2 is provided in Algorithm 2. The first recursive call to LHS-CHI2 is carried out with $X = \emptyset$ and $Y = \mathcal{R}$.

Example 14: The results of LHS-CHI2 with $MinSup = 0.2$, $MinPerc = 0.25$, and $MinCor = 0.25$ for our relation example (see Table I) are shown in Table III.

Algorithm 2 LHS-CHI2 Algorithm

Input: *X* and *Y* two patterns
Output: $\{itemset\ Z \subseteq X$ such that $\chi^2(Z) \geq MinCor\}$

```
1: if Y = ∅ and |X| ≥ 2 and ∃c ∈ C : c ∈ X and χ²(X) ≥ MinCor then
2:    Output X, χ²(X)
3: end if
4: A := max(Y)
5: Y := Y\{A}
6: LHS-CHI2(X, Y)
7: Z := X ∪ {A}
8: if ∀z ∈ Z, ∃W ∈ BD⁺ : {Z\z} ⊆ W then
9:    VC(Z) := CREATE_CV(CV(X), Tid(A))
10:   if |Z| ≤ MaxCard and
      CtPerc(CV(Z), MinPerc, MinSup) then
11:      BD⁺ := max⊆(BD⁺ ∪ Z)
12:      LHS-CHI2(Z, Y)
13:   end if
14: end if
```

TABLE III
RESULTS OF THE LHS-CHI2 ALGORITHM OVER TABLE I

Decision Correlation Rule	χ^2 Value
At_1	0.48
BCt_1	0.28
BFt_1	0.28

IV. MINECOR SOFTWARE

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 CSV files given by our manufacturer partners (see the end of Section I), we have first performed column elimination and discretization stages [1], [31]. These steps, known as data cleaning or cleansing in the literature, are summarized in Sections IV-A and IV-B. The output of the two steps is placed into a feature database, which serves as a source for the data mining phase. Finally, after the mining step, the results are interpreted, what is resumed in Section IV-C.

A. Preprocessing Stage

The first step of data cleaning is the preprocessing stage. Data has to be prepared for two reasons: 1) if each value of each column is considered as a single item, the search space explodes combinatorially, and results cannot be provided in a reasonable amount of time, and 2) we cannot expect this task to be performed by an expert, because manual cleaning of data is laborious and subject to errors.

Preprocessing consists in the reduction of the data structure [32] by eliminating columns (and rows) of low significance. Such situations can result, for example, from the dysfunction of one or more sensors, or from the occurrence of a maintenance step. As a consequence, corresponding columns contain many null or default values, and must be deleted from the source file. 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. Another classical technique is the elimination of columns having small standard deviation. Since all values are almost the same, we consider that they do not have a significant impact on the result; but their inclusion pollutes the search space and reduces the response time of *MineCor*. Attention is finally paid to missing or inconsistent values, such as "outliers" and noisy columns. Elimination is performed through thresholds specified by the end-user.

B. Discretization Stage

Discrete values deal with value intervals, 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. In this section, we only summarize these methods. Discretization can be performed [33]: 1) in a supervised or unsupervised manner, depending on whether class information is at one's disposal; 2) in a dynamic or static way: with a static discretization approach, discretization is done before the classification task; and 3) using splitting or merging techniques. In the latter case, using a bottom-up approach while examining the search space.

We represent continuous real valued columns by associating each of their values with an interval code. The bins are created either using equal-width or equal-frequency discretization, which are nonsupervised, static, and splitting methods. In both approaches, arity k is the number of intervals to use. And the different values associated with each set S are managed in the same way through initial normalization.

1) *Equal Width Discretization (EWD):* Let $\mathcal{S}$ be the set of values to be discretized, and respectively $Min_{\mathcal{S}}$ and $Max_{\mathcal{S}}$ the smallest and the largest value of $\mathcal{S}$. Each interval has a length of $l = \frac{Max_{\mathcal{S}} - Min_{\mathcal{S}}}{k}$. The computed classes are $c_1 : [Min_{\mathcal{S}}, Min_{\mathcal{S}} + l[$, $c_2 : [Min_{\mathcal{S}} + l, Min_{\mathcal{S}} + 2l[$, ...

2) *Equal Frequency Discretization (EFD):* The goal is to obtain classes having, if possible, the same number of continuous values. The Jenks' natural breaks method minimizes the in-class difference and maximizes the between-class difference [34]. This can be measured by the goodness of variance fit (GVF)

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

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 $\mathcal{S}$. Jenks' method is the best from a statistical point of view because it creates homogeneous groups. Its main drawback is the high computational complexity of the class generation, which is C_{d-1}^{k-1}, where d represents the number of distinct values in the set $\mathcal{S}$. Thus, we use instead the Fisher's exact optimization method [35] proposed for grouping n elements into k mutually exclusive and exhaustive subsets having maximum homogeneity. The partition is guaranteed to be optimal, but not unique, which is not important while the obtained time gain is. This is why the EFD method is also referred to, in the next sections of this paper, as the Fisher–Jenks' method.

				Target Attribute		Chi2
Column1 (B)		Column2 (C)		t1		0.28
[1.4000, 2.2000]	0.600	[2.8000, 4.6000]	0.600	[2.7000, 7.4500]	0.500	
Column1 (B)		Column3 (F)		t1		0.28
[1.4000, 2.2000]	0.600	[9.8000, 15.4000]	0.400	[2.7000, 7.4500]	0.500	

Fig. 2. Output produced by *MineCor*.

Example 15: Let $\mathcal{S} = \{1.8, 1.9, 2.1, 2.2, 1.3, 2.0, 0.5, 0.6, 0.5, \text{NULL}\}$ be the set to discretize. If we specify two output classes, the proposed methods produce the following results.

1) EWD: since $\frac{Max_{\mathcal{S}} - Min_{\mathcal{S}}}{2} = 1.35$, this method computes the classes $[0.6, 1.3]$, $[1.8, 2.2]$. As a consequence, the set $\mathcal{S}$ is encoded by the vector {B, B, B, B, A, B, A, A, A, - } in the output of the discretization step ("-" symbolizes the NULL value).
2) EFD: the Fisher–Jenks' method produces ten class generation possibilities. The one which maximizes the squared sum is $[0.5, 0.6]$, $[1.3, 2.2]$. The following vector is produced to represent the set $\mathcal{S}$: {B, B, B, B, B, B, A, A, A, - }. Let us underline that we retrieve here partial results presented in Table I.

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 16: Fig. 2 provides an example of output produced by *MineCor*, limited to some 3-patterns. Given a row, the last column is the computed χ^2 value for the associated decision correlation rule.

As mentioned in Section IV-B, and because EWD is the default method, the results shown are slightly different than those presented in Table III.

V. Experimental Analysis

Some representative results of the LHS-Chi2 algorithm are presented below. The comparison is made with a standard levelwise (a complete *a priori*) algorithm, hereafter called LevelWise, based on the same monotone and antimonotone constraints as those used in LHS-Chi2 (see Section III). The main difference is that the LevelWise method does not use contingency vectors but uses standard computation of contingency tables.

As emphasized in Section I-C, the experiments were done on different CSV files of real value measures supplied by STM and ATM. These files have one or more target columns, resulting from the concatenation of several measurement files. The characteristics of the datasets used for experiments can be found in Table IV. All experiments were conducted on a HP Workstation (1.8 GHz processor with a 4 GB RAM).

TABLE IV
DATASET EXAMPLES

Name	Number of Columns	Number of Rows
STM File	1281	297
ATM File	749	213

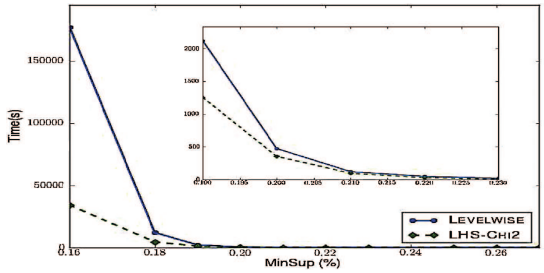

Fig. 3. Execution time with *MinPerc* = 0.34, *MinCor* = 1.6 (STM file: target1).

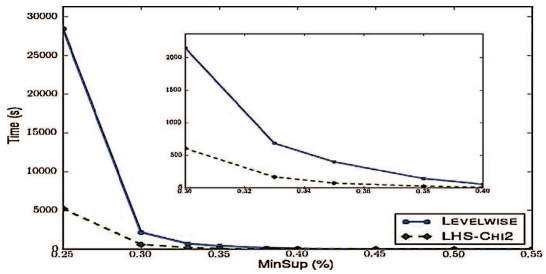

Fig. 4. Execution time with *MinPerc* = 0.24, *MinCor* = 2.8 (ATM file: target2).

Experimental results are presented on Figs. 3 to 8(c). The EWD discretization method is used in all the experiments carried out in Sections V-A to V-C.

A. Execution Times for LHS-CHI2 and LEVELWISE Algorithms

Figs. 3 and 4 show the evolution of the execution times for both methods for the two files when *MinSup* varies and *MinPerc* and *MinCor* are fixed. As the graphs point it out, the response times of our method are between 30% and 70% better than LEVELWISE, even if they remain high when using small thresholds. In each case, an increasing windowing of the results is provided for subsequent subintervals of *MinSup*.

B. Impact of the MinPerc Parameter

Fig. 5 shows the execution times for the STM file (using the same configuration as the experiment in Fig. 3) when *MinSup* and *MinCor* are constant, and when *MinPerc* varies. The staircase curve thus explains. A CT associated with a i-pattern containing 2^i cells, specifying that *MinPerc* of its cells must have the support means that $\lceil 2^i * MinPerc \rceil$ cells must have it. So, for a 3-pattern, to define a value for *MinPerc* varying between 0% and 12.49% means specifying that one single cell of the CT has to have the support, and so on. The scale is logarithmic, because response times for small values of *MinPerc* are very high (more than 13 h for LHS-CHI2, and about 69 h for LEVELWISE with *MinPerc* = 0.12).

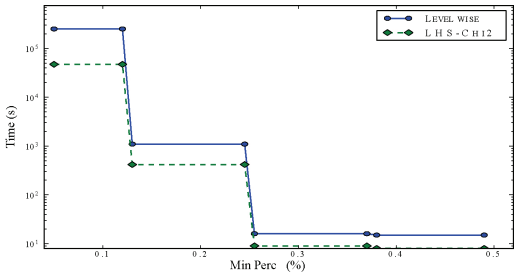

Fig. 5. Execution time with *MinSup* = 0.24, *MinCor* = 6.9 (STM file: target1).

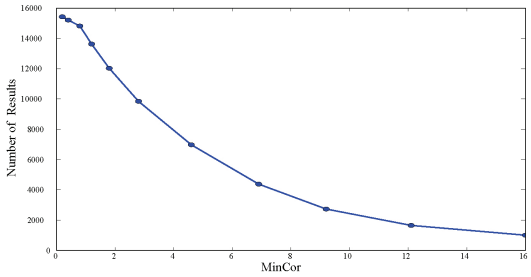

Fig. 6. Results with *MinSup* = 0.38, *MinPerc* = 0.24 (ATM file: target3).

C. Impact of the MinCor Parameter

Fig. 6 shows the number of extracted rules (identical in both methods) after mining when *MinPerc* and *MinSup* are fixed with suitable values and when *MinCor* varies. In that particular case, execution times are identical whatever the *MinCor* value, but are of the order of 2 min with LHS-CHI2, and about 17 min for LEVELWISE. This means that the *MinCor* threshold only has a small effect on performance.

D. Impact of the Discretization Stage

Figs. 7(a) and 8(a) show the number of items kept after the preprocessing and discretization stages. This number only depends on the *MinSup* threshold, while the number of bins is constant [4 in Fig. 7(a), and 6 in Fig. 8(a)]. In each example, all items with a support greater than *MinSup* are kept.

As illustrated in Figs. 7(a) and 8(a), the smaller the threshold *MinSup*, the larger the number of items kept for the mining stage, whatever the discretization method. Figs. 7(b) and 8(b) show the number of rules 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 [see Figs. 7(c) and 8(c)]. These results come from the perspective that *MineCor* tries to provide rules of "best" quality: 1) low in number; 2) significant; and 3) computed quickly.

Finally, let us emphasize that the experimental sets used in Fig. 7 produce decision correlation rules with a cardinality of 4. This is the kind of information that is of interest for semiconductor manufacturers, as well as different possible crossings using other techniques (see Section I-B) between rules of cardinality 3 and 4.

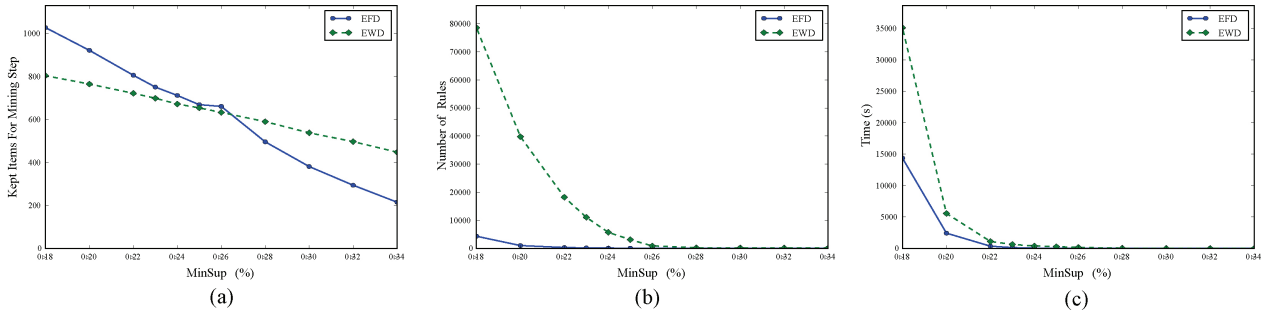

Fig. 7. Results with four intervals, *CtPerc* = 0.34, *MinCorr* = 1.6 (STM file: target1). (a) Number of items kept after discretization/preprocessing stages. (b) Number of generated decision correlation rules. (c) Execution time.

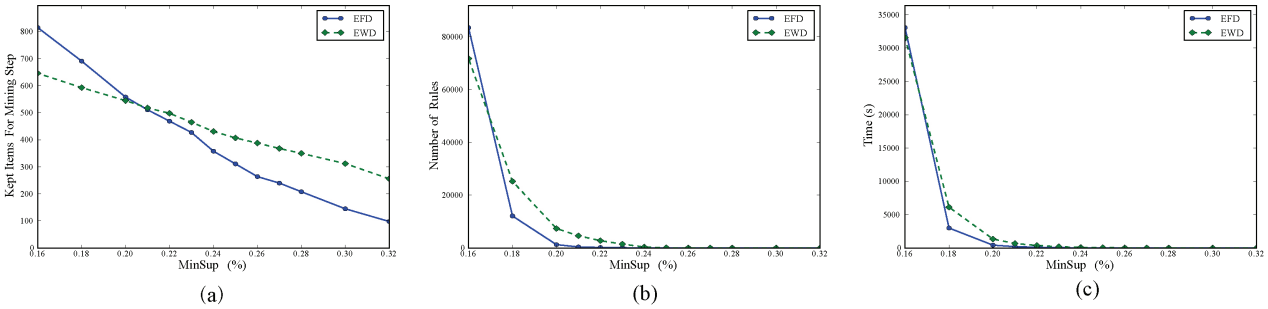

Fig. 8. Results with six intervals, *CtPerc* = 0.3, *MinCorr* = 2.8 (ATM file: target3). (a) Number of items kept after discretization/preprocessing stages. (b) Number of generated decision correlation rules. (c) Execution time.

VI. Conclusion and Future Work

In this paper, we showed the different facets of the *MineCor* software. CSV parameter measurement files given by semiconductor manufacturers (STMicroelectronics and ATMEL) are used as input, and produce as output values of parameters with most influence on the yield. To achieve this objective, we built a complete knowledge discovery in databases model, based on:

1) decision correlation rules, i.e., a restriction of correlation rules containing a target attribute value;
2) contingency vectors, i.e., an alternative representation of contingency tables, which are more concise and offer better performance related properties. We finally proposed an algorithm based on the lectic order to go through the powerset lattice.

The LHS-Chi2 algorithm is the heart of our model. It uses the inference property of the contingency vector of a pattern given the contingency vector of one of its direct subsets. The experiments show that the proposed method computes rules faster than those offered by levelwise algorithms. Moreover, we implemented two methods at the discretization stage: 1) equal width discretization, and 2) equal frequency discretization based on the Fisher–Jenks' method. Experiments show that, in most cases, the latter method produces decision correlation rules faster and of better quality. Furthermore, the software enables us to find new correlations between the parameters of the files that have been studied. As an example, approximately 25% of the correlation rules determined by the first experiment were unknown to STM, and the quasi-totality of the results obtained have been experimentally validated.

Finally, let us emphasize that the presented *post hoc* method could also be applied in real-time, i.e., associated with specific process steps, from the moment on the relevant configuration parameters are set up in an optimal way. Moreover, our KDD model could be used in other domains than wafer manufacturing.

Some new issues to our work are: 1) to optimize memory management in order to increase the performance of LHS-Chi2; 2) to compare our approach with other mining methods; 3) 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 more significant results; and 4) to broaden the correlation rule extraction problem on items to those on literalsets.

References

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Waltham, MA: Morgan Kaufmann, 2000.
[2] G. Piatetsky-Shapiro and C. J. Matheus, "The interingness of deviations," in *Proc. KDD Workshop*, 1994, pp. 25–36.
[3] M. Klemettinen, H. Mannila, and H. Toivonen, "A data mining methodology and its application to semi-automatic knowledge acquisition," in *Proc. DEXA Workshop*, 1997, pp. 670–677.
[4] A. Choudhary, J. Harding, and M. Tiwari, "Data mining in manufacturing: A review based on the kind of knowledge," *J. Intell. Manuf.*, vol. 20, no. 5, pp. 501–521, 2009.
[5] J. Pan and D. Tai, "Implementing virtual metrology for in-line quality control in semiconductor manufacturing," *Int. J. Syst. Sci.*, vol. 40, no. 5, pp. 461–470, 2009.
[6] L. Huisman, *Data Mining and Diagnosing IC Fails*. Berlin, Germany: Springer, 2005.
[7] M. A. Karim, S. K. Halgamuge, A. J. R. Smith, and A. L. Hsu, "Manufacturing yield improvement by clustering," in *Proc. ICONIP*, vol. 3. 2006, pp. 526–534.

[8] D. Braha and A. Shmilovici, "Data mining for improving a cleaning process in the semiconductor industry," *IEEE Trans. Semicond. Manuf.*, vol. 15, no. 1, pp. 91–101, Feb. 2002.
[9] D. Braha and A. Shmilovici, "On the use of decision tree induction for discovery of interactions in a photolithographic process," *IEEE Trans. Semicond. Manuf.*, vol. 16, no. 4, pp. 644–652, Nov. 2003.
[10] D.-H. Baek, I.-J. Jeong, and C. H. Han, "Application of data mining for improving yield in wafer fabrication system," in *Proc. ICCSA*, vol. 4. 2005, pp. 222–231.
[11] D. MacKay, *An Example Inference Task: Clustering*. Cambridge, U.K.: Cambridge University Press, 2003, ch. 20, pp. 284–292.
[12] C.-F. Chien, W.-C. Wang, and J.-C. Cheng, "Data mining for yield enhancement in semiconductor manufacturing and an empirical study," *Expert Syst. Appl.*, vol. 33, no. 1, pp. 192–198, 2007.
[13] L. Rokach and O. Maimon, "Data mining for improving the quality of manufacturing: A feature set decomposition approach," *J. Intell. Manuf.*, vol. 7, no. 3, pp. 285–299, 2006.
[14] C. Huang and R. Chen, "Application of new *a priori* algorithm MDNC to TFT-LCD array manufacturing yield improvement," *Int. J. Comput. Applicat. Technol.*, vol. 28, nos. 2–3, pp. 161–168, 2007.
[15] W. Chen, S. Tseng, and C. Wang, "A novel manufacturing defect detection method using association rule mining techniques," *Expert Syst. Applicat.*, vol. 29, no. 4, pp. 807–815, Nov. 2005.
[16] H. Sadoyan, A. Zakarian, and P. Mohanty, "Data mining algorithm for manufacturing process control," *Int. J. Adv. Manuf. Technol.*, vol. 28, nos. 3–4, pp. 342–350, 2006.
[17] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Berlin, Germany: Springer, 1999.
[18] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Proc. SIGMOD Conf.*, 1997, pp. 265–276.
[19] M. Laporte, N. Novelli, R. Cicchetti, and L. Lakhal, "Computing full and iceberg datacubes using partitions," in *Proc. ISMIS*, 2002, pp. 244–254.
[20] 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*. Cambridge, MA: AAAI/MIT Press, 1996, pp. 307–328.
[21] 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.
[22] G. Grahne, L. V. S. Lakshmanan, and X. Wang, "Efficient mining of constrained correlated sets," in *Proc. ICDE*, 2000, pp. 512–521.
[23] M. Vel, *Theory of Convex Structures*. Amsterdam, The Netherlands: North-Holland, 1993.
[24] H. Hirsh, "Generalizing version spaces," *Mach. Learning*, vol. 17, no. 1, pp. 5–46, 1994.
[25] M. Spiegel and L. Stephens, *Outline of Statistics*. New York: McGraw-Hill, 1998.
[26] D. Moore, "Measures of lack of fit from tests of chi-squared type," *J. Statist. Planning Inference*, vol. 10, no. 2, pp. 151–166, 1984.
[27] C. Ernst and A. Casali, "Extraction de règles de corrélation décisionnelles," in *Proc. EGC*, 2009, pp. 187–192.
[28] D. Laurent and N. Spyratos, "Partition semantics for incomplete information in relational databases," in *Proc. SIGMOD Conf.*, 1988, pp. 66–73.
[29] H. Mannila and H. Toivonen, "Levelwise search and borders of theories in knowledge discovery," *Data Min. Knowl. Discov.*, vol. 1, no. 3, pp. 241–258, 1997.
[30] F. Flouvat, F. D. Marchi, and J.-M. Petit, "A thorough experimental study of datasets for frequent itemsets," in *Proc. ICDM*, 2005, pp. 162–169.
[31] D. Pyle, *Data Preparation for Data Mining*. Waltham, MA: Morgan Kaufmann, 1999.
[32] 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*. Boston, MA/Dordrecht, The Netherlands/London, U.K.: K. A. Publishers, 2003, pp. 107–117.
[33] 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.
[34] G. Jenks, "The data model concept in statistical mapping," in *International Yearbook of Cartography*, vol. 7, E. Imhof, Ed. Gutersloh, 1967, pp. 186–190.
[35] W. Fisher, "On grouping for maximum homogeneity," *J. Am. Statist. Assoc.*, vol. 53, pp. 789–798, Dec. 1958.

Alain Casali received the Ph.D. degree in computer science from Aix Marseilles University, Aix-en-Provence/Marseille, France, in 2005.

He is currently an Assistant Professor with Aix Marseilles University—IUT (Technical Institute) of Aix-en-Provence, and is a member of the Laboratoire d'Informatique Fondamentale de Marseille. He is currently doing research on lattice algorithmics and multidimensional database mining.

Christian Ernst received the Ph.D. degree in computer science from the University of Nice, Nice, France, in 1987.

He is currently an Associate Professor with the Provence Microelectronics Center, Ecole des Mines de Saint-Etienne, Gardanne, France. After spending 6 years with two software engineering companies, he joined the Institut Superieur de MicroElectronique Appliquee Graduate Engineering School, Marseille, France, where he was the Director of Studies from 1996 to 2005. He has been involved in more than 20 industrial and academic research-oriented projects. His current research interests include algorithmics, data base systems, and data mining techniques.

Discovering Correlated Parameters in Semiconductor Manufacturing Processes: A Data Mining Approach

Alain Casali and Christian Ernst

***Abstract*—Data mining tools are nowadays becoming more and more popular in the semiconductor manufacturing industry, and especially in yield-oriented enhancement techniques. This is because conventional approaches fail to extract hidden relationships between numerous complex process control parameters. In order to highlight correlations between such parameters, we propose in this paper a complete knowledge discovery in databases (KDD) model. The mining heart of the model uses a new method derived from association rules programming, and is based on two concepts: decision correlation rules and contingency vectors. The first concept results from a cross fertilization between correlation and decision rules. It enables relevant links to be highlighted between sets of values of a relation and the values of sets of targets belonging to the same relation. Decision correlation rules are built on the twofold basis of the chi-squared measure and of the support of the extracted values. Due to the very nature of the problem, levelwise algorithms only allow extraction of results with long execution times and huge memory occupation. To offset these two problems, we propose an algorithm based both on the lectic order and contingency vectors, an alternate representation of contingency tables. This algorithm is the basis of our KDD model software, called *MineCor*. An overall presentation of its other functions, of some significant experimental results, and of associated performances are provided and discussed.**

***Index Terms*—Chi^2 correlation statistic, data mining, decision rule, semiconductor manufacturing.**

I. Introduction and Motivation

IN THIS SECTION, we first introduce why and how data mining techniques are useful to enhance semiconductor fabrication capabilities. Discussion is set on how to detect the main parameters which have an impact on yield loss rather than on how to improve final yield. Then we present our approach that determines the main correlated production parameters impacting the yield.

A. Data Mining Techniques in the Manufacturing Industry

Data mining [1] allows us to extract data in terms of models which may be rules, concepts, patterns, anomalies, or trends that are useful and intelligible for the end-user. Nowadays, databases or data warehouses of significant size implicitly contain a large amount of relevant information. Their extraction presents an interest in various domains such as marketing, design, medical research [2], telecommunication networks [3], dynamic restructuring of websites [3], manufacturing sciences [4], and so on.

Data mining models can be categorized into four types [1]: classification, clustering, prediction, and association rules. Such approaches have been widely carried out in manufacturing areas [4]. Data mining extracts knowledge to identify hidden patterns in the parameters that control manufacturing processes or to determine and to improve product quality. Unfortunately, there is no standard scalable model for manufacturing applications. The models used are a collection of "implementation specific" data mining algorithms. Associated applications can be roughly divided into six categories.

1) Customer relationships: the objective is to develop the relationship with the customers in order to maximize profits.
2) Engineering design: based on historical data, the goal is to optimize design specification by matching the temporal data of a new product with the knowledge base.
3) Manufacturing systems: in such environments, the need and importance of data is ever present for statistical process control (SPC) purposes; SPC consists in effective statistical methods for monitoring a process through the use of control charts, by enabling the use of objective criteria for distinguishing background variation from events of significance.
4) (Equipment) maintenance: since databases contain information to improve processes, they also contain the reasons for machine failures.
5) Fault detection and quality improvement: examining what happened in the past is used to better understand the process, and therefore to predict and to improve the future system's performance. Virtual metrology [5] is here one of the most novel tools.
6) Decision support systems: the goal is to determine links between control parameters and product quality, essentially in the form of (decision) rules.

We focus hereafter on the last two points, which deal with quality, and thus implicitly with product yield. Moreover, we

Manuscript received September 2, 2010; revised March 21, 2011; accepted July 31, 2011. This work was initially supported by Research Project "Rousset 2003–2008," financed by the Communauté du Pays d'Aix, Conseil Général des Bouches du Rhône, and Conseil Régional Provence Alpes Côte d'Azur.

A. Casali is with the Laboratoire d'Informatique Fondamentale de Marseille, Aix Marseille University, IUT (Technical Institute) of Aix-en-Provence, Aix-en-Provence 13625, France (e-mail: alain.casali@lif.univ-mrs.fr).

C. Ernst is with the Ecole des Mines de Saint-Etienne, Gardanne 13541, France (e-mail: ernst@emse.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSM.2011.2171375

concentrate on a particular application area, semiconductor wafer manufacturing. Yield is here defined as the ratio of nondefective chips in a finished wafer to the number of input products. However, we do not directly focus on yield enhancement, but on front-end issues, as shown in the next paragraph.

B. Enabling Quality in Semiconductor Fabs

In semiconductor manufacturing facilities, the volume and the complexity of the collected data are generally much more consequent than in other manufacturing fields due to the very nature of the domain. Fabrication processes include several hundred steps with regard to the produced "chip." Each of these steps uses various chemico-physical recipes, divided into four main phase units (photolithography, etch, implant, and chemical mechanical polishing).

Two approaches are used to improve the yield: real-time and *post hoc*. The first approach monitors on-line measurements of specific process steps, and undertakes corrective action to ensure that the parameter being measured remains within the 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), an extension of SPC, considers both aspects by highlighting correlations between production parameters in order to rectify possible shifts of the associated process(es).

This can be done for specific equipment and process steps in real-time: fault detection and classification (FDC) tools, and run to run (R2R) feedback and feedforward regulation loops are most representative APC techniques. FDC detects monitored key parameters which tend to drift. After identifying an abnormal status of a tool or a process running on it, the goal is to classify the detected failure. Associated data are finally checked by conventional SPC tools such as univariate or multivariate statistical methods, or by knowledge based procedures. R2R is an increasingly used process control method where process recipes are modified during the fabrication chain to diminish process drifts. The recipe contains all the equipment parameters required for a given process. A R2R control loop is able to center a process on a given value, by acting on defined parameters to reduce the process variability.

Correlations can also be discovered *post hoc*, i.e., after the whole fabrication process has been completed. This is the framework of our paper.

Both approaches try to identify the parameters causing particular yield excursion. By automatically deriving correlations between variability in process parameters including yield, model-based analysis can then reduce the time required to determine the yield loss causes. Let us emphasize that this second nontrivial problematic is excluded from the scope of this paper.

However, in manufacturing plants, conventional methods are inaccurate to improve yield, because they fail to extract underlying features from complex data [6], [7]. These methods include SPC and derived techniques such as FDC, design of experiments, or spatial mapping analysis. Component specification changes, mean process shift and variance reduction are well-known SPC techniques, while control charts aim at monitoring processes in order to detect abnormal drifts but cannot point out which parameters impact them.

When studying data mining techniques in semiconductor fabrication, the most widely used method is classification, even if it generally focuses on very specific process stages, such as cleaning [8], or photolithography [9]. The aim of classification is to build a classifier by induction from a set of pre-classified instances (the sensor measures). The classifier is then used to categorize "unlabeled" instances. Decision tree induction is the most representative approach in the field [10].

Clustering methods correspond to a particular classification of values into clusters. Among the relevant hierarchical algorithms that search to minimize a formal objective function, the most widely used is K-means clustering [11]. K-means remains also the simplest and most commonly used nonhierarchical algorithm employing a squared error criterion [12].

Prediction systems search to perform automatic discovery of significant parameters having an impact on the yield. Genetic programming [13] or neural networks [8] are therefore employed. Input data are first grouped into categorical classes. Field engineers can then build the relationship between the low yield lots and the in-line measurements at specific stages and, by the way, use these measurements to predict the future line yield.

Finally, only association rules are not often used to try to enhance yield. In [14], the authors used a modified *a priori* algorithm in LCD panel manufacturing to locate machines with low yield after completion of processes, and thus to improve the yield rate. In [15], correlations are also analyzed between combinations of used tools and defective products. Other relevant approaches do not directly deal with the semiconductor area [16].

C. Our Approach

We present hereafter a whole KDD model based on specific association rules. Within this framework, and in collaboration with STMicroelectronics (STM) and ATMEL (ATM), this paper is focused on the detection of the main control parameters impacting the yield. The goal is not to directly enhance the yield, but to propose indicators to which special attention should be paid in further production cycles through, for example, the construction of yield enhancement models.

Our *post hoc* analysis is based on comma-separated values (CSV) files of real valued measurements associated with production lots. These data have themselves been extracted in a previous step from very large manufacturer databases, covering the four fabrication units mentioned at the beginning of Section I-B. The main characteristic of the CSV 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: a particular column of the file, the yield. To detect these correlations, we introduce the concept of decision correlation rules, a restriction of correlation rules containing a value of one target column. In order to compute these rules:

1) We use the lectic order [17] to browse the powerset lattice (the search space).

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

2) We propose the concept of contingency vector: a new approach to contingency tables.
3) We show how to build the contingency vector of a pattern with a cardinality i with the contingency vector of one of its subsets with a cardinality $i-1$ (which is impossible with contingency tables).
4) We take advantage of the lectic order, the contingency vectors and the recursing mechanisms of construction to propose the LHS-CHI2 algorithm.

This paper is organized as follows. In Section II, the bases of association and correlation rules, and of the lectic order are recalled. Section III describes the concepts used for mining decisional correlation rules and our algorithm. In Section IV, we expose the other functions of the software—called *MineCor*—developed for mining decisional correlation rules. Experiments are detailed in Section V. As a conclusion, we summarize our contributions and outline some research perspectives.

II. RELATED WORK

In this section, we recall the definitions of association rules, correlation rules [18], and lectic order [17]. Then, we introduce the LS algorithm [19]. It allows the browsing of the search space according to the lectic order.

A. Statement of the Problem

An association rule is an approximate implication $X \rightarrow Y$ between two sets of items X and Y. Two measures are used to extract such rules: 1) support: the proportion of transactions (rows) containing X and Y, and 2) confidence: the ratio between the support of Y and the support of X (the degree of truth of the implication).

Example 1: The relation example r of Table I illustrates the introduced concepts. The BC pattern has a support equal to 4, and the rule $B \rightarrow C$ has a confidence equal to 2/3. This means that two thirds of the transactions including pattern B also contain pattern C.

Agrawal *et al.* [20] introduced levelwise algorithms for the computation of association rules in reasonable response times. Because the underlying semantics of an association rule are fairly poor, Wu *et al.* [21] introduced literal sets and proposed the computation of positive and/or negative association rules such as $\neg X \rightarrow Y$.

A literal is a pattern $X\overline{Y}$ in which X is also called the positive part and $\overline{Y}$ the negative part. To compute such rules, the authors still use the support-confidence platform by redefining the support of a literal: the number of transactions of the binary relation including X and containing no 1-item (item of cardinality 1) of Y.

Example 2: With the relation example r, the literalset $B\overline{C}$ has a support equal to 2. As a consequence, the association rule $B \rightarrow \overline{C}$ has a confidence equal to 1/3. This means that one third of the transactions containing pattern B does not include pattern C.

Brin *et al.* [18] proposed the extraction of correlation rules. The platform is no longer based on the support nor the confidence of the rules, but on the chi-squared statistical measure, written χ^2. The use of χ^2 is well-suited for several reasons: 1) it is a more significant measure in a statistical way than an association rule; 2) the measure takes into account not only the presence but also the absence of the items; and 3) the measure is nondirectional, and can thus highlight more complex existing links than a "*simple*" implication.

The crucial problem, when computing correlation rules, is the memory usage required by levelwise algorithms. For a pattern X, the computation of the χ^2 function is based on a table including $2^{|X|}$ cells. Thus, at level i, C_n^i candidates (where n is the number of values of r) have to be generated and stored, in the worst case scenario, as well as the associated contingency tables. With cells encoded over 2 Bytes, corresponding storage space requires 2.5 GB of memory at the third level, and 1.3 TB at the fourth level. This is why Brin *et al.* [18] computed only correlations between two values of a binary relation. Using an end-user threshold *MinCor*, Grahne *et al.* [22] showed that the "$\chi^2(X) \geq MinCor$" constraint is monotone. Consequently, the resulting set of rules is a convex space [23], which can be represented by its minimal border [24], noted L. In this paper, the author proposed a levelwise algorithm to compute L, and used an approximation to compute the χ^2 value of any pattern belonging to that convex space.

B. Correlation Rules

Let r be a binary relation (a transaction database) over a set of items $\mathcal{R} = \mathcal{I} \cup T$. In our approach, $\mathcal{I}$ represents the values (the items) of the binary relation used as analysis criteria, and $\mathcal{T}$ is a target attribute. For a given transaction, the target attribute does not necessarily have a value. The computation of the value for the χ^2 function for an item $X \subseteq \mathcal{R}$ is based on its contingency table. In order to simplify the notation, we first introduce the lattice of the literalsets associated with a pattern $X \subseteq \mathcal{R}$. This set contains all the literalsets that can be built up given X, and with a cardinality $|X|$.

Definition 1 (Literalset Lattice): Let $X \subseteq \mathcal{R}$ be a pattern. We denote by $\mathbb{P}(X)$ the literalset lattice associated with X: $\mathbb{P}(X) = \{Y\overline{Z}$ such that $X = Y \cup Z$ and $Y \cap Z = \emptyset\} = \{Y\overline{Z}$ such that $Y \subseteq X$ and $Z = X \backslash Y\}$.

Example 3: The literalset lattice associated with $X = \{A, B, C\}$ contains the following elements: $\{ABC, AB\overline{C}, AC\overline{B}, BC\overline{A}, A\overline{BC}, B\overline{AC}, C\overline{AB}, \overline{ABC}\}$.

Definition 2 (Contingency Table): For a given pattern X, its contingency table, noted $CT(X)$, contains exactly $2^{|X|}$ cells. Each cell stores the support of a literalset $Y\overline{Z}$ belonging to the literalset lattice associated with X.

TABLE II
CONTINGENCY TABLE OF PATTERN BC

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

Example 4: With the relation example r given in Table I, Table II shows the contingency table of pattern BC.

For each cell $Y\overline{Z}$ of $CT(X)$, we compute its expectation value: the theoretical frequency in case of independence of the 1-items included in $Y\overline{Z}$ [see (1)]

$$E(Y\overline{Z}) = |r| * \prod_{y \in Y} \frac{Supp(y)}{|r|} * \prod_{z \in Z} \frac{Supp(\overline{z})}{|r|}. \quad (1)$$

Formula (2) finally computes the value of the χ^2 function for a pattern X

$$\chi^2(X) = \sum_{Y\overline{Z} \in \mathbb{P}(X)} \frac{(Supp(Y\overline{Z}) - E(Y\overline{Z}))^2}{E(Y\overline{Z})}. \quad (2)$$

Brin *et al.* [18] showed that there is a single degree of freedom between the items. A table giving the centile values with regard to the χ^2 value for X can be used in order to obtain the correlation rate for X [25].

Example 5: Continuing our example, $\chi^2(BC) \simeq 0.28$, which corresponds to a correlation rate of about 45%.

Unlike association rules, a correlation rule is not represented by an implication but by the patterns for which the value of the χ^2 function is larger than a threshold.

Definition 3 (Correlation Rule): Let $MinCor$ be a threshold (≥ 0), and $X \subseteq \mathcal{R}$ a pattern. If the value for the χ^2 function for X is larger than or equal to $MinCor$, then this pattern represents a valid correlation rule.

Many authors have proposed additional constraints to evaluate whether a correlation rule is semantically valid [26]. Generally, the Cochran criteria are used: 1) all literalsets of a contingency table must have an expectation value not equal to zero (which never happens in our context), and 2) 80% of them must have a support larger than 5% of the whole population. This last criterion has been generalized by Brin *et al.* [18] as follows: $MinPerc$ of the literalsets of a contingency table must have a support larger than $MinSup$, where $MinPerc$ and $MinSup$ are also thresholds.

Example 6: Let $MinCor = 0.25$, then the correlation rule materialized by the BC pattern is valid ($\chi^2(BC) \simeq 0.28$). However, the correlation rule represented by the Bt_1 pattern is not valid ($\chi^2(Bt_1) \simeq 0.1$).

C. Lectic Order

The lectic order, noted $<_{lec}$, enumerates all the subsets of an itemset $\mathcal{I}$. This order allows the closed lattice of a binary relation to be computed [17], or to serve as a basis for the computation of the partition cube [19]: a lossless reduction of the data cube.

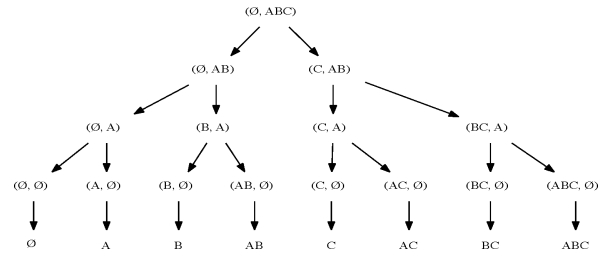

Fig. 1. Execution tree of LS for $\mathcal{I} = \{A, B, C\}$.

Definition 4 (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 $X <_{lec} Y \Leftrightarrow max_{\preceq}(X \backslash (X \cap Y)) \preceq max_{\preceq}(Y \backslash (X \cap Y))$.

Example 7: Let us consider the set $\mathcal{I} = \{A, B, C\}$, totally ordered according to the lexicographic order. The enumeration of the subsets of $\mathcal{I}$, according to the lectic order, produces the following result: $\emptyset <_{lec} A <_{lec} B <_{lec} AB <_{lec} C <_{lec} AC <_{lec} BC <_{lec} ABC$.

In order to enumerate all the subsets of $\mathcal{I}$ according to the lectic order, the lectic subset algorithm, noted LS [19], [27], is used. It is a simplified version of Algorithm 2 (limited to lines 1–7, 12). The associated execution tree is a balanced tree, based on a double recursive call. Being given a node of the tree (representing a pattern $X \subseteq \mathcal{I}$), the left subtree generates subpatterns of X not containing $max_{\preceq}(X)$, whereas the right subtree leads to subpatterns of X containing $max_{\preceq}(X)$.

Example 8: Fig. 1 shows the execution tree of the LS Algorithm for $\mathcal{I} = \{A, B, C\}$.

Proposition 1 expresses the fact that the lectic order is compatible with the antimonotone constraints. Consequently, we can modify the LS algorithm to take into account a conjunction of antimonotone constraints.

Proposition 1: Let be $X,\ Y \subseteq \mathcal{I}$ two itemsets. If $X \subset Y$, then $X <_{lec} Y$ [17].

III. LHS-CHI2 ALGORITHM

In this section, we introduce the contingency vectors: another representation of the contingency tables. We show that, for a given pattern $X \cup A$ ($X \subseteq \mathcal{R}$, $A \in \mathcal{R} \backslash X$), the computation of its contingency vector is possible using the contingency vector of X and the list of the row identifiers of the relation containing A. Then, we present the concept of the decision correlation rule: a restriction of correlation rules, in such a way that only the rules containing a value of the target attribute are kept. Finally, in order to compute these rules, we describe the LHS-CHI2 Algorithm.

A. Contingency Vectors

A literal $Y\overline{Z}$, belonging to the literalset lattice associated with a pattern X, is represented in a computer with vectors of $|X|$ bits. For a 1-item $x \in X$, the value of the bit vector has a value of 1 if $x \in Y$ (the 1-item belongs to the positive part of the literal), and 0 otherwise. Thus, comparing two literals $Y_1\overline{Z_1}$ and $Y_2\overline{Z_2}$ belonging to the literalset lattice associated with pattern X, consists in comparing each integer corresponding

to the binary value of the associated bit vector. The comparison is equivalent to extending the definition of the lectic order to the literalset one.

This order allows the total ordering of the whole literalset lattice associated with pattern X.

Definition 5 (Lectic Order for a Literalset): Let $X \subseteq \mathcal{R}$ be a pattern, $Y_1\overline{Z_1}$ and $Y_2\overline{Z_2}$ two elements of the literalset lattice associated with the X pattern. The definition of the lectic order is extended over the literalsets as follows: $Y_1\overline{Z_1} <_{lec} Y_2\overline{Z_2}$ if and only if $Y_1 <_{lec} Y_2$.

Example 9: The literalset lattice associated with the pattern $X = \{A, B, C\}$ according to the lectic order is the following: $\overline{ABC} <_{lec} A\overline{BC} <_{lec} B\overline{AC} <_{lec} AB\overline{C} <_{lec} C\overline{AB} <_{lec} AC\overline{B} <_{lec} BC\overline{A} <_{lec} ABC$.

Definition 6 (Equivalence Class Associated with a Literal): Let $Y\overline{Z}$ be a literal. Let us denote by $[Y\overline{Z}]$ the equivalence class associated with the literal $Y\overline{Z}$. This class contains the set of transaction identifiers of the relation including Y and containing no value of Z (i.e., $[Y\overline{Z}] = \{i \in Tid(r)$ such that $Y \subseteq Tid(i)$ and $Z \cap Tid(i) = \emptyset\}$).

Example 10: With our relation example (see Table I), we have $[B\overline{C}] = \{5, 6\}$.

Proposition 2: Let $X \subseteq \mathcal{R}$ be a pattern. The union of the equivalence classes $[Y\overline{Z}]$ of the literalset lattice associated with X is a partition [28] of the identifiers of relation r. In other words

$$\bigcup_{Y\overline{Z} \in \mathbb{P}(X)} [Y\overline{Z}] = Tid(r).$$

Definition 7 (Contingency Vector): Let $X \subseteq \mathcal{R}$ be 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.

Proposition 2 ensures that each transaction identifier belongs only to one single equivalence class. Consequently, for a given pattern X, its CV is an exact representation of its contingency table. To derive the contingency table from a contingency vector, it is sufficient to compute the cardinality of each of its equivalence classes. If the literalsets, related to the equivalence classes of a CV, are ordered according to the lectic order, it is possible to know the literal relative to a position i of a contingency vector ($i \in [0; |X|-1]$). This is because the literal and the integer i have the same binary coding.

Example 11: With our sample relation (see Table I), the contingency vector associated with the BC pattern is the following: $CV(BC) = \{[\overline{BC}], [B\overline{C}], [C\overline{B}], [BC]\} = \{\{9, 10\}, \{5, 6\}, \{7, 8\}, \{1, 2, 3, 4\}\}$.

Theorem 1 is the main result of our paper. It shows how to compute the CV of the $X \cup A$ pattern given the CV of X and the set of identifiers of the relation containing pattern A.

Theorem 1: Let $X \subseteq \mathcal{R}$ be a pattern and $A \in \mathcal{R} \backslash X$ a 1-item. The contingency vector of $X \cup A$ can be computed given the contingency vectors of X and A as follows:

$$CV(X \cup A) = (CV(X) \cap [\overline{A}]) \cup (CV(X) \cap [A]). \quad (3)$$

Example 12: With the relation example (see Table I), 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 Theorem 1, the contingency vector of BC is the following: $CV(BC) = \{\{9, 10\}, \{5, 6\}, \{7, 8\}, \{1, 2, 3, 4\}\}$. Thus, we retrieve the result of Example 11.

Algorithm 1 CREATE_CV Algorithm

Input: $CV(X)$ contingency vector of X, $Tid(A)$
Output: contingency vector of $X \cup A$ sorted according to the lectic order

1: $CV(Z) := \{\emptyset\}$
2: **for all** Equivalence classes $[Y\overline{Z}] \in \mathbb{P}(X)$ according to the lectic order **do**
3: $\quad CV(Z) := CV(Z) \cup ([Y\overline{Z}] \cap (Tid(r) \backslash (Tid(A))) \cup ([Y\overline{Z}] \cap Tid(A))$
4: **end for**
5: **return** $CV(Z)$

Algorithm 1 is used, given the CV of a pattern X and the set of the transaction identifiers containing a 1-item A, to build the CV of the $X \cup A$ pattern sorted according to the lectic order over the literalset lattice $\mathbb{P}(X \cup A)$. Line 3 is an adaptation of Theorem 1 to our context.

The computation of a CV needs one database scan, and the following transition to the associated CT another one (overheads are ignored). This leads to a complexity of $2 * |r|$ or $\mathcal{O}(|r|)$, whatever the number of cells in the CT. A classical computation of a CT at level i also needs one database scan; but here, in the worst case, each of the CT cells is involved in one operation, which globally forces $2^i * |r|$ operations. Because 2^i is generally much smaller in comparison to $|r|$, the complexity is also of $\mathcal{O}(|r|)$. But when going into detail, the difference between the two methods is $2^{i-1} * |r|$ operations.

B. Decision Correlation Rules

Definition 8 (Decision Correlation Rules): Let $X \subseteq \mathcal{R}$ be a pattern, and $MinCor$ a given threshold. X represents a valid decision correlation rule if and only if:

1) X contains a value of the target attribute $\mathcal{T}$;
2) $\chi^2(X) \geq MinCor$.

Example 13: With our relation example (see Table I), if $MinCor = 0.25$, the decision correlation rule materialized by the BCt_1 pattern is a valid rule because:

1) $t_1 \in \mathcal{T}$ and $t_1 \in BCt_1$;
2) $\chi^2(BCt_1) \simeq 0.28$ ($\geq MinCor$).

The lectic hybrid subset-Chi2 algorithm, or LHS-CHI2, permits to extract the whole set of decision correlation rules for a relation r satisfying the threshold constraint $MinCor$ for the χ^2 function. This algorithm is an adaptation of the LS Algorithm to our context, by taking into account contingency vectors. Moreover, we added several monotone and antimonotone constraints in order to prune the search space [22].

1) A value of the target attribute must be present in the extracted pattern (monotone constraint).
2) As the χ^2 computation has no significance for a 1-item, we only examine patterns of cardinality larger than or equal to two (monotone constraint).

3) Since the χ^2 function is an increasing function, we impose a maximum cardinality, noted *MaxCard*, on the number of 1-items belonging to the patterns to examine (antimonotone constraint).
4) All literalsets of a *CT* must have an expectation value not equal to zero (antimonotone constraint).
5) Because the obtained rules must have a semantics on the relation, at least *MinPerc* of the cells of a *CT* must have a support larger than or equal to *MinSup*. This constraint is expressed in our algorithm by the *CtPerc* predicate, which parameters are the contingency vector, *MinPerc* and *MinSup* (antimonotone constraint).

Laporte *et al.* [19] modified the LS Algorithm in order to compute "iceberg" data cubes. The authors included an antimonotone constraint threshold, evaluated before the second recursive call of the LS Algorithm; they used a pruning step with the negative border [29] in order to only examine the most "*interesting*" cuboids (patterns in our context). In the same spirit, we modify LS in order to take into account the five constraints above, and to compute the χ^2. The result is an algorithm requiring, in the worst case, $|\mathcal{R}|+MaxCard+1$ *CVs* in memory. We need $|\mathcal{R}|$ *CVs* for the 1-items, the height of our tree is bounded by *MaxCard*, and we need an additional *CV* for the current node computation. This value has to be compared to the number of contingency tables to be computed at each level using a levelwise algorithm (see the end of Section II-A).

Proposition 1 justifies the inclusion of these constraints into our algorithm. However, we do not carry out pruning using the negative border. Instead, we use the positive border [29] relating to predicate *CtPerc*. The use of the positive border is justified on the basis of the experiments carried out by Flouvat *et al.* [30]. The authors showed that the positive border is of highly reduced cardinality in comparison with the negative one. As a consequence, the satisfiability tests of the antimonotone constraints are faster when the positive border is used. In our context, we make sure that the *Z* pattern, used as a parameter within the second recursive call of the algorithm, has all its direct subsets included in one of the elements of the positive border (line 8). Let us emphasize that this test is carried out in the *AprioriGen* function [20] during the generation of the candidates of level $i+1$ using the frequent i-itemsets. If pattern *Z* is a candidate, then we compute its contingency vector by making sure that the literalsets relating to the classes of equivalence are sorted according to the lectic order (line 9) by calling Algorithm 1. If the pattern satisfies the antimonotone constraints (line 10), we update the positive border (line 11), and carry out the second recursive call of the algorithm (line 12). The monotone constraints are evaluated on the leaves of the execution tree (line 1). By convention, we have $CV(\emptyset) = \{Tid(R),\ \emptyset\}$. The positive border is initialized with $\{\emptyset\}$. The pseudo code of LHS-CHI2 is provided in Algorithm 2. The first recursive call to LHS-CHI2 is carried out with $X = \emptyset$ and $Y = \mathcal{R}$.

Example 14: The results of LHS-CHI2 with $MinSup = 0.2$, $MinPerc = 0.25$, and $MinCor = 0.25$ for our relation example (see Table I) are shown in Table III.

Algorithm 2 LHS-CHI2 Algorithm

Input: *X* and *Y* two patterns
Output: $\{itemset\ Z \subseteq X$ such that $\chi^2(Z) \geq MinCor\}$

```
1: if Y = ∅ and |X| ≥ 2 and ∃c ∈ C : c ∈ X and χ²(X) ≥ MinCor then
2:    Output X, χ²(X)
3: end if
4: A := max(Y)
5: Y := Y\{A}
6: LHS-CHI2(X, Y)
7: Z := X ∪ {A}
8: if ∀z ∈ Z, ∃W ∈ BD⁺ : {Z\z} ⊆ W then
9:    VC(Z) := CREATE_CV(CV(X), Tid(A))
10:   if |Z| ≤ MaxCard and
      CtPerc(CV(Z), MinPerc, MinSup) then
11:      BD⁺ := max⊆(BD⁺ ∪ Z)
12:      LHS-CHI2(Z, Y)
13:   end if
14: end if
```

TABLE III
RESULTS OF THE LHS-CHI2 ALGORITHM OVER TABLE I

Decision Correlation Rule	χ^2 Value
At_1	0.48
BCt_1	0.28
BFt_1	0.28

IV. MINECOR SOFTWARE

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 CSV files given by our manufacturer partners (see the end of Section I), we have first performed column elimination and discretization stages [1], [31]. These steps, known as data cleaning or cleansing in the literature, are summarized in Sections IV-A and IV-B. The output of the two steps is placed into a feature database, which serves as a source for the data mining phase. Finally, after the mining step, the results are interpreted, what is resumed in Section IV-C.

A. Preprocessing Stage

The first step of data cleaning is the preprocessing stage. Data has to be prepared for two reasons: 1) if each value of each column is considered as a single item, the search space explodes combinatorially, and results cannot be provided in a reasonable amount of time, and 2) we cannot expect this task to be performed by an expert, because manual cleaning of data is laborious and subject to errors.

Preprocessing consists in the reduction of the data structure [32] by eliminating columns (and rows) of low significance. Such situations can result, for example, from the dysfunction of one or more sensors, or from the occurrence of a maintenance step. As a consequence, corresponding columns contain many null or default values, and must be deleted from the source file. 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. Another classical technique is the elimination of columns having small standard deviation. Since all values are almost the same, we consider that they do not have a significant impact on the result; but their inclusion pollutes the search space and reduces the response time of *MineCor*. Attention is finally paid to missing or inconsistent values, such as "outliers" and noisy columns. Elimination is performed through thresholds specified by the end-user.

B. Discretization Stage

Discrete values deal with value intervals, 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. In this section, we only summarize these methods. Discretization can be performed [33]: 1) in a supervised or unsupervised manner, depending on whether class information is at one's disposal; 2) in a dynamic or static way: with a static discretization approach, discretization is done before the classification task; and 3) using splitting or merging techniques. In the latter case, using a bottom-up approach while examining the search space.

We represent continuous real valued columns by associating each of their values with an interval code. The bins are created either using equal-width or equal-frequency discretization, which are nonsupervised, static, and splitting methods. In both approaches, arity k is the number of intervals to use. And the different values associated with each set S are managed in the same way through initial normalization.

1) *Equal Width Discretization (EWD):* Let $\mathcal{S}$ be the set of values to be discretized, and respectively $Min_{\mathcal{S}}$ and $Max_{\mathcal{S}}$ the smallest and the largest value of $\mathcal{S}$. Each interval has a length of $l = \frac{Max_{\mathcal{S}} - Min_{\mathcal{S}}}{k}$. The computed classes are $c_1 : [Min_{\mathcal{S}}, Min_{\mathcal{S}} + l[$, $c_2 : [Min_{\mathcal{S}} + l, Min_{\mathcal{S}} + 2l[$, ...

2) *Equal Frequency Discretization (EFD):* The goal is to obtain classes having, if possible, the same number of continuous values. The Jenks' natural breaks method minimizes the in-class difference and maximizes the between-class difference [34]. This can be measured by the goodness of variance fit (GVF)

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

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 $\mathcal{S}$. Jenks' method is the best from a statistical point of view because it creates homogeneous groups. Its main drawback is the high computational complexity of the class generation, which is C_{d-1}^{k-1}, where d represents the number of distinct values in the set $\mathcal{S}$. Thus, we use instead the Fisher's exact optimization method [35] proposed for grouping n elements into k mutually exclusive and exhaustive subsets having maximum homogeneity. The partition is guaranteed to be optimal, but not unique, which is not important while the obtained time gain is. This is why the EFD method is also referred to, in the next sections of this paper, as the Fisher–Jenks' method.

				Target Attribute		Chi2
Column1 (B)		Column2 (C)		t1		0.28
[1.4000, 2.2000]	0.600	[2.8000, 4.6000]	0.600	[2.7000, 7.4500]	0.500	
Column1 (B)		Column3 (F)		t1		0.28
[1.4000, 2.2000]	0.600	[9.8000, 15.4000]	0.400	[2.7000, 7.4500]	0.500	

Fig. 2. Output produced by *MineCor*.

Example 15: Let $\mathcal{S} = \{1.8, 1.9, 2.1, 2.2, 1.3, 2.0, 0.5, 0.6, 0.5, \text{NULL}\}$ be the set to discretize. If we specify two output classes, the proposed methods produce the following results.

1) EWD: since $\frac{Max_{\mathcal{S}} - Min_{\mathcal{S}}}{2} = 1.35$, this method computes the classes $[0.6, 1.3]$, $[1.8, 2.2]$. As a consequence, the set $\mathcal{S}$ is encoded by the vector {B, B, B, B, A, B, A, A, A, - } in the output of the discretization step ("-" symbolizes the NULL value).
2) EFD: the Fisher–Jenks' method produces ten class generation possibilities. The one which maximizes the squared sum is $[0.5, 0.6]$, $[1.3, 2.2]$. The following vector is produced to represent the set $\mathcal{S}$: {B, B, B, B, B, B, A, A, A, - }. Let us underline that we retrieve here partial results presented in Table I.

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 16: Fig. 2 provides an example of output produced by *MineCor*, limited to some 3-patterns. Given a row, the last column is the computed χ^2 value for the associated decision correlation rule.

As mentioned in Section IV-B, and because EWD is the default method, the results shown are slightly different than those presented in Table III.

V. Experimental Analysis

Some representative results of the LHS-Chi2 algorithm are presented below. The comparison is made with a standard levelwise (a complete *a priori*) algorithm, hereafter called LevelWise, based on the same monotone and antimonotone constraints as those used in LHS-Chi2 (see Section III). The main difference is that the LevelWise method does not use contingency vectors but uses standard computation of contingency tables.

As emphasized in Section I-C, the experiments were done on different CSV files of real value measures supplied by STM and ATM. These files have one or more target columns, resulting from the concatenation of several measurement files. The characteristics of the datasets used for experiments can be found in Table IV. All experiments were conducted on a HP Workstation (1.8 GHz processor with a 4 GB RAM).

TABLE IV
DATASET EXAMPLES

Name	Number of Columns	Number of Rows
STM File	1281	297
ATM File	749	213

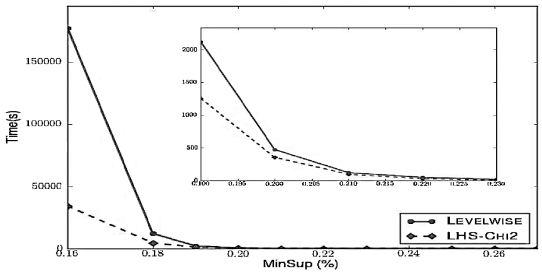

Fig. 3. Execution time with *MinPerc* = 0.34, *MinCor* = 1.6 (STM file: target1).

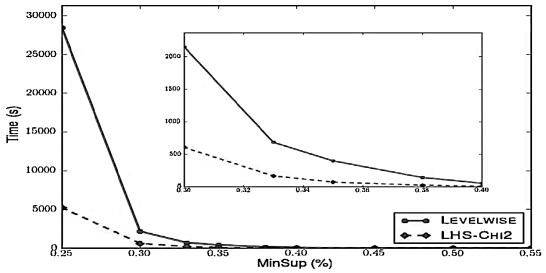

Fig. 4. Execution time with *MinPerc* = 0.24, *MinCor* = 2.8 (ATM file: target2).

Experimental results are presented on Figs. 3 to 8(c). The EWD discretization method is used in all the experiments carried out in Sections V-A to V-C.

A. Execution Times for LHS-CHI2 and LEVELWISE Algorithms

Figs. 3 and 4 show the evolution of the execution times for both methods for the two files when *MinSup* varies and *MinPerc* and *MinCor* are fixed. As the graphs point it out, the response times of our method are between 30% and 70% better than LEVELWISE, even if they remain high when using small thresholds. In each case, an increasing windowing of the results is provided for subsequent subintervals of *MinSup*.

B. Impact of the MinPerc Parameter

Fig. 5 shows the execution times for the STM file (using the same configuration as the experiment in Fig. 3) when *MinSup* and *MinCor* are constant, and when *MinPerc* varies. The staircase curve thus explains. A CT associated with a i-pattern containing 2^i cells, specifying that *MinPerc* of its cells must have the support means that $\lceil 2^i * MinPerc \rceil$ cells must have it. So, for a 3-pattern, to define a value for *MinPerc* varying between 0% and 12.49% means specifying that one single cell of the CT has to have the support, and so on. The scale is logarithmic, because response times for small values of *MinPerc* are very high (more than 13 h for LHS-CHI2, and about 69 h for LEVELWISE with *MinPerc* = 0.12).

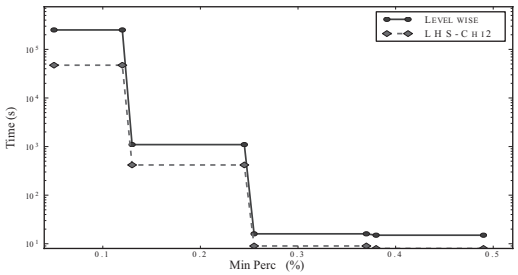

Fig. 5. Execution time with *MinSup* = 0.24, *MinCor* = 6.9 (STM file: target1).

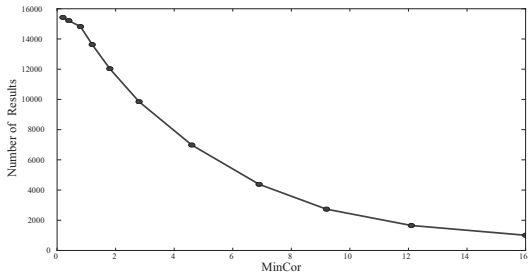

Fig. 6. Results with *MinSup* = 0.38, *MinPerc* = 0.24 (ATM file: target3).

C. Impact of the MinCor Parameter

Fig. 6 shows the number of extracted rules (identical in both methods) after mining when *MinPerc* and *MinSup* are fixed with suitable values and when *MinCor* varies. In that particular case, execution times are identical whatever the *MinCor* value, but are of the order of 2 min with LHS-CHI2, and about 17 min for LEVELWISE. This means that the *MinCor* threshold only has a small effect on performance.

D. Impact of the Discretization Stage

Figs. 7(a) and 8(a) show the number of items kept after the preprocessing and discretization stages. This number only depends on the *MinSup* threshold, while the number of bins is constant [4 in Fig. 7(a), and 6 in Fig. 8(a)]. In each example, all items with a support greater than *MinSup* are kept.

As illustrated in Figs. 7(a) and 8(a), the smaller the threshold *MinSup*, the larger the number of items kept for the mining stage, whatever the discretization method. Figs. 7(b) and 8(b) show the number of rules 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 [see Figs. 7(c) and 8(c)]. These results come from the perspective that *MineCor* tries to provide rules of "best" quality: 1) low in number; 2) significant; and 3) computed quickly.

Finally, let us emphasize that the experimental sets used in Fig. 7 produce decision correlation rules with a cardinality of 4. This is the kind of information that is of interest for semiconductor manufacturers, as well as different possible crossings using other techniques (see Section I-B) between rules of cardinality 3 and 4.

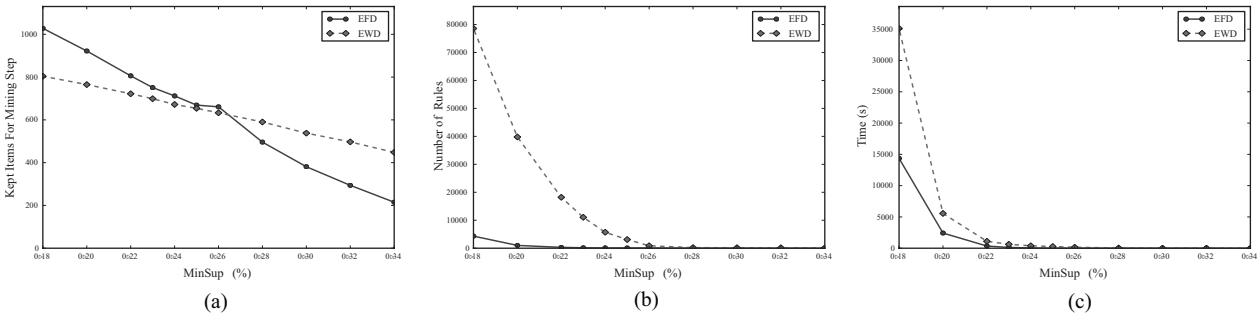

Fig. 7. Results with four intervals, *CtPerc* = 0.34, *MinCorr* = 1.6 (STM file: target1). (a) Number of items kept after discretization/preprocessing stages. (b) Number of generated decision correlation rules. (c) Execution time.

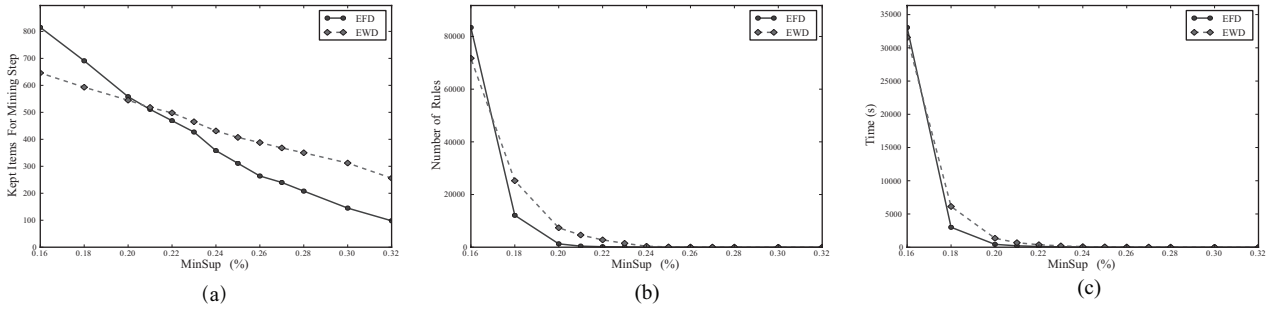

Fig. 8. Results with six intervals, *CtPerc* = 0.3, *MinCorr* = 2.8 (ATM file: target3). (a) Number of items kept after discretization/preprocessing stages. (b) Number of generated decision correlation rules. (c) Execution time.

VI. Conclusion and Future Work

In this paper, we showed the different facets of the *MineCor* software. CSV parameter measurement files given by semiconductor manufacturers (STMicroelectronics and ATMEL) are used as input, and produce as output values of parameters with most influence on the yield. To achieve this objective, we built a complete knowledge discovery in databases model, based on:

1) decision correlation rules, i.e., a restriction of correlation rules containing a target attribute value;
2) contingency vectors, i.e., an alternative representation of contingency tables, which are more concise and offer better performance related properties. We finally proposed an algorithm based on the lectic order to go through the powerset lattice.

The LHS-Chi2 algorithm is the heart of our model. It uses the inference property of the contingency vector of a pattern given the contingency vector of one of its direct subsets. The experiments show that the proposed method computes rules faster than those offered by levelwise algorithms. Moreover, we implemented two methods at the discretization stage: 1) equal width discretization, and 2) equal frequency discretization based on the Fisher–Jenks' method. Experiments show that, in most cases, the latter method produces decision correlation rules faster and of better quality. Furthermore, the software enables us to find new correlations between the parameters of the files that have been studied. As an example, approximately 25% of the correlation rules determined by the first experiment were unknown to STM, and the quasi-totality of the results obtained have been experimentally validated.

Finally, let us emphasize that the presented *post hoc* method could also be applied in real-time, i.e., associated with specific process steps, from the moment on the relevant configuration parameters are set up in an optimal way. Moreover, our KDD model could be used in other domains than wafer manufacturing.

Some new issues to our work are: 1) to optimize memory management in order to increase the performance of LHS-Chi2; 2) to compare our approach with other mining methods; 3) 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 more significant results; and 4) to broaden the correlation rule extraction problem on items to those on literalsets.

References

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Waltham, MA: Morgan Kaufmann, 2000.
[2] G. Piatetsky-Shapiro and C. J. Matheus, "The interingness of deviations," in *Proc. KDD Workshop*, 1994, pp. 25–36.
[3] M. Klemettinen, H. Mannila, and H. Toivonen, "A data mining methodology and its application to semi-automatic knowledge acquisition," in *Proc. DEXA Workshop*, 1997, pp. 670–677.
[4] A. Choudhary, J. Harding, and M. Tiwari, "Data mining in manufacturing: A review based on the kind of knowledge," *J. Intell. Manuf.*, vol. 20, no. 5, pp. 501–521, 2009.
[5] J. Pan and D. Tai, "Implementing virtual metrology for in-line quality control in semiconductor manufacturing," *Int. J. Syst. Sci.*, vol. 40, no. 5, pp. 461–470, 2009.
[6] L. Huisman, *Data Mining and Diagnosing IC Fails*. Berlin, Germany: Springer, 2005.
[7] M. A. Karim, S. K. Halgamuge, A. J. R. Smith, and A. L. Hsu, "Manufacturing yield improvement by clustering," in *Proc. ICONIP*, vol. 3. 2006, pp. 526–534.

[8] D. Braha and A. Shmilovici, "Data mining for improving a cleaning process in the semiconductor industry," *IEEE Trans. Semicond. Manuf.*, vol. 15, no. 1, pp. 91–101, Feb. 2002.
[9] D. Braha and A. Shmilovici, "On the use of decision tree induction for discovery of interactions in a photolithographic process," *IEEE Trans. Semicond. Manuf.*, vol. 16, no. 4, pp. 644–652, Nov. 2003.
[10] D.-H. Baek, I.-J. Jeong, and C. H. Han, "Application of data mining for improving yield in wafer fabrication system," in *Proc. ICCSA*, vol. 4. 2005, pp. 222–231.
[11] D. MacKay, *An Example Inference Task: Clustering*. Cambridge, U.K.: Cambridge University Press, 2003, ch. 20, pp. 284–292.
[12] C.-F. Chien, W.-C. Wang, and J.-C. Cheng, "Data mining for yield enhancement in semiconductor manufacturing and an empirical study," *Expert Syst. Appl.*, vol. 33, no. 1, pp. 192–198, 2007.
[13] L. Rokach and O. Maimon, "Data mining for improving the quality of manufacturing: A feature set decomposition approach," *J. Intell. Manuf.*, vol. 7, no. 3, pp. 285–299, 2006.
[14] C. Huang and R. Chen, "Application of new *a priori* algorithm MDNC to TFT-LCD array manufacturing yield improvement," *Int. J. Comput. Applicat. Technol.*, vol. 28, nos. 2–3, pp. 161–168, 2007.
[15] W. Chen, S. Tseng, and C. Wang, "A novel manufacturing defect detection method using association rule mining techniques," *Expert Syst. Applicat.*, vol. 29, no. 4, pp. 807–815, Nov. 2005.
[16] H. Sadoyan, A. Zakarian, and P. Mohanty, "Data mining algorithm for manufacturing process control," *Int. J. Adv. Manuf. Technol.*, vol. 28, nos. 3–4, pp. 342–350, 2006.
[17] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Berlin, Germany: Springer, 1999.
[18] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Proc. SIGMOD Conf.*, 1997, pp. 265–276.
[19] M. Laporte, N. Novelli, R. Cicchetti, and L. Lakhal, "Computing full and iceberg datacubes using partitions," in *Proc. ISMIS*, 2002, pp. 244–254.
[20] 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*. Cambridge, MA: AAAI/MIT Press, 1996, pp. 307–328.
[21] 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.
[22] G. Grahne, L. V. S. Lakshmanan, and X. Wang, "Efficient mining of constrained correlated sets," in *Proc. ICDE*, 2000, pp. 512–521.
[23] M. Vel, *Theory of Convex Structures*. Amsterdam, The Netherlands: North-Holland, 1993.
[24] H. Hirsh, "Generalizing version spaces," *Mach. Learning*, vol. 17, no. 1, pp. 5–46, 1994.
[25] M. Spiegel and L. Stephens, *Outline of Statistics*. New York: McGraw-Hill, 1998.
[26] D. Moore, "Measures of lack of fit from tests of chi-squared type," *J. Statist. Planning Inference*, vol. 10, no. 2, pp. 151–166, 1984.
[27] C. Ernst and A. Casali, "Extraction de règles de corrélation décisionnelles," in *Proc. EGC*, 2009, pp. 187–192.
[28] D. Laurent and N. Spyratos, "Partition semantics for incomplete information in relational databases," in *Proc. SIGMOD Conf.*, 1988, pp. 66–73.
[29] H. Mannila and H. Toivonen, "Levelwise search and borders of theories in knowledge discovery," *Data Min. Knowl. Discov.*, vol. 1, no. 3, pp. 241–258, 1997.
[30] F. Flouvat, F. D. Marchi, and J.-M. Petit, "A thorough experimental study of datasets for frequent itemsets," in *Proc. ICDM*, 2005, pp. 162–169.
[31] D. Pyle, *Data Preparation for Data Mining*. Waltham, MA: Morgan Kaufmann, 1999.
[32] 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*. Boston, MA/Dordrecht, The Netherlands/London, U.K.: K. A. Publishers, 2003, pp. 107–117.
[33] 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.
[34] G. Jenks, "The data model concept in statistical mapping," in *International Yearbook of Cartography*, vol. 7, E. Imhof, Ed. Gutersloh, 1967, pp. 186–190.
[35] W. Fisher, "On grouping for maximum homogeneity," *J. Am. Statist. Assoc.*, vol. 53, pp. 789–798, Dec. 1958.

Alain Casali received the Ph.D. degree in computer science from Aix Marseilles University, Aix-en-Provence/Marseille, France, in 2005.

He is currently an Assistant Professor with Aix Marseilles University—IUT (Technical Institute) of Aix-en-Provence, and is a member of the Laboratoire d'Informatique Fondamentale de Marseille. He is currently doing research on lattice algorithmics and multidimensional database mining.

Christian Ernst received the Ph.D. degree in computer science from the University of Nice, Nice, France, in 1987.

He is currently an Associate Professor with the Provence Microelectronics Center, Ecole des Mines de Saint-Etienne, Gardanne, France. After spending 6 years with two software engineering companies, he joined the Institut Superieur de MicroElectronique Appliquee Graduate Engineering School, Marseille, France, where he was the Director of Studies from 1996 to 2005. He has been involved in more than 20 industrial and academic research-oriented projects. His current research interests include algorithmics, data base systems, and data mining techniques.