International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:3, 2013

M2LGP: Mining Multiple Level Gradual Patterns

Yogi Satrya Aryadinata, Anne Laurent, and Michel Sala

***Abstract*—**Gradual patterns have been studied for many years as they contain precious information. They have been integrated in many expert systems and rule-based systems, for instance to reason on knowledge such as "the greater the number of turns, the greater the number of car crashes". In many cases, this knowledge has been considered as a rule "the greater the number of turns → the greater the number of car crashes" Historically, works have thus been focused on the representation of such rules, studying how implication could be defined, especially fuzzy implication. These rules were defined by experts who were in charge to describe the systems they were working on in order to turn them to operate automatically. More recently, approaches have been proposed in order to mine databases for automatically discovering such knowledge. Several approaches have been studied, the main scientific topics being: how to determine what is an relevant gradual pattern, and how to discover them as efficiently as possible (in terms of both memory and CPU usage). However, in some cases, end-users are not interested in raw level knowledge, and are rather interested in trends. Moreover, it may be the case that no relevant pattern can be discovered at a low level of granularity (e.g. city), whereas some can be discovered at a higher level (e.g. county). In this paper, we thus extend gradual pattern approaches in order to consider multiple level gradual patterns. For this purpose, we consider two aggregation policies, namely horizontal and vertical.

***Keywords*—**Gradual Pattern.

I. Introduction

GRADUAL pattern mining has been recently introduced as the topic addressing the automatic discovery of gradual patterns from large databases. Such databases are structured over several attributes which domains are totally ordered, considering a relation ≤. Example 1 reports an example of such a database.

Example 1: We consider the database containing sales (number of kg) from a shop selling fruits. Each tuple from the database corresponds to a cashier ticket.

Yogi Satrya Aryadinata, Anne Laurent, and Michel Sala are with the Montpellier Laboratory of Informatics, Robotics, and Microelectronics (LIRMM in French), Montpellier, France (e-mail: aryadinata@lirmm.fr, laurent@lirmm.fr, michel.sala@lirmm.fr).

TABLE I
FRUIT SALES DATABASE

Id	Pineapples	RedApples	Cherries	Durian
T1	0	3	0	0
T2	2	1	1	0
T3	4	4	2	3
T4	2	1	1	1
T5	7	0	3	0

Several approaches have been defined in order to extract patterns like "The greater the Pineapple, the greater the Cherries" (denoted by {(Pineapples, ↑), (Cherries, ↑)}), these work focusing on two main points pointed below.

The first point is to study how to define the support of such a pattern, i.e. the extend to which this pattern is true. In our example, the support is associated with the proportion of tickets matching some properties. Then, only frequent patterns, i.e. patterns which support is greater than a given threshold, will be considered as being relevant. In the literature, several propositions have been done. Reference [15] considers statistical methods (linear regression). [10] considers a heuristic. [9] considers an exhaustive counting of the support based on precedence graphs, while [16] considers a counting based on rank correlation. Fuzzy approaches have been considered in [1], [18] in order to soften the approach.

For instance, it can be noticed that pineapples and cherries sold on Tickets 1, 2, 3 and 5 can be seen as increasing together. We thus have 4 tickets out of 5 on which we can claim that when the sale of pineapples grows, then the sale of cherries grows, and conversely. It is very important to note that we consider patterns and not rules, which is the reason why we consider a reversible definition.

The second point is to study how to automatically extract such patterns from large databases. Approaches can be compared regarding their efficiency. Methods have been developed by adopting pattern mining algorithms. The efficiency thus relies on the anti-monocity property which states that no frequent pattern containing n attributes can be built over a pattern containing a subset of these n attributes. For instance, if the pattern "The greater the Pineapples, the greater the RedApples" is not relevant, then there no way that the pattern "The greater the Pineapples, the greater the RedApples, the greater the Cherries" is relevant. This property allows to scan the search space in a very efficient manner, by pruning subspaces as they do not have any chance to contain relevant patterns. In [9], binary matrices are defined in order to speed up the calculation, while parallel approaches have been considered in [12], [17].

These approaches are of great interest, they have been successfully applied on several application domains, such as psychology and biology, where discovering co-variation of gene expressions is very important, and can hardly be performed using traditional statistical methods. The number of attributes (genes) does indeed not allow to test all the possibilities. Moreover, the interesting co-variations stand among several attributes and many methods stop searching after the test of 2 attributes together (e.g., statistical correlation).

However, it may be the case that considering data at a low level of granularity does not convey sense in terms of trend analysis. For instance, it may be the case that the sales within one day cannot be regarded as containing any gradually, while the same data aggregated over days (all sales from the same day are summed) may contain gradual trends. The database being considered is thus transformed in order to merge lines. An aggregation function is considered in order to fusion the values, thus leading to the same discussions that the ones from the data warehouse framework: the data being considered (sales here) may be additive, semi-additive or non additive.

In our example, Sales are additive over all attributes and over lines (tickets).

Example 2: We consider the database from Example 1, where T1, T2 and T3 are from Monday and T4 and T5 are from Tuesday.

TABLE II
VERTICAL AGGREGATION

Id	Pineapples	RedApples	Cherries	Durian
Monday (T1-3)	0	3	0	0
Tuesday (T4-5)	7	0	3	0

Conversely, it may also be the case that regarding the data attribute per attribute does not convey information, while it can be the case by aggregating columns from the database.

Example 3: For instance, the first two columns may be merged.

TABLE III
HORIZONTAL AGGREGATION

Id	WithoutKernel Pineapples + RedApples	Cherries
T1	3	0
T2	3	1
T3	8	2
T4	3	1
T5	7	3

In this paper, we thus consider these two ways of building multiple level gradual patterns. This work can be linked to data warehouses and OLAP mining as we consider data mining over several attributes (dimensions) described over several levels of granularities. However our framework is not exactly the same, especially because aggregations can be performed on some attributes and not for some other ones, meaning that two attributes may be merged, while other ones are not. This kind of operation is considered as being possible from the semantic point of view because we consider ordinal scales where the value itself does not impact. It is thus possible to compare large values over one high level attribute (e.g., RedFruits) and small ones over another low level attribute (Pineapple).

The paper is organized as follows. Section II details the preliminary definitions. Section III reports existing work on gradual pattern mining and Section IV states the problem we address, while Sections V and VI introduce our contribution, namely the definitions of what Multiple Level Gradual Patterns are and how they can be extracted.

II. PRELIMINARY DEFINITION

We introduce below the definitions taken from the literature in order to define gradual pattern mining. Unfortunately, no unique notation is available in the papers, we thus propose to consider the ones given below.

Definition 1: Gradual-Attribute. A gradual attribute I is defined over a domain $dom(I_j)$ on which an order $\leq_j$ (or simply $\leq$) is defined.

Definition 2: Gradual-DB. A gradual database is a set of tuples T defined over the schema $S = \{Id, I_1, ..., I_n\}$ of n gradual attributes where Id is an identifier (primary key).

Example 1 shows an example of a database which schema is S = {Pineapples, RedApples, Cherries} containing 5 tuples defined over three attributes which domains are IN.

Definition 3: Gradual item. A gradual item is a pair (i, v) where i is an item and v is variation $v \in \{\uparrow, \downarrow\}$. $\uparrow$ stands for an increasing variation while $\downarrow$ stands for a decreasing variation. For example, (Pineapples, $\uparrow$) is a gradual item.

Definition 4: Gradual Pattern (also known as Gradual Itemset). A gradual pattern is a set of gradual items, denoted by $GP = \{(i_1, v_1), \ldots, (i_n, v_n)\}$. The set of all gradual patterns that can be defined is denoted by GP.

For example, {(Pineapples,$\uparrow$),(RedApples,$\uparrow$)} is a gradual itemset.

Definition 5: Tuple Ordering Over a Set of Attributes A. The tuples from a gradual database are ordered by defining an order $\prec$ with respect to a gradual pattern GP $\{(i_1, v_1), \ldots, (i_n, v_n)\}$. Two tuples t and t′ can be ordered with respect to GP , denoted by $t \in GP\ t'$ if all the values of the corresponding items can be ordered with respect to the variations: for every $i_k (k \in [1,n])$, $t.i_k \leq' t.i_k$ if $v_l = \uparrow$ and $t'.i_k \leq t.i_k$ if $v_l = \downarrow$.

The support of a gradual pattern indicated to which extend it can be found in the database.

Definition 6: Gradual Support. The support of a gradual pattern over a gradual database GDB is a function sup from GP to [0,1] that holds the following property: for all $GP_1, GP_2 \in GP$, $GP_1 \in GP_2 \in supp(GP_1) \geq supp(GP_2)$.

III. RELATED WORK

A. Gradual Patten Mining

In the 70s, gradual patterns were used to model system behaviors. Patterns were designed by experts. A complete

theoretical framework of gradual rules into the fuzzy context is given in [11], with a comparison of fuzzy implication for gradual dependencies. Among them the most used is RescherGaines (RG) implication.

Regarding crisp data mining approaches, the authors agree on the definition of gradual item (which couples a classical item and a variation) and gradual itemsets (set of gradual items), but have proposed several ways of defining the support.

[15] considers a regression-based definition. [9] the authors consider another definition of the support based on the maximum proportion of tuples that can be ordered according to the gradual itemset. While [3] and [16] consider the number of tuples that are concordant and discordant, in the idea of exploiting the Kendall's tau ranking correlation coefficient [14].

[15] defines gradually as a co-variation of gradual dependency, where A gradual dependency such as the more A, the more B holds if an increase in A comes along with an increase in B. In order to identify such relationships, it proposes to perform a linear regression analysis between the two attributes. The validity of the gradual tendency is evaluated from the quality of the regression. This definition and this extraction method apply to pairs of attributes. The extension proposed by [15] to longer itemsets considers the case of fuzzy data, for which attributes contain the membership degrees of the data to modalities. It exploits this fuzzy logic framework and the fact that itemsets are interpreted as conjunction of the items they contain: a membership degree to the itemset can be computed using a t-norm, applied to the membership degrees to the items of the considered itemset. The gradual tendency is then understood as a covariance constraint between the aggregated membership degrees. Thus itemsets of length higher than 2 can be handled as itemsets of length 2.

Other works take a different point of view and interpret gradual dependencies as constraints imposed to the order induced by the attributes, and not to their numerical values: in [3] gradual dependencies are considered as generalizations of functional dependencies that replace the equality conditions by variation conditions on the values. This definition takes into account a causality relationship between the itemsets. It states that the ordering induced by attribute A must be identical to that derived from attribute B. In the case of dependencies such as the more A, the less B, the constraint imposes that the orders must be reversed.

[10] proposes an approach based on conflict sets. The authors propose a heuristic to compute the support for gradual itemsets, in a level-wise process that considers itemsets of increasing lengths. It consists in discarding, at each level, the rows whose so-called conflict set is maximal, i.e. the rows that prevent the maximal number of rows to be sorted. The selection is made by random in case of several conflict sets having same size. Note that this heuristic may lead to loose some loss. Indeed, it may be the case that a raw seam to create a lot of conflicts compared to another one, but that this number of conflicts is still lower than the conflicts generated at the next level.

Example 4: The table below reports the conflicts regarding the pattern {(Pineapples, ↑), (RedApples, ↓)}. Notethat the lines have been reordered to facilitate the reading by considering the ranking on the values over the attribute Pineapples. In the Table IV, the conflict set reports the Tickets that are in conflict with the one being considered.

TABLE IV
CONFLICT SETS

Id	Pineapples	RedApples	Cherries	Conflict Set
T1	0	3	0	{T2,T3}
T2	2	1	1	{T1,T3}
T4	2	1	1	{T3}
T3	4	4	1	{T1,T2,T3,T4}
T5	7	0	3	Ø

In [9], the support is defined as being the longest path of the precedence graph of tuples regarding a gradual pattern. For instance, we can design the graph of Tickets regarding the ordering on the gradual pattern {(Pineapples, ↑), (RedApples, ↓)} as displayed on Fig. 1. It can for example be seen that T3 is only linked with T5. T3 precedes T5 as T3.Pineapples ≤ T5.Pineapples and T3.RedApples ≥ T5.RedApples.

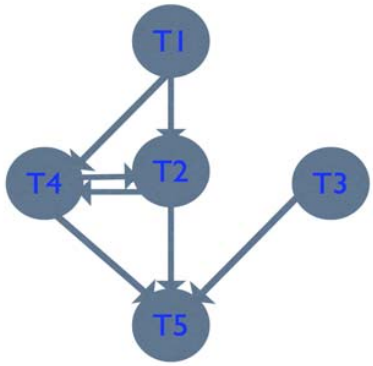

Fig. 1 Precedence Graph for {(Pineapples, ↑), (RedApples, ↓)}

The GRAANK approach from [16] interprets gradual patterns in terms of ranking correlation. Binary matrices from [9] can then be considered in order to speed up the algorithms. The problem of rank correlation has been extensively studied by statisticians, and several measures have been proposed, distinguishing between two ranks and multiple rank comparison. Regarding ranking pairs, the most used measures are the Spearman correlation and the Kendall's tau. The Kendall's tau is defined as follows: given n objects to be ranked, and σ_k, k = 1,2 two rankings where $\sigma_k(x)$ gives the rank of object x in σk ranking, the Kendall's tau relies on the definition of concordant and discordant pairs: concordant pairs (i,j) are pairs for which the rankings agree, i.e. either $\sigma_1(i) \leq \sigma_1(j)$ and $\sigma_2(i) \leq \sigma_2(j)$, or $\sigma_1(i) \geq \sigma_1(j)$ and $\sigma_2(i) \geq \sigma_2(j)$. Non concordant pairs are called discordant pairs. The

Kendall's tau is then defined as the proportion of discordant pairs, i.e. the frequency of pair-wise inversions.

Example 5: For instance, from Example 1, when considering the gradual pattern {(Pineapples, ↑), (RedApples, ↓)}.

- T1 and T2 are concordant
- T1 and T3 are discordant
- T1 and T4 are concordant
- T1 and T5 are concordant
- T2 and T3 are discordant
- T2 and T4 are concordant
- T2 and T5 are concordant
- T3 and T4 are concordant
- T3 and T5 are concordant
- T4 and T5 are concordant

Fuzzy extensions have been defined in order to deal with real life applications where data and knowledge are often not crisp. [1], [2] study the possibility that the graduality is not over all the attribute but may be hidden somewhere in the domain of values. For instance, when mining gene expression, it may be the case that there is no pattern such as "The more/less the expression of gene G_i, the more the expression of gene G_j" but that it is rather the case that the pattern "The more/less the expression of gene G_i is almost 0.2, the more the expression of gene G_j is almost 0.8". The proposal proposes a definition of such fuzzy patterns and algorithms based on genetic programming in order to discover the most relevant parts of the universe (e.g. almost 0.2 and almost 0.8 in the above example).

[17], [18] propose to consider fuzzy orders instead of crisp orders so as to tackle the problem of data where differences between values may not always convey a crisp decision. For instance, it may be the case that an expression of gene of 0.1887 may not be that lower than an expression of gene of 0.1888. The work is based on the definitions given by [4]–[6].

B. OLAP Mining and Hierarchical Data Mining

When dealing with hierarchies in databases, the main works have been done in the framework on data warehouses and OLAP (On-Line Analytical Processing). Data warehousing refers to the process of constructing and exploiting of the data warehouse. Data warehousing thus includes the integration of the data from multiple sources into a unified schema at a single location to facilitate data analysis for decision making. Thus, the construction of a data warehouse includes data integration, data cleansing, data consolidation and OLAP [7], [8]. OLAP systems are constructed in a data warehouse environment that serves as a repository of the data to be processed. Data are organized over measures (e.g., number of sales) studied with respect to dimensions (e.g., product, month, city) which can be organized with hierarchies (e.g., month-quarter-year). OLAP operations are defined to help end-users to navigate through the data. Some of the operations are dedicated to the management of hierarchies: roll-up allows to go from data described at a low level of granularity (e.g., month) to data given at a higher level of granularity (e.g., quarter).

OLAP Mining has been first introduced in 1997 by Jiawei Han as a mechanism which integrates OLAP with data mining so that mining can be performed in different portions of databases or data warehouses and at different levels of abstraction at users finger tips.

However, authors have considered multiple level data mining before studying intensively data warehouses. In [20], the beginnings of the hierarchy management in the extraction of association rules and sequential patterns are proposed. The authors suppose that the hierarchical relations between the items are represented by a set of taxonomies. They make it possible to extract association rules or sequential reasons according to several levels of hierarchy. They modify the transactions by adding for each item all of its ancestors in associated taxonomy. Then, they generate the frequent sequences while trying to filter with the maximum redundant information and by optimizing the process using several properties. However, this approach cannot be scalable in a multidimensional context. Indeed, to add on each dimension the list of the ancestors of one item in taxonomy, for each transaction, is unthinkable.

That would be equivalent, in the worst case, to multiply the size of the database by the maximum depth of a hierarchy and this for each dimension of analysis, scan of this basis would be then too much expensive.

The approach of J. Han et al. [13] is quite different. The authors tackle the association rule extraction problem, but this approach can be adapted to sequential pattern extraction. Starting at the highest level of the hierarchy, they extract the rules on each level while lowering the support when going down in the hierarchy. The process is reiterated until no rules can be extracted or until the lowest level of the hierarchy. However, this method does not make it possible to extract rules containing items of different levels. For example wine and drinks cannot cohabit in such a rule. This method thus proposes the extraction of intra level of hierarchy association rules. It thus does not make it possible to answer the general problems of extraction of the sequences on various levels of hierarchy. Furthermore, the implementation of this approach in a multidimensional context can be discussed. If several taxonomies exist (one by dimension), does one have to move on the same levels of hierarchy on various taxonomies or to combine these levels? This type of extraction can be expensive in time, because the mechanism of extraction of knowledge can be reiterated several times (depth of taxonomy), which is not inconsiderable.

[19] introduces multiple level sequential patterns. As it is proven that finding out all sequential patterns from a database is an np-complete problem, the authors propose to consider convergent and divergent sequential patterns. When considering convergent patterns, the levels of granularity of the patterns along the sequence decreases (e.g. from county to city). When considering divergent patterns, the levels of granularity of the patterns along the sequence increases (e.g. from city to county). Such patterns are of the form < (coke, Paris)(soda, Paris) > meaning that in many cases, there has

been coke sold in Paris and then soda (higher level compared to coke) sold in Paris. The authors highlight the interest of extracting the "most specific" patterns, i.e. the patterns such as there does not exist any pattern included within it (either because some items can be deleted, or because some items can be expressed at a lower level of granularity) which is frequent. Looking for search patterns is interesting as patterns at high level of granularity are often already known and do not convey relevant knowledge for the end-users.

In this section, we have presented the works related to our proposal. In the next session, we introduce our contribution M2LGP standing for Mining Multiple Level Gradual Patterns.

IV. M2LGP: Problem Statements

Starting from the definitions from Section II consolidated from the existing works, we state an original framework. We claim that many data are equipped with hierarchies, and we want to define:

- how multiple level gradual databases can be defined;
- how multiple level gradual patterns can be defined;
- how the support on multiple level gradual patterns can be defined;
- how multiple level gradual patterns can be extracted.

For this purpose, we rely on the existing work for defining two main ways of considering multiple level data. Indeed, data can be aggregated over lines (horizontal aggregation) or columns (vertical aggregation).

Considering the first case, when lines are aggregated, the problem is to define how the values are aggregated over every attribute. It should be noted that lines can be aggregated over one or several attributes, but that it may be the case that some attributes are not aggregated. In this case, all the lines remain in the database but some values change in order to represent data at a higher level of granularity.

Example 6: For instance, data from Example 1 can be merged as shown above, where the 5 lines are merged into 2 lines (Monday and Tuesday). They may also be merged for certain attributes but not for all. For instance, below is the dataset transformed by aggregating over lines for attribute Pineapples.

This kind of operation may be interesting when some attributes are not relevant at a low granularity level while some other ones are. It could thus be compared to a kind of semantic normalization. Such a partial aggregation is noted with parenthesis on the values of aggregated attributes (Pineapples here).

TABLE V
Aggregation Attributes

Id	(Pineapples)	RedApples	Cherries
T1	(6)	3	0
T2	(6)	1	1
T3	(6)	4	2
T4	(9)	1	1
T5	(9)	0	3

Considering the case where columns are merged, the main issue to address is the definition of the hierarchies. In the case of complex hierarchies, it may indeed be the case that some values can be grouped into different super-partitions.

Example 7: For instance, PineApples and RedApples are considered as being WithoutKernel and thus grouped, but we may also consider that RedApples and Cherries can be grouped in order to build the "RedFruit" group.

TABLE VI
Red Fruit Aggregation

Id	NotRed	Red (RedApples + Cherries)
T1	0	3
T2	2	2
T3	4	6
T4	2	2
T5	7	3

V. M2LGP: Definitions

A multiple level attribute MLA is an attribute equipped with a hierarchy. This hierarchy is defined as a set of levels where every level is represented as a partition, all the partitions being embedded.

Definition 7: Level of Granularity. Given a domain D of an attribute. A level of Granularity $L_i(D)$ is defined as a partition $P_i(D)$ of D and a set of labels describing every element of the partition, this set of label being the domain $dom(L_i)$ of the level.

Example 8: We consider the Date attribute with dom(Date) ={01JAN2012−8:00,02JAN2012−11:00, 05AUG2012−07:00, 27SEP2013−09:00, 30SEP2013−05:00, 30SEP2013−08:30}. We may define three levels: L_0 = DateDay, L_1 = DateMonth, L_2 = DateYear, where :

– dom (DateDay) = dom (Date), dom (DateMonth) = {JAN2012, AUG2012, SEP2013}, dom (DateYear) = {2012, 2013}

– Date Month = {{01JAN2012 − 8:00, 02JAN2012 − 11:00}, {05AUG2012 − 07:00, {27SEP2013 −09:00}, 30SEP2013− 05:00, 30SEP2013− 08:30}

Definition 8: Embedded Levels of Granularity. Two levels of granularity L_i and L_j are said to be embedded ($L_i \in L_j$) if L_j defines a partition of L_i.

Definition 9: ML-Attribute. An ML-Attribute MLA is defined by:

-a label,

-a domain dom (MLA),

-a set of embedded levels of granularity $L = L_0(MLA),...,L_g(MLA)$ such as every domain $dom(L_i)$ is ordered with a relation $\leq L_i$.

Example 9: Temporal attributes are a particular case of attributes. Although not being numeric, dates can be aggregated when lines are merged in an adequate manner. For instance, days can be regrouped into months. The attribute Date with domain dom(Date) and levels .given above dom(DateDay) is ordered as: 01JAN2012 − 8 : 00 ≤ 02JAN2012 − 11 : 00 ≤ 05AUG2012−07 : 00 ≤ 27SEP2013 −

09 : 00 ≤ 30SEP2013 − 05 : 00 ≤ 30SEP2013 − 08 : 30. Dom (DateMonth) is ordered as JAN2012 ≤ AUG2012 ≤ SEP 2013. Dom (DateYear) is ordered as 2012 ≤ 2013.

Remark 1: This definition does not allow complex hierarchies as they can be found in data warehouses, although such complex hierarchies could allow to manage the example shown above where there are two incompatible levels of granularity (With/Without kernel and NotRed/Red).

Given an ML-attribute MLA, an aggregation function Agg is defined over the values taken by this attribute that allows to compute the aggregated value when going from one level of granularity up to another one. The attribute is said to be additive if values can be summed up.

Definition 10: ML-DB. A multiple level gradual database is a set of tuples T defined over the schema $S = \{Id,A_1,...,A_n\}$ of n+1 multiple level gradual attributes (ML-Attributes).

Remark 2: Note that the identifier attribute Id is an ML-attribute. This allows to manage vertical aggregation.

A gradual database can be displayed at several levels of granularity. It is said to be additive if all attributes are additive, semi-additive if at least one but not all attributes are additive, and non-additive if none of the attributes is additive.

Definition 11: ML-DB-Disp. A gradual database displayed is given by an ML-DB and a set of levels $\{L_1,...,L_n\}$ associated with every attribute from the schema.

For instance, the information is displayed at different levels of granularity in the previous tables.

Remark 3: It should be noted that based on this definition, the number of items can change when the display is changed.

Definition 12: ML-Item. Given an ML-DB, an ML-Item is a pair (I, variation) where $I \in U_{A \in S}\ U_{LI \in LA}\ dom_A(L_i)$, and variation is ↑ or ↓.

Example 10: For instance, (RedFruit, ↓), (Pineapples, ↑) are ML-Items.

Definition 13: Compatible ML-Item. Two ML-Items are said to be compatible if none of their first part I is the value taken from an upper hierarchy level.

Example 11: For instance, (RedFruit, ↓) and (Pineapples, ↑) are compatible, while (RedFruit, ↓) and (Cherries, ↑) are not.

Definition 14: ML-Pattern. An ML-Pattern is a set of compatible ML-items.

Definition 15: ML-Pattern Inclusion. A pattern P_1 is said to be "ML-included" in another pattern P_2 ($P_1 \in MLP_2$). if for every ML-Item I^k_1 of P_1, there exists an item I^l_2 from P_2 such as either $I^k_1 = I^l_2$ or I^l_2 is an item from an upper level and the variation is the same.

Example 12: For instance P = {(Date, DateDay, ↑), (Fruit, RawFruit, ↓)} is P′ = {(Date,DateDay,↑),(Fruit,ColorFruit,↓)}.

Definition 16: Multiple Level Gradual Support. Given an ML-DB-Disp MLDBD and an ML-Pattern P, the support of P over MLDBD is a function sup from GP to [0,1] that holds the following property: for all $P_1, P_2 \in GP$, $P_1 \in MLP_2 \in supp(P_1) \geq supp(P_2)$.

On the basis of the two support definitions, the main point to observe is that they are strongly related to the number of lines from the database.

When performing a horizontal aggregation (merging columns), the number of lines does not change, although it is changed when lines are merged. However, the gradual pattern being considered has changed (as the columns changed).

When performing a vertical aggregation (merging lines), the number of lines may change, either for one or for all the attributes.

Whatever the aggregation, the support can still be computed using either the longest path or the rank correlation.

VI. M2LGP: Efficient Mining

Although aggregating the columns and/or lines may be seen as a reduction, it is not the case that it simplifies the complexity of the algorithm. Indeed, the reduction is true only at one level of granularity while all levels may be considered.

In order to address the problem of navigating through this huge search space, we define the following strategies and discuss their pros and cons.

A. Building the Search Space: MLGP Operators and MLGP Lattice

The navigation through candidate patterns is performed via operators that allow to apply horizontal and vertical (dis)aggregation. These operators are connected to the OLAP roll-up and drill-down operators, although not being exactly identical.

B. Navigating through the Search Space: Defining Strategies

Considering these operations and the resulting lattice representing the search space, we consider several strategies.

Given an ML-DB-Disp D, we can aggregate or disaggregate this database over lines or columns.

Definition 17: ML-HAgg. An H-Aggregation consists in merging several lines with respect with a line hierarchy (given a source level and a target level), regarding one or several attributes. The value taken on these attributes are aggregated. Such an aggregation is denoted by ()

Example 13: For instance, lines of Ticket1 to Ticket3 one the one hand, and of Ticket4 and Ticket5 on the other hand have been merged over the Pineapples attribute, which is then denoted by (Pineapples), the values being themselves enclosed within parenthesis.

Remark 4: Although having been merged, the lines may remain separated if some attributes are not concerned with the ML-HAgg operation, which is a major difference with OLAP roll-up. The aggregated value is repeated on all the lines, which is not problematic as we consider equal values as being comparable in both ways (≤).

Example 14: In our example, the lines are merged over the Pineapples attribute but not over the other attributes.

Definition 18: ML-HDisagg. An ML-HDisagg with respect with a hierarchy (given a source level and a target level),

regarding one or several attributes, consists in drilling down by generating several lines from one line and by recomposing (from the raw data) the unmerged values. The value taken on these attributes are aggregated.

Definition 19: ML-VAgg. A vertical aggregation consists in merging several columns with respect with a hierarchy (given a source low level and a target high level).

Definition 20: ML-VDisagg. A vertical disaggregation consists in splitting one column into several columns with respect with a hierarchy (given a source high level and a target low level).

Definition 21: Most Specific ML-Pattern. Given a minimum support threshold minsup, an ML-pattern P_1 is said to be the "most specific" pattern if there does not exist any P_2 such as $P_2 \in_{ML} P_1$ and $supp(P_2) \geq minsup$.

Taking these operations into account, the lattice representing the search space is based on the classical lattice of all the parts of the n items and variations, which is enriched by levels.

The first strategy is to explore all the search space. Instead of having a lattice composed by 2^n nodes in the case of n attributes, the search space is then composed of 2^l where l is the sum of all numbers of levels. For instance, we may consider the attribute Fruit with 2 levels (raw level, and Color Level). Moreover, a strategy should be followed in order to navigate through the levels. However, there is no useful monotonicity between levels, which prevents from pruning the space in an efficient way. It can thus be the case that exploring this search space is not possible. This is the reason why other methods can be considered.

Convergent and divergent patterns have been proposed in the sequential pattern mining framework [19], where items are naturally organized as they are ordered over time. In the context of multiple level gradual patterns, it is unfortunately not the case that we can go up or down while going through attributes.

The last proposal is thus to use a heuristic. In our framework, we propose to compute all the supports for all multiple level gradual patterns of size 2 (containing 2 ML-items) and to consider the levels that maximize the support at this step. Note that this also holds for the horizontal aggregation in order to decide to which extend lines must be merged.

Example 15: For instance, if considering the dataset structured over the schema Date, Fruits, all the possibilities of levels and crossing will be considered: (DateDay, RawFruits),(DateMonth, RawFruits), (DateYear, RawFruits), (DateDay, ColorFruits), DateMonth, ColorFruits) (DateYear, ColorFruits) and the most successful regarding the support will be chosen.

Note that pairs of such ML-items may be incompatible. For instance, it may be the case that Fruits is considered at the RawFruits level when being combined with Date but considered at the ColorFruits level when combined with another attribute. In this case, we propose to choose between two strategies: keep all levels or choose the one optimizing the support.

VII. M2LGP: Algorithms and Experiments

Our algorithms are based on the decomposition of the database in order to deal with all the hierarchies, as described in the algorithms below. We then run experiments using existing implementations of gradual pattern mining algorithms.

ALGORITHM I
ALGOM2LFGP

Input : DB
Output : Set of G^D

$G^D \leftarrow \emptyset$;
Transform DB (DB);
for each *transformed database D* ***do***
  AlgoGP (D);
end

Experiments have been run on both synthetic and real data reporting expressions of gene in the case of cancer. As can be seen on Fig. 2, the presence of the hierarchies makes the problem difficult to tackle as it exponentially increases the search space. We thus aim at using parallel programming in further work.

ALGORITHM 2
TRANSFORM DB

Input : DB
Output : Set of tranformed database(D)

D← ∅;
GenerateHierarchy(nbColsDB,nbLignesDB);
for each $h \oslash H_H$ ***do***
D ← D ∅ Htransform(DB, h);
end
D′ ← D;
freach $(h,h') \oslash H_{H2}$ ***do***
D′ ← D′ ∅ HHtransform(DB, h, h′);
end
D″ ← ∅;
for each $h \oslash H_V$ ***do***
D″ ← D″ ∅ Vtransform(DB, h);
end
D‴ ← D″;
for each $h \oslash D''$ ***do***
D‴ ← D‴ ∅ Htransform(DB, h);
end
return set of D(D, D′, D″, D‴);

ALGORITHM 3
GENERATE HIERARCHY

Input : nbColsDB,nbLignesDB
Output : Hierarchy(H)

$H_H \leftarrow \emptyset$;
$H_{H2} \leftarrow \emptyset$;
$V_H \leftarrow \emptyset$;
$H \leftarrow H_H, H_{H2}, V_H$;
for each $h \oslash H$ ***do***
  if $h=H_H$ ***then***
    h ← GenerateNodeEdges(nbColsDB,nbTopNode);
  else if $h = H_{H2}$ ***then***
    h← GenerateNodeEdges(nbLignesDB,nbTopNode);
  else
    h ← GenerateNodeEdges();
  end
 end
return H;

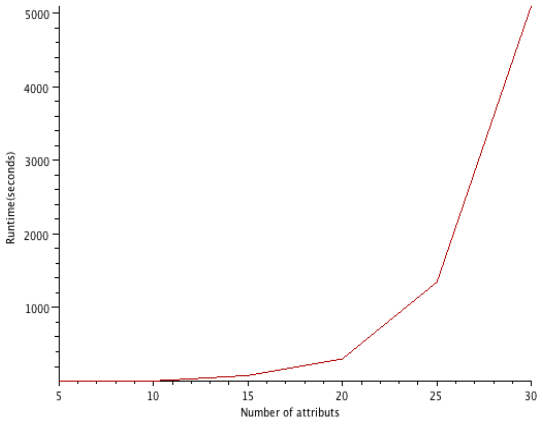

Fig. 2 Runtime regarding the number of attributes

VIII. Conclusion and Future Works

In this paper, we propose the original method M2LGP for mining multiple level gradual patterns. Such patterns allow to take into account the natural structures of the information that can often be considered at different levels of granularity. The paper proposes a formal framework for redefining all the concepts in this particular framework, including a discussion on the computation of the support. The topic is of great importance since data mining has been pointed as a key challenge, especially when dealing with big data. The problematic is not easy, since multiple level gradual patterns are similar to existing frameworks (gradual patterns, multiple level association rules, OLAP Mining) but are still an original approach as they contain specificities that are difficult to manage.

Future work include the study of fuzzy partitions for defining fuzzy multiple level gradual patterns and their support. We will also further investigate the algorithms for extracting the multiple level gradual patterns. The efficient implementation of this proposal will be based on the existing parallel softwares such as [12].

References

[1] Ayouni S., Ben Yahia S., Laurent A., Poncelet P. Fuzzy Gradual Patterns: What Fuzzy Modality For What Result?. In Proc. of the International Conference on Soft Computing and Pattern Recognition (SoCPaR'10). 2010.

[2] Ayouni S., Ben Yahia S., Laurent A., Poncelet P. Genetic Programming for Optimizing Fuzzy Gradual Pattern Discovery.. In Proc. of the Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-2011). 2011.

[3] Berzal, F., Cubero, J.C., Sanchez, D., Vila, M.A., Serrano, J.M.: An alternative approach to discover gradual dependencies. Int. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS). 15(5) (2007) 559–570.

[4] Bodenhofer, U.: Orderings of Fuzzy Sets Based on Fuzzy Orderings Part I: The Basic Approach. In Mathware & Soft Computing 15 (2008) 201-218.

[5] Bodenhofer, U.: Orderings of Fuzzy Sets Based on Fuzzy Orderings Part II: Generalizations. In Mathware & Soft Computing 15 (2008) 219-249.

[6] Bodenhofer, U., and Klawonn, F.: Towards Robust Rank Correlation Measures for Numerical Observations on the Basis of Fuzzy Orderings. In 5th Conference of the European Society for Fuzzy Logic and Technology, septembre, 2007, pp. 321 – 327.

[7] Chaudhuri S. and Dayal U. An overview of data warehousing and OLAP technology. ACM-SIGMOD Records, 26(1):65–74, 1997.

[8] Codd E.F., Codd S.B., and Salley C.T. Providing OLAP (OnLine Analytical Processing) to user-analysts: An it mandate. In White Paper, 1993.

[9] Di Jorio L., Laurent A., Teisseire M. Mining Frequent Gradual Itemsets From Large Databases. Intelligent Data Analysis (IDA09). 2009.

[10] DiJorio L.,Laurent A.,Teisseire M.Gradual Rules: AHeuristic Based Method and Application to Outlier Extraction. International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM). Vol.1, pp.145-154. 2009.

[11] Dubois, D., Prade, H.: Gradual inference rules in approximate reasoning. Information Sciences 61(1-2) (1992) 103–122.

[12] Do T.D.T., Laurent, A., and Termier, A. PGLMC: Efficient Parallel Mining of Closed Frequent Gradual Itemsets. In Proc. International Conference on Data Mining (ICDM). 2010.

[13] Han J. and Fu Y. Mining multiple-level association rules in large databases. IEEE Trans. Knowl. Data Eng., 11(5):798–804, 1999.

[14] Kendall M. and Babington Smith B., The problem of m rankings, The annals of mathematical statistics, 1939, vol.10(3):275- 287.

[15] Koh, H-W., and Hullermeier, E.:Mining Gradual Dependencies Based on Fuzzy Rank Correlation. In Combining Soft Computing and Statistical Methods in Data Analysis. Springer, 2010.

[16] Laurent, A., Lesot, M.-J., and Rifqi, M.: GRAANK: Exploiting Rank Correlations for Extracting Gradual Itemsets. In Proc. of the Eighth International Conference on Flexible Query Answering Systems (FQAS09). 2009.

[17] Quintero M., Del Razo F., Laurent A., Poncelet P. and Sicard N. Fuzzy Orderings for Fuzzy Gradual Dependencies: Efficient Storage of Concordance Degrees. In Proc. of the IEEE International Conference on Fuzzy Systems. 2012.

[18] Quintero M., Laurent A. and Poncelet P. Fuzzy Orderings for Fuzzy Gradual Patterns. In Proc. of the Int. Conference on Flexible Query Answering Systems. 2011.

[19] Plantevit M.,Choong Y. W.,Laurent A.,Laurent D.,Teisseire M. Mining Multidimensional and Multiple-Level Sequential Patterns. ACM Transactions on Knowledge Discovery from Data (ACM TKDD). 2009.

[20] Srikant R. and Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In EDBT, pages 3–17, 1996.