BOHR International Journal of Data Mining and Big Data
2021, Vol. 1, No. 1, pp. 10–25

https://doi.org/10.54646/bijdmbd.003
www.bohrpub.com

An Efficient Hybrid by Partitioning Approach for Extracting Maximal Gradual Patterns in Large Databases (MPSGrite)

Tabueu Fotso Laurent Cabrel

Department of Computer Engineering, UIT-FV, University of Dschang, Cameroon;
Department of Mathematics and Computer Science, FS, University of Dschang, Cameroon
E-mail: laurent.tabueu@gmail.com
Preprint submitted to BOHR International Journal of Data Mining and Big Data January 5, 2022

Abstract. Since automatic knowledge extraction must be performed in large volume databases, empirical studies already show that at the level of generalized patterns, association rules and frequent graded patterns, there is an exponential increase in the search space and in the number of relevant patterns extracted. Faced with this problem, many approaches have been proposed, with the aim of reducing the size of the search space and the waiting time in order to offer users a sufficiently small number of relevant patterns to make decisions or refine their analyses in a reasonable and realistic time. Incremental frequency extraction algorithms in large databases are CPU intensive. This paper presents a new technique to improve the performance of maximal frequent gradual pattern extraction algorithms. The exploitation of this technique leads to a new and more efficient hybrid algorithm called MSPGrite. Experiments performed confirm the interest of the proposed approach.

Keywords: Pattern mining, pruning search space, maximal gradual support, lattice, adjacency matrix, partitioning.

1 Introduction

Data mining is part of a process known as knowledge extraction (KDE), which appeared in the scientific community in the 90s. It is a fast-growing research field aiming at exploiting the large quantities of data collected every day in various fields of computer science. This multidisciplinary field is at the crossroads of different domains, such as statistics, databases, big data, algorithms, artificial intelligence, etc. The type of data mining algorithm varies according to the type of data (binary, categorical, numerical, time series, spatial, etc.) of the dataset on which the algorithms will be applied, or the type of relationship between the patterns searched (sequence, co-variation, co-occurrence,...) as well as the level of complexity and semantics of the analyzed data [1]. It is generally about finding co-occurrences or dependencies between attributes or items and relationships between objects or transactions in the dataset. Maximum Graduated Pattern Mining, which is the focus of this paper is generally used to find relationships between attributes, while clustering is used to find relationships between objects [2]. Since automatic knowledge extraction has to be performed in large volume databases, empirical studies already show that at the level of generalized patterns, association rules and frequent gradual patterns, one has to exponentially increase the size of the search space to be explored in order to extract useful knowledge. Faced with this problem, a large number of approaches have been proposed, with the aim of reducing the size of the search space, the waiting time to offer users a number of relevant patterns reduced enough to make decisions or refine their analyses in a reasonable and realistic time. Thus was born the technique of mining closed gradual patterns [3]. The goal is to extract a condensed representation of fuzzy gradual patterns based on the notion of closure of the Galois correspondence. It is used as a generator of rules and gradual patterns. Other approaches are based on the use of parallel algorithms and multi-core architectures that minimize the extraction time compared to their sequential versions. We can cite the work of Negreverge with his Paraminer algorithm [4], pglcm by Alexandre Termier. However, very little work is directed towards the extraction of frequent and maximal

gradual patterns, hence our orientation in this article on the extraction of frequent and maximal gradual patterns using a hybrid approach based on the SGrite algorithm and the partitioning of the dataset.

1.1 Objectifs

General objective: Extract maximum gradual patterns from large database

Specific objective 1: Our approach is based on the one hand on a reduction of half in a first step of the research space; This using as a search space, the lattice has at least one positive term. Two simultaneous traverses of said lattice are performed: an ascendant constructs the candidate sets of size 1 to a size k, $k * n$, with n the number of items. During the browse, sgrite join is used, and the other descendant to manage the maximum gradual candidates and frequencies. This first objective is the exploitation of the lattice with at least one positive term as well as a two-way lag with a view to further reducing the operations of calculation of the support, and thus the search space.

Specific objective 2: Guarantee extraction in large databases

The observation of the memory occupation during the extraction of gradual patterns by the methods, Grite, SGrite, and Graank often requires preprocessing to reduce the size of the database [5] in the case of very correlate or dense data. This adaptation is necessary to carry out the extraction. However, this search remains partial and can lead to the loss of quality patterns, indeed certain co-variations between the attributes considered in the original dataset and the rest of the dataset ignored remains unvalued. To partially solve this problem, we propose to process a search by partitioning the dataset, which will be described in Section 4.

2 Literature Review

2.1 Definitions

Definition 1. Gradual item [1, 6, 7, 8, 5]: A gradual item is of the form A^*, where A is an attribute and $* \in \{\leq, \geq, <, >\}$ and expresses a variation on the values of attribute A. If the comparison operator * is equal to $\geq$ (resp. $\leq$), A^* translates an increasing (resp. decreasing) variation of the values of A.

Example 1: $S^{\geq}$, $S^{\leq}$ and $V^{\geq}$ are examples of gradual items taken from the Table 1. They can be interpreted respectively as "the more salary increases", "the more salary decreases" and "the more location of the vehicle increases".

Definition 2. Gradual itemset [9, 10, 6, 7, 8, 5]: A gradual itemset, denoted $\{(A_i, *i), i = 1 \dots k\}$ or $\{A_i^{*i}, i = 1 \dots k\}$, is a concatenation of several gradual items interpreted semantically as a conjunction of gradual items. It expresses a co-variation of the values of its attributes. The size of a gradual itemset is the number of its attributes.

For example, the gradual itemset $A^{\geq}S^{\geq}$, taken from Table 1, translates the covariation "*the more the age increases, the more the salary increases*".

Definition 3. Complementary gradual pattern [9, 1, 5]: The complement of the gradual pattern $M = \{(A_i^{*i}), i = 1 \dots k\}$ is the gradual pattern $c(M) = \{ (A_i^{c(*i)}), i = 1 \dots k\}$, where c(*i) is the complement of the comparison operator *i.

In the literature [1], $c(\leq) =\geq$, $c(\geq) = \leq$, $c(<) =>$, $c(>) =<$.

For example, given that we have a dataset with three attributes as shown in Table 1, A > (resp. A > S >) have for complementary gradual pattern A < (resp. A < S <).

Definition 4. Inclusion of gradual patterns: Gradual pattern A is included in pattern B, denoted by $A \subseteq B$, if every gradual item of A occurs in B.

The following two properties are true for each of the three definitions of the notion of gradual support presented in Section 2.2. They allow to significantly prune the search space. Gradual support equality property [1, 11]: The support of a gradual pattern is equal to the gradual support of its complement. Gradual support anti-monotonicity property [1, 11, 10]: If the gradual pattern A is included in the gradual pattern B then $SG(A) \geq SG(B)$. The Definition 4 can be illustrated by the example of the digital database presented in Table 1 which describes $n = 8$ people according to $m = 3$ attributes.

For example, the gradual pattern $A^>$ $S^>$ is included in the gradual patterns $A^>S^>V^>$ and $A^>$ $S^>$ $V^<$. The gradual item $A^<$ is included in $A^<S^<$, $A^<S^>$ and $A^<S^>V^<$.

Definition 5. Lattice of gradual patterns: A lattice of gradual patterns is a lattice induced by the set of gradual patterns provided with the inclusion relation. The set of nodes of the lattice is the set of gradual patterns. An arc which goes from a pattern A to a pattern B reflects the inclusion of A in B.

Definition 6. Lattice of gradual patterns having at least one positive term: The trellis of the gradual patterns having at least one positive term is the sub-trellis of the trellis of the gradual patterns which contains only the gradual patterns having at least one gradual item which reflects an increasing *i variation. Such a pattern is of the form $A_1^{\geq}\{A_i^{*i}\}, i = 2 \dots k$.

The lattice of gradual patterns represents the search space for frequent gradual patterns. Lemma 2.1 can be used to reduce the search space by half, for example to the lattice of gradual patterns having at least one positive term. The concept of lattice is illustrated in Figures 1 and 2.

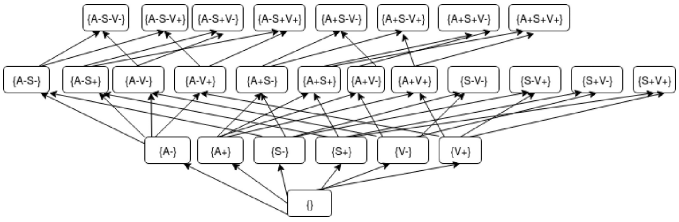


Figure 1. Lattice of gradual patterns based on the attributes of Table 1 [5, 11].

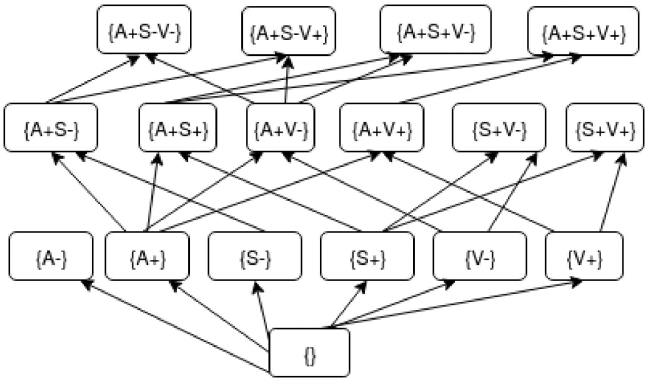


Figure 2. Lattice of gradual patterns having at least one positive term obtained from the attributes of Table 1 [5].

2.2 Gradual Pattern Extraction Techniques

The linear regression approach proposed by Hüllermeier [12] makes it possible to extract gradual rules whose support and confidence are greater than the threshold set by the user. This approach only considers fuzzy data and rules whose premise and conclusion are smaller than or equal to two. However, the concept of T-Norm which is part of this approach allows to overcome the limit of the size of the premise and the conclusion of the rules.

In the approach Berzal et al. [13], the weight of a gradual pattern, also called Gradual Support (SG) is equal to the number of couples of distinct objects which verify the order induced by the pattern divided by the P total number of couples of distinct objects in the database. Thus, SG(M) $= \frac{\sum_{o,o' \in \mathcal{D}x\mathcal{D}} |o \preceq_M o'|}{|\mathcal{D}|(|\mathcal{D}|-1)}$, where M is a gradual pattern and D is the database. The approach of Laurent et al. [10]. is an improvement of the approach of Berzal et al. [13] which exploits the fact that if a couple of distinct objects (o, o') verifies the order induced by a gradual pattern then its complementary (o', o) does not verify it. In this approach, we have: SG (M) $= \frac{\sum_{o,o' \in \mathcal{D}x\mathcal{D}} |o \preceq_M o'|}{(|\mathcal{D}|(|\mathcal{D}|-1))/2}$. In the so-called maximum paths approach [11, 9, 14, 15], the gradual support of a gradual pattern M is equal to the length of a maximal path associated to M divided by the total number of objects in the database. In this approach we have: SG(M) $= \frac{\max_{\mathcal{D} \in \mathcal{L}(M)} |D|}{|\mathcal{D}|}$. The SGrite [16, 5] approach is based on Grite. It is an algorithm for discovering gradual patterns on the basis of maximum paths. It uses the anti-monotonicity and complementary patterns properties of the support to prune the search space. The lattice with first positive term used reduce to half the search space. Another particularity of SGrite is that requires one sweep of the dependence graph while Grite requires two sweeps for calculation of gradual support. She uses two classes of gradual support computation algorithms which perform a single sweep of the precedence graph.

2.3 Presentation of SGrite Algorithm

In SGrite algorithm, the two fundamental operations are the generation of candidates and the computation of the support. The most requested and most CPU intensive operation is the support calculation since it is performed for each candidate. The SGrite algorithm is based on a number of concepts defined below. In the following definitions, O is the set of objects, o and o' are objects.

Definition 7. Adjacency matrix: The adjacency matrix of a gradual pattern M is a binary matrix which associates with each pair of objects (o, o') the value 1 if the pair of objects respects the order induced by the pattern M and 0 otherwise.

The adjacency matrix of a pattern induces a dependence graph whose nodes are objects and non-zero inputs of the adjacency matrix represent the dependences between couples of nodes.

Definition 8. Father node and son node: Given the adjacency matrix Adj_M of a pattern M, if $\text{Adj}_M[o, o'] = 1$ then o is the father of o' and o' is a child of o.

Definition 9. Isolated node: It is a node which is not tied to another node, i.e. it has neither father nor son. Given the adjacency matrix Adj_M of a pattern M, the set of isolated nodes is: $\{o \in O | \forall o' \in O, Adj_M[o, o'] = 0 \wedge Adj_M[o', o] = 0\}$.

Definition 10. Root: It is a node that does not have a father, but is linked to all the other nodes. Given the adjacency matrix Adj_M of a pattern M, the set of root nodes is: $\{o \in O | \forall o' \in O, Adj_M[o, o'] = 1 \wedge Adj_M[o', o] = 0\}$.

Definition 11. Leaf: It is a node having no son, but which is not isolated. Given the adjacency matrix Adj_M of a pattern M. the set of leaves is: $\{o \in O | \forall o' \in O, Adj_M[o, o'] = 0, \exists o'' \in O | Adj_M[o'', o] = 1\}$.

The SGrite algorithm takes as input an adjacency matrix, an object node $\in$ O and a vector, called Memory, of size $|O|$ whose indexes are the objects of O. We assume that Memory[o] $= -1 \forall o \in O$ before the execution of each algorithm. During the execution of each algorithm, Memory[o] contains the current maximum distance between o and any leaf. At the end of the execution of each algorithm Memory[o] contains the final maximum distance between o and

any leaf. The first class of algorithms is to update the values of the parents of a node as soon as the value of this node is updated. The second class of algorithms is to use only the final value of a node to update the values of the nodes of his parents. In Sgrite approach, we have four versions of the so-called SGrite [5] algorithm, namely SGOpt, SGB1, SGB2 and SG1, depending on the technique of improving the computation of the gradual supports. The SGOpt, SG1 and SGB1 method produce the best result on various datasets[?].

3 Methodology

To achieve our goals, several properties must be considered: (P1) anti-monotony support, (P2) complementary gradual patterns, (P3) use of the frequency of sub-patterns of a frequent gradual maximum for pruning. The role of P1 and P2 is already known and shown in sgrite for the upward traversal for the generation of frequent gradual patterns. P3 makes it possible in the downward traversal to ignore frequent gradual sub-patterns of a maximal gradual pattern that I have to determine in advance. At the same time during the extraction process, in the downward path of the lattice with at least one positive term, we construct the maximal progressive candidates which belong to the lattice has at least one positive term. Hence the guarantee of the conservation of the properties of the optimal search space. In addition, by browsing down, we can reduce the search space further. By a chain filtering. Thus, we use to prune on the one hand the set of infrequent gradual sub-patterns which is applied to the maximal gradual candidates, and on the other hand the sub-patterns of a frequent and maximal gradual pattern discovered beforehand starting from a set of the candidate gradual maxima of larger size during the procedure, and finally, the last pruning is done by calculating the gradual support to check the frequency of the candidates and their relevance. To complete the extraction process, there are two stop conditions. Either the current set of candidates is exhausted during the upward traverse, or the maximum number of candidates is exhausted first and the search is also terminated.

3.1 Hypotheses

1. Global time reduction of the support computation (the one brought by sgrite, omission of the computation of the gradual support of the frequent sub-patterns and the reduction of the depths of the dependency graphs associated to the maximal candidate pattern), following the computation of the fusion of the n gradual items composing the candidate in question.
2. Reduction of the search space.
3. The reduction in the number of gradual knowledge produced, put with the possibility of listing them exhaustively.
4. The choice of the size of the different partitions. To begin we take 2 partitions and we take the partition 1 the biggest possible extractable by sgrite, then the partition 2 it is a supplement of items of the considered data set.

4 Presentation of the Hybrid Extraction Method for Maximal Gradual Patterns

In this section we will explain the general operating principles of the MPSGrite method. Section 4.1, explain how the partitioning method works. Section 4.2 presents the operation of the algorithm for finding maximum frequent gradual patterns. The notations used in this part are summarized in the Table 4. It is important to specify that what motivates this algorithm is its relevance. Indeed, the algorithm is oriented towards a different concept from that of "SGrite" and its extensions; Partition will have the particularity of being much more efficient on very large databases, like the current OLTP[1] systems, hence the importance of his study.

4.1 Partition Working Principle

To simplify the description we will limit to 2 partitions and consider a dataset D, which has n items. D is partitioned into two data sets D_1 of size n_1 and D_2 of size n_2, such that $n = n_1 + n_2$. D_1 represents partition 1 of the database containing the n_1 first items, while 2 is the second partition containing the n_2 last items of D.

Definition 12. partition of a database: It is a part of D which has the same number of transactions as D and which takes a contiguous subset of the items from the dataset D representing the items taken into account in the score. Denote by D and $I = \{i_k\} k = 1 \dots n$ the items, $n_k <$ n, if D_k the k-th partition which starts with item number $l / 1 \leq l <$ n we have $D_k = (O, I)$ with $I \subseteq I$ and $I = \{i_k\}_{k=l \dots l+n_k-1}$.

Definition 13. Independence of two partitions of a database: Let $D_1 = (O, I_1)$, $D_2 = (O, I_2)$ two partitions of a dataset D = (O, I) / I 1 $\subset$ I, I 2 $\subset$ I. They are independent iff $I_1 \cap I_2 = \emptyset$.

Example 14. Illustration of partition: Let I = {A, S, V} be the set of attributes of the salary data set, see Table 1; where A is the age attribute, S is the salary and V is the vehicle location number attribute.

For example D_1 = (O,{A, S}) and D_2 = (O,{V}) are two independent partitions of dataset of Table 1.

[1] OnLine Transactional Processing.

Table 1. Salary data set D.

id	Age(A)	Salary(S)	Vehicle(V)
o1	20	1200	1
o2	28	1850	1
o3	24	1200	0
o4	35	2200	1
o5	30	2000	1
o6	40	3400	1
o7	52	3400	2
o8	41	5000	2

Table 2. Partition D_1 of D.

id	Age(A)	Salary(S)
o1	20	1200
o2	28	1850
o3	24	1200
o4	35	2200
o5	30	2000
o6	40	3400
o7	52	3400
o8	41	5000

Table 3. Partition D_2 of D.

id	Vehicle(V)
o1	1
o2	1
o3	0
o4	1
o5	1
o6	1
o7	2
o8	2

Order defined on the sets.

Definition 15. Order on items: The order relation on the set of items of a data set D, denoted $<_I$ denotes the natural order relation of appearance of items in D.

For example in the data set of the Table 1 we will have as order between the items A $<_I$ S $<_I$ V.

Definition 16. ordered gradual itemset: An ordered gradual itemset is a gradual itemset which respects on the set of items which constitutes it the order defined by its position in the set of items for the data set D = (O, I) considered. Denote by $M = \{A_i^{*i}\}_{i\in\{1,2,\ldots,n\}}$ a gradual k-pattern, it is ordered iff ∀j, $1 \leq j < k$, with j which represents the index of appearance of the item in the pattern M, we have $A_j <_I A_{j+1}$.

Definition 17. Gradual positive ordered itemset: A positive ordered gradual itemset is an ordered gradual itemset so at least the first term has increasing variation.

For example, the gradual itemsets $A^>$ $S^<$, $A^<$ $S^<$, $A^>$ $V^<$ and $A^<$ $S^<$ $V^<$ are ordered gradual itemsets. On the other hand $S^<$ $A^>$ is not an ordered gradual itemset, even if it has the same meaning and the same gradual support as $A^>S^<$ ordered. itemset, even if it has the same meaning and the same gradual support as $A^>$ $S^<$ ordered.

Proposition 18. Total order on two gradual items: When doing a pairwise comparison of gradual items, for example A_1^{*1} and A_2^{*2}, if $A_1 <_I A_2$ then $A_1^{*1} <_I^m A_2^{*2}$ and vice versa, but in the case of A_1 is equal to A_2 the order is induced by the variation associated with each gradual item as follows: if *1 = *2 then $A_1^{*1} <_I^m A_2^{*2}$, if on the other hand *1 = < and * 2 = > then $A_1^{*1} <_I^m A_2^{*2}$, so if *1 = > and * 2 = < then $A_2^{*2} <_I^m A_1^{*1}$.

For example in the Table 1 $S^< <_I^m S^>$.

Definition 19. Order on two gradual ordered pattern: Let $M_1 = \{A1_i^{*i}\}_{i=1\ldots k1}$, $M_2 = \{A2_i^{*i}\}_{i=1\ldots k2}$ such that $k_1 = k_2$ two ordered gradual itemsets see Def. 16. They are ordered iff $\forall k = 1 \ldots min(k_1, k_2)$, $A1_k <_I A2_k$ or $A1_k^{*k} <^m A2_k^{*k}$ see Prop. 18. Note this order relation between ordered gradual patterns $<_I^m$ we then have $M_1 <_I^m M_2$.

For example, for the gradual itemsets $A^>$ $S^<$, $A^>$ $V^<$ respects the following $A^>$ $S^< <_I^m A^>V^<$, which means that the gradual pattern $A^>$ $S^<$ is less than $A^>V^<$ following this order relation.

Definition 20. Set of ordered gradual patterns: It is a set of gradual patterns ordered by the relation $<_I^m$. Denote by $L_{MgO} = \{L_{mgo}^i\}_{i=1\ldots n}$ the set of sets of ordered gradual patterns $\forall L_{mgo}^i \in L_{MgO}$ such that $L_{mgo}^i = \{m_j\}_{j=1\ldots k}$ we have then $\forall m_j, m_{j+1} \in L_{mgo}^i$, $m_j <_I^m m_{j+1}$, with $j = 1 \ldots k-1$. Each L_{mgo}^i is a set of ordered gradual patterns.

4.1.1 Principe

The search by partitioning is based on the principle of SGrite, that is to say one of these optimal variants SGOpt or SG1 [5] on each of the independent partitions considered. Once all the sets of frequent patterns of each of the partitions have been determined and organized by level, according to the sizes of patterns, we must initially merge the gradual patterns of the same level of each partition. The next step is to generate the missing potential candidates by the method described below. this process of determining the frequencies of whole database goes from frequent item set of size 1 to the maximum possible size. The steps below always take place in the search space for the lattice at the first positive term. We can summarize the approach in 4 steps:

1. Step 1: determination of the frequent and infrequent gradual patterns of each partition;
2. Merging, level by level, of the gradual itemsets of all the partitions, on the one hand the frequent ones, on the other hand the infrequent ones;
3. Iterative and pairwise generation of candidate patterns, based on the gradual patterns determined in

step 2 as follows:

(a) Choose two levels to start the generation of gradual candidates of size k + 1 from the frequents of size k previously known. At the start level and nextLevel are respectively 1 and 2;
(b) For each candidate c k = $\{A_1^{*i}{}_{i=1...k}\}$, frequent, of the current ordered set of the gradual patterns considered 'er ś of size k, extract its prefix, $\text{Prefix}_{k-1} = \{A_1^{*i}{}_{i=1...k-1}\}$
(c) From the obtained prefix, construct its adjacency matrix, and find the bounds in the set of ordered frequent and infrequent gradual patterns of size k having as prefix Prefix_{k-1};
(d) Then retrieve the value of maximum attribute, noted max. It is the gradual item in position k, resulting from the two sets of size k (frequent and infrequent) which has the greatest value;
(e) Generate the suffixes that are used for the merge with Prefix_{k-1}, and noted IGc = $\{(\text{max} + \text{i})^<, (\text{max} + \text{i})^>\}_{\text{i}=1...(|\text{I}|-\text{max}+1)}$ of gradual items.
(f) determine support for new candidates c new = $\text{Prefix}_{k-1} \cup \text{e}, \forall \text{e} \in \text{IGc}$. The frequents are added to the frequent gradual k-patterns of level k and the infrequent to the infrequent gradual k-motfs.
(g) repeat the process 3-(a) to 3-(f) until the set of frequent ordered graduals of size k are exhausted.

4. Repeat steps 2) and 3) until the current candidate set is empty.

Illustration of Partition

Example 21. Step 1: set of frequent gradual patterns by partition

frequent gradual patterns of partition 1, Table 2	frequent gradual patterns of partition 2, Table 3
level 2	**level 2**
$A^>S^<$ SG: 25.0%; $A^>S^<$ SG: 75.0%	
level 1	**level 1**
$A^<$ SG: 100.0%; $A^>$ SG: 100.0%; $S^<$ SG: 75.0%; $S^>$ SG: 75.0%	$V^<$ SG: 37.5%; $V^>$ SG: 37.5%

Example 22. Step 2: merge the gradual patterns from partitions 1 and 2 (see Tables 2, 3)

frequent gradual patterns of initial fusion
level 2
$A^>S^<$ SG: 25.0%; $A^>S^<$ SG: 75.0%
level 1
$A^<$ SG: 100%; $A^>$ SG: 100%; $S^<$ SG: 75%; $S^>$ SG: 75%; $V^<$ SG: 37.5%; $V^>$ SG: 37.5%

Example 23. Final step: Set of frequent gradual patterns of all partitions

Frequent gradual patterns
level 3
$A^>S^>V^>$ SG: 37.5.0%
level 2
$A^<S^<$ SG: 25.0%; $A^>S^>$ SG: 75.0%; $A^>V^>$ SG: 25.0%; $A^>V^>$ SG: 37.5%; $S^>V^>$ SG: 37.5%
level 1
$A^<$ SG: 100%; $A^>$ SG: 100%; $S^<$ SG: 75%; $S^>$ SG: 75%; $V^<$ SG: 37.5%; $V^>$ SG: 37.5%

In this phase we can see that in the gradual 2-patterns, those in red color are generated by new patterns formed from items of the 2 partitions.

Example 24. Set of Frequent Gradual Patterns of Sgrite

Frequent gradual patterns of Sgrite
level 3
$A^>S^>V^>$ SG: 37.5.0%
level 2
$A^>S^<$ SG: 25.0%; $A^>S^>$ SG: 75.0%; $A^>V^<$ SG: 25.0%; $A^>V^>$ SG: 37.5%; $S^>V^>$ SG: 37.5%
level 1
$A^<$ SG: 100%; $A^>$ SG: 100%; $S^<$ SG: 75%; $S^>$ SG: 75%; $V^<$ SG: 37.5%; $V^>$ SG: 37.5%

Table 4. Notations used in the Partition algorithm.

n	number of partitions in data set D.
$(\text{n}_1, \text{n}_2, \ldots, \text{n}_\text{n})$	array of size n containing the number of items in each partition; n k is the number of items in the kth partition
D_r	1. rth partition of the dataset.
C_k^G	Set of global candidate gradual k-itemsets (potential frequent gradual itemsets).
$mapC^G$	Set of global candidate gradual itemsets (potential frequent itemsets)
$mapF_r$	Set of frequent gradual itemsets in partition D_r
$mapF_k^r$	Sets of gradual k-itemsets ordered (see Def 16) in the partition D_r
$mapF^G$	Set of global frequent itemsets (frequent itemsets).
IF_k^G	Set of global infrequent gradual k-itemsets, i.e. for all partitions.
$mapIF_r$	Set of infrequent gradual itemsets in the partition D_r.
$mapIF_k^r$	Sets of ordered infrequent gradual k-itemsets(see Def 16) in the partition D_r.
$mapIF^G$	Set of global infrequent itemsets, i.e. for all partitions (infrequent itemsets)

In the above notations, the sets $mapC^G$, IF_k^G all have the two fields: gradual pattern and gradual support (SG), for each of the elements belonging to these sets. The sets

$mapF_r$, $mapF^G$, $mapC^G$, $mapIF_r$ and $mapIF^G$ are maps or association tables where the keys are sizes of gradual patterns of each level and the values are a set of k-gradual patterns ordered by the level k considered (i.e. of key k).

Remark: all sets of gradual itemsets contains the positive ordered gradual itemsets.

- The **Partition-Gen-Sgrite** (D_r, minsupport) algorithm uses the Sgrite principle on D_r and returns the set $mapF_r$ local gradual frequent patterns, i.e. gradual patterns ordered according to the definition 20 which are frequent in the partition D_r range by frequency size level. It updates during the traversal of each level of the lattice with at least one positive term, the infrequent gradual patterns of said level in the set $mapIF_k^r$ from $mapIF^G$.
- The procedure **genCandidateFreqUnionTwoPartitionConsecutive** (F_{p1}, F_{p2}, $mapF^G$, $mapIF^G$) allows to generate and update the set of global frequent and infrequent candidates obtained from the sets F p 1 and F p 2 here partitions being processed.

In the algorithm 2, the data structure *resultatR* has five fields: *typeMap* which represents the indicator on the set choose between $mapF^G$ and $mapIF^G$, where we will find the gradual item index value of the suffix of the level pattern prefix, $Prefix_{level}$ maximal, min1 and max1 are the index bounds of over-patterns of $Prefix_{level}$ in $mapF^G_{level+1}$; min2 and max2 are the index bounds of the over-patterns of $Prefix_{level}$ in $mapIF^G_{level+1}$. The construction of *resultatR* is carried out using the function **byPrefixFindPositionsMinMax** of the algorithm 2. The function **matrixAdjacency** determines the adjacency matrix of the gradual pattern taken as a parameter. The **genCandidatOfALevel** function generates candidate patterns of size level + 1, following the principle described in step 3-(e) of Section 4.1. In this algorithm, on line 1, the **productCartesian** function generates a set of patterns resulting from the Cartesian product of the two sets taken as a parameter; Here ($get(F_{ki}^{pi})$) (resp. $\overline{get(F_{ki}^{pi})}$) represents the ordered set of the gradual patterns of the level k_i of $mapF_{pi}$ (resp. the set of gradual patterns complementary to each pattern of level k_i of $mapIF_{pi}$).

The function *filterSetByInfrequentSetAndSupportCompute* (*candidateFusion*) allows: (1) to prune first the candidates of its set candidateFusion in parameter, which are supermotif of a inferred pattern of $mapIF^G$. If the candidate to prune is $C_k = \{A_i\}_{i=1...k}$ then it generates two new candidates of size k-1, $C1_{k-1} = \{A_i\}_{i=1...k-1}$, $C2_{k-1} = \{A_i\}_{i=1...k-2} \bigcup A_k$ which are added to the potential candidate list list. On the other hand if C k is frequent then, we add C_k at level k of $mapF^G$ and its two frequent sub-patterns $C1_{k-1}, C2_{k-1}$ build exactly as above at level k-1 of $mapF^G$. Once *candidateFusion* = $\emptyset$ we have completed the process (1) and we have a valid candidate set list. Second, in (2) we have to perform another filtering by support calculation. Here, for any candidate k-pattern $C_k = \{A_i\}_{i=1...k}$ of *list*, $C_k \in list$: if C_k is frequent then we add C_k at level k of $mapF^G$, and $C1_{k-1}$, $C2_{k-1}$ at level k-1 of $mapF^G$, otherwise delete C k of list and add at the end of the list C1 k-1 , C2 k-1 as a new candidate in the list list. We repeat this process until *list* = $\emptyset$.

Note: In each of the algorithms below, before calculating the support of a k-pattern we check if it does not belong to one or the other of the sets $mapF_k^G$ or $mapIF_k^G$, because indeed the k-pattern may well have been determined during the ascending scan of the search space of the trellis to the first positive term or during the descending scan. The interest is to reduce the number of support computations to the maximum which is a greedy operation.

4.2 Principle of Finding Maximum Gradual Patterns

The method we use is based on Sgrite, which itself is an optimized method of Grite in terms CPU time for extracting gradual patterns. Indeed the MPSgrite method that we develop in this article has two objectives to achieve. The first objective is to optimize the time for extracting the gradual patterns considered, and the second goal is to reduce the number of gradual patterns extracted. Indeed, in real life, experts in the field say that the fewer patterns extracted, the easier it is to interpret and make decisions. We first opt for an approach of dual traversal of the lattice space to the first positive term from levels 1 to n by Sgrite and simultaneously from levels n to 1. The first problem of this combined approach is the generation of the candidates of the maximal set which is of the order of 2^{n-1}, with n the number of gradual items of the database. Consequently, the generation of candidates will have a higher CPU time cost. In addition, we also note that:

Lemma 25. The greater the number of maximum initial candidates, the higher the determination of the following maximal candidate sets, as well as the operation of fusion of n adjacency matrix composing the n-candidate gradual motifs considered.

Thus, to keep an optimal method of extracting the gradual patterns of said maxima, it will be a question of opting for a method hybridization. The base will be based on the choice to be made according to the parameter n number of gradual items. Consider dataset D of n items, t transactions, a fixed value p representing which method to use. Under these conditions, if n is less than or equal to p then MPSGrite uses the two-way browse method sense, that is, simultaneously ascending and descending from the lattice to a positive term. On the contrary, if n is strictly greater than p then a bottom-up lattice traversal approach is used which first efficiently generates the lattice of frequent gradual patterns, and then conversely in the descent of the lattice, by "backtracking", we extract the frequent and maximal gradual patterns.

Algorithm 1 Extraction of frequent gradual patterns with partitioning.

Require: dataset $\mathcal{D}$; minimum support threshold **minsupport**; number of partitions n; $(n_1, n_2, \ldots, n_n)$;
Ensure: $mapF^G$; $mapIF^G$;
{Creation of partitions p1 to pn}
partition $(\mathcal{D}, n, [n_1, n_2, \ldots, n_n])$;
{Extraction of frequent gradual itemsets in each partition}
for $r = 1$ **to** n **do**
  read partition $\mathcal{D}_r$;
  $mapF_r \leftarrow$ Partition-Gen-Sgrite ($\mathcal{D}_r$, minsupport); {rule}^{2}
end for
{Mergers of gradual itemsets frequent excerpts same level in each partition.}
for $(k = 1, k <= n; k++)$ **do**
  $mapF_k^G \leftarrow \cup_{r=1}^{r=n} mapF_k^r$;
  putToMap (k,$mapF_k^G, mapF^G$);
  putToMap (k,$\cup_{r=1}^{r=n} mapIF_k^r, mapIF^G$);
end for
$Fp_1 \leftarrow mapF_1$;
$Fp_2 \leftarrow mapF_2$;
$Fp_1 \leftarrow genCandidateFreqUnionTwoPartitionConsecutive(Fp_1, Fp_2, mapF^G, mapIF^G)$;
{Calculation of global itemset support}
for $r = 3$ **to** n **do**
  $Fp_2 \leftarrow mapF_r$;
  $Fp_1 \leftarrow genCandidateFreqUnionTwoPartitionConsecutive(Fp_1, Fp_2, mapF^G, mapIF^G)$;
end for
return $(mapF^G, mapIF^G)$;

4.3 Presentation of the Components of the Hybrid Method

The search space is limited in each of the components below to the lattice with a positive term. Two components of the hybrid method are required, namely component 1 and component 2. Component 1 takes place following the path in two simultaneously ascending and descending directions of the positive lattice, its description is carried out in Section 4.

4.3.1 Component 2: ascending the positive lattice

This algorithmic component proceeds in two main steps: In step 1, it is a question of extracting the frequent gradual patterns performed by Sgrite. Once this first step is completed its result will be an entry for step 2.

In step 2, we generate the maximum gradual patterns from the frequencies of step 1. During the generation we must respect the notion of lexicographic order of Apriori, Grite and SGrite. Let m $\leq$ n is the size of frequent gradual patterns of maximum cardinality.

In this case the set of so-called frequent maximum is initialized by all the frequent gradual m-patterns. Then one proceeds iteratively to prune the level k-1 of all the sub-patterns which allowed the construction of the maximum gradual k-patterns previously determined and purified at iteration k. This process continues in this way until the current processing value of k is 1. In fact, when k = 1, the maximum 1-gradual patterns are determined. This completes the "backtracking" determination of the maximum step patterns.

5 Experimentation

This section experimentally compares the performance of SGRrite and the novel hybrid approach MPSGrite. We used three sets of data. The first two are test data called F20Att100Li having 20 attributes, 100 transactions and F20Att500Li having 20 attributes, 500 transactions and taken from the site https://github.com/bnegreve/paraminer/tree/master/data/gri. The last set of data made up of meteorological data from the site http://www.meteo-paris.com/ile-de-france/station-meteo-paris/pro/. For further experimentation we have added five other datasets: C250-A100-50 a test dataset and 4 other real ones which are LifeExpectancydevelopped, LifeExpectancydevelopping, winequality-red and fundamental.

5.1 Description of the Datasets

This part presents the data used for the experiments carried out in this work.

We used a practical database called the weather forecast downloaded from the site http://www.meteo-paris.com/ile-de-france/station-meteo-paris/pro/: these data come from the Parisian weather station of

Algorithm 2 genCandidateFreqUnionTwoPartitionConsecutive.

Require: $Fp_1; Fp_2; mapF^G; mapIF^G;$
Ensure: $mapF^G; mapIF^G;$
{fusion of the frequent patterns of the highest level of the 2 partitions of level k_1 and k_2}
1: $candidateFusion \leftarrow productCartesian(get(Fp_1^{k_1}), (get(Fp_2^{k_2}) \cup \overline{get(Fp_2^{k_2})});$
{Filtering of candidates: pruning of over-patterns of infrequents and infrequents determined by calculation of the support}
2: **while** $candidateFusion \neq \emptyset$ **do**
3: $candidateFusion =$ filterSetByInfrequentSetAndSupportCompute ($candidateFusion$);
4: **end while**
{initial reference level for the lattice path};
5: $level = 1; nextLevel = 2; k = k_1 + k_2;$
6: refList $\leftarrow mapF^G_{level}$;
7: **while** $level \leq length(mapF^G) - 1$ and $nextLevel \leq length(mapF^G)$ **do**
8: **for** $j = 1$ **to** *length(refList)* **do**
9: $Prefix_{level} \leftarrow get(j, refList);$
10: $resultatR =$ byPrefixFindPositionsMinMax$(mapF^G, mapIF^G$ $Prefix_{level}, nextLevel);$
11: $adjPrefix =$ matrixAdjacency$(Prefix_{level});$
12: genCandidatOfALevel$(mapF^G, mapIF^G, resultatR, adjPrefix, Prefix_{level}, nextLevel);$ $adjPrefix, Prefix_{level}, nextLevel)$
13: **end for**
14: $level \leftarrow level + 1;$
15: $nextLevel \leftarrow nextLevel + 1;$
16: $refList = \leftarrow mapF^G_{level};$
17: **if** $not \exists mapF^G_{nextLevel}$ **then**
18: putToMap($mapF^G$, nextLevel,$\emptyset$);
19: **end if**
20: **end while**
21: **return** $(mapF^G, mapIF^G);$

Saint-Germain-des-Prés. The database contains 516 practical observations made over two days (July 22–23, 2017), described by 26 numerical attributes such as temperature, accumulated rain (mm), humidity (%), pressure (hPa), wind speed (km/h), wind gust speed (km/h), perceived temperature or distance traveled by the wind (km). Results obtained by Sgrite by setting the gradual support threshold s = 10%, 130 gradual patterns are extracted by the Sgrite algorithm [5].

The C250-A100-50 data set is taken from the site https://github.com/bnegreve/paraminer/tree/master/data/gri. For reason of memory space we have reduced the initial number of item 100 to 12, because otherwise the extraction is not possible on our computer. winequality-red.data is taken from the site https://archive.ics.uci.edu/ml/datasets/wine+quality. It is the Wine Quality dataset related to red vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests. The attributs of dataset uses the input variables (based on physicochemical tests): fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and output variable (based on sensory data) quality (score between 0 and 10).

LifeExpectancydevelopped.csv and LifeExpectancydeveloping.csv [5]. This two data set is also the real data set is taken for the site https://www.kaggle.com/kumarajarshi/life-expectancy-who that are open access data. The data was collected from World Health Organization (WHO) and United Nations website with the help of Deeksha Russell and Duan Wang. For this life expectancy dataset, attributes 1 and 3 are removed and the rest are used [5]. The dataset is designed to answer some key questions such as: do the different predictors I initially select actually affect life expectancy? Should countries with low life expectancy (under 65) increase health spending to increase life expectancy? Is life expectancy related to diet, lifestyle, exercise, smoking, alcohol etc.? Is there a positive or negative relationship between life expectancy and alcohol consumption? Do densely populated countries have a lower life expectancy? How does immunization coverage affect life expectancy? The final merged file (final dataset) consists of 22 columns and 2938 rows or 20 predictors. All prognostic variables are: immunization factors,

mortality factors, economic factors and social factors. Due to the size of the original dataset, we split the data into two groups: LifeExpectancydevelopping.csv for developed countries and LifeExpectancydevelopping.csv for developing countries, where we removed transactions with values empty.

The fundamental dataset is a playground for fundamental and technical analysis of the New York Stock Exchange (S&P 500 companies historical prices with fundamental data) The fundamentals.csv dataset contains metrics extracted from annual SEC 10K fillings (2012–2016), should be enough to derive most of popular fundamental indicators. The fundamentals.csv come from Nasdaq Financials. For this dataset, we have at begining 77 attributes. After preprocessing which consisted of removing empty-valued transactions, we derived a dataset with 1299 transactions and 74 attributes. The removed attributes are the first four: stock symbol, end of period, accounts payable, accounts receivable. for more information, see https://www.kaggle.com/dgawlik/nyse?select=fundamentals.csv; for reasons related to the characteristics of our small memory computer, we extracted part of the fundamental.csv data set for the experiments, which gave us a data set of 300 transactions and 35 attributes. Transactions are the top 300 and attributes are the top 35 [5].

5.2 Evaluation of Algorithms

All experiments on the datasets described in previous section were done on a Intel Core T M i7-2630QM CPU @ 2.00GHz × 8 with 8GB main memory, running Ubuntu 16.04 LTS. For each dataset, we considered a number of support thresholds and recorded the corresponding execution times as depicted on Figures 3, 5, 8, 10, 12, 14, 16, and 18 and the number of extracted patterns as depicted on Figures 4, 6, 7, 9, 11, 13, 15, 17, and 19. In these figures, (N It. X M Tr.) denotes the number items (N) and the number of transactions(M) of a dataset.

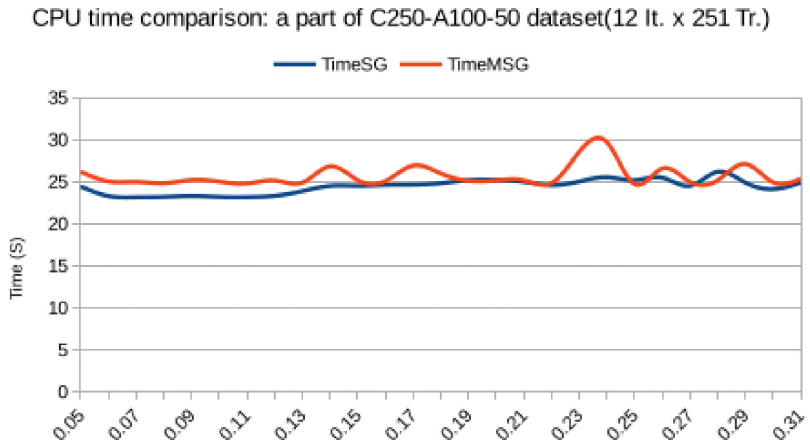
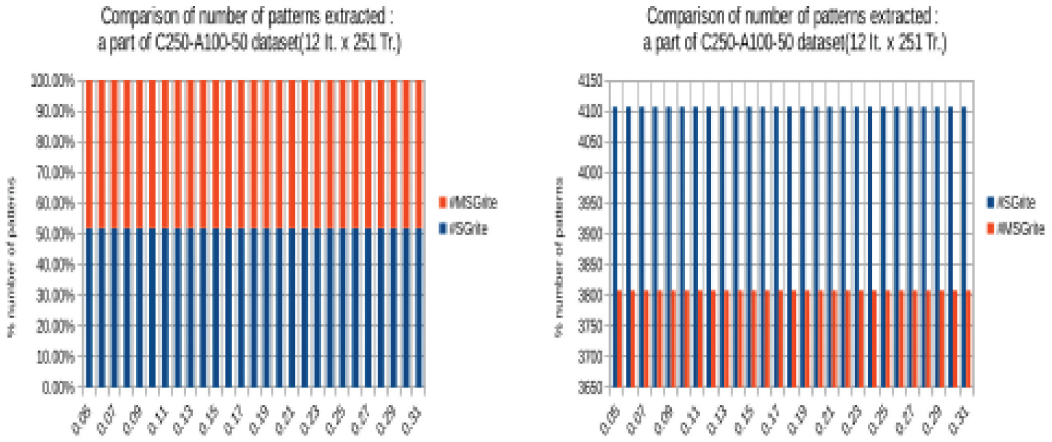


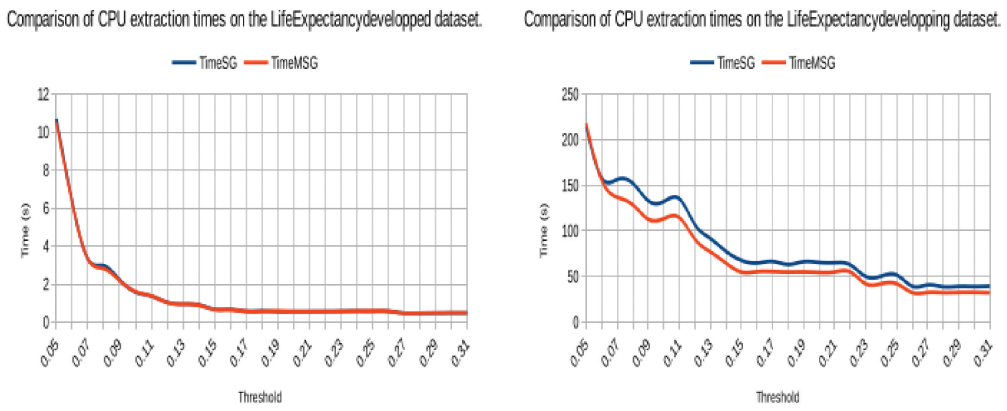
Figure 3. Different CPU times (Tr. (Resp. It.) Denotes Transactions (resp. Items)) Dataset C250-A100-50, 251 Tr. et 12 It.



(a) experimentation 1 Data set C250-A100-50.

(b) experimentation 2 Data set C250-A100-50

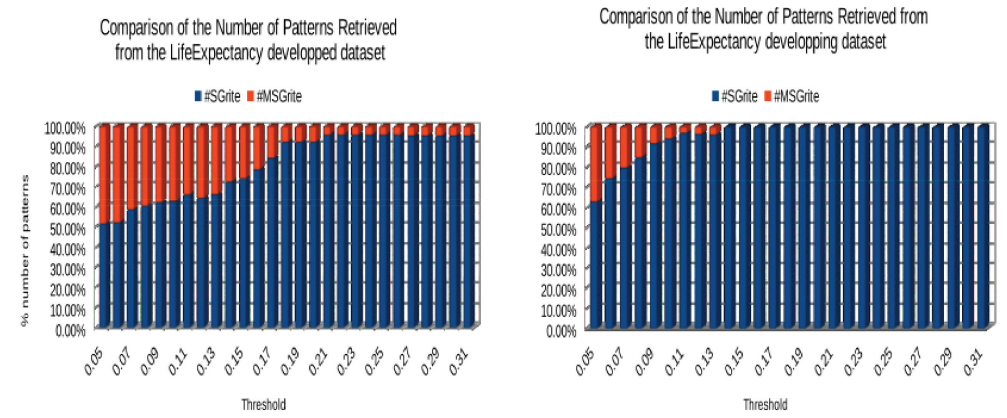
Figure 4. Experimentation on dataset C250-A100-50 for the number of patterns gradual extracted.



(a) Data set LifeExpectancydevelopped, 245 Tr. et 20 It.

(b) Data set LifeExpectancydeveloping, 1407 Tr. et 20 It.

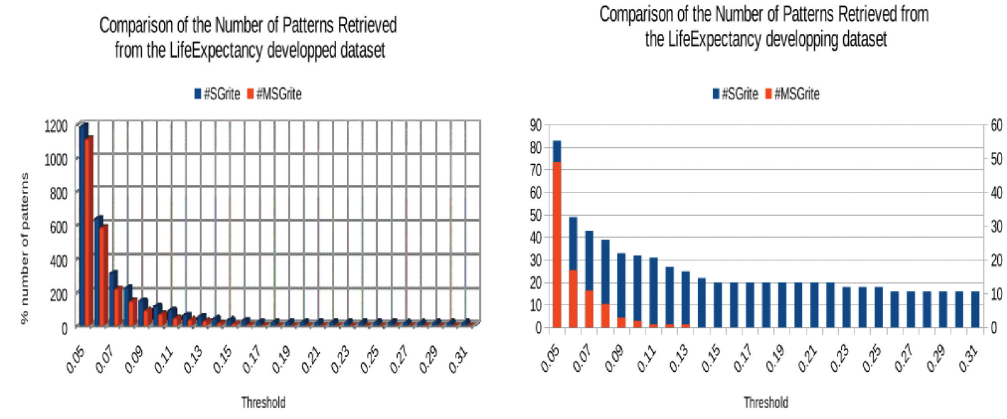
Figure 5. Different CPU times (Tr. (Resp. It.) Denotes Transactions (resp. Items)) for Life Expectancy.



(a) exp. 1 Data set Life Expectancy developped.

(b) exp. 1 Data set Life Expectancy developping.

Figure 6. Experimentation on dataset Life Expectancy for the number of gradual patterns extracted.



(a) exp. 2 Data set Life Expectancy developped.

(b) exp. 2 Data set Life Expectancy developping.

Figure 7. Number of gradual frequent patterns extracted over the life expectancy dataset.

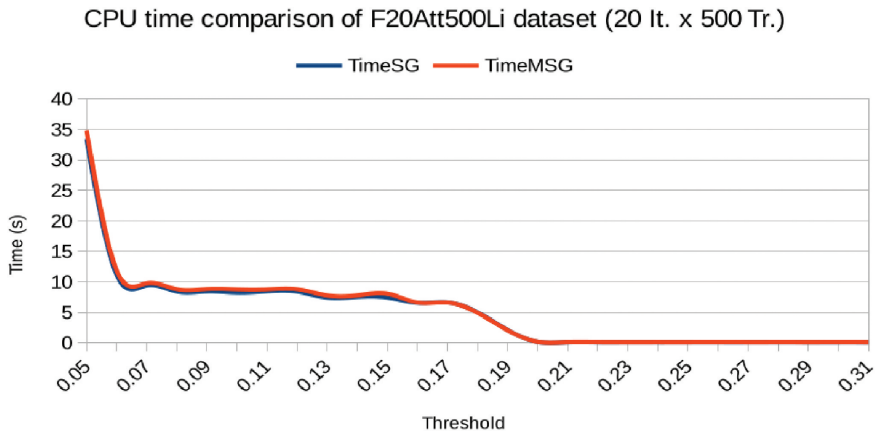
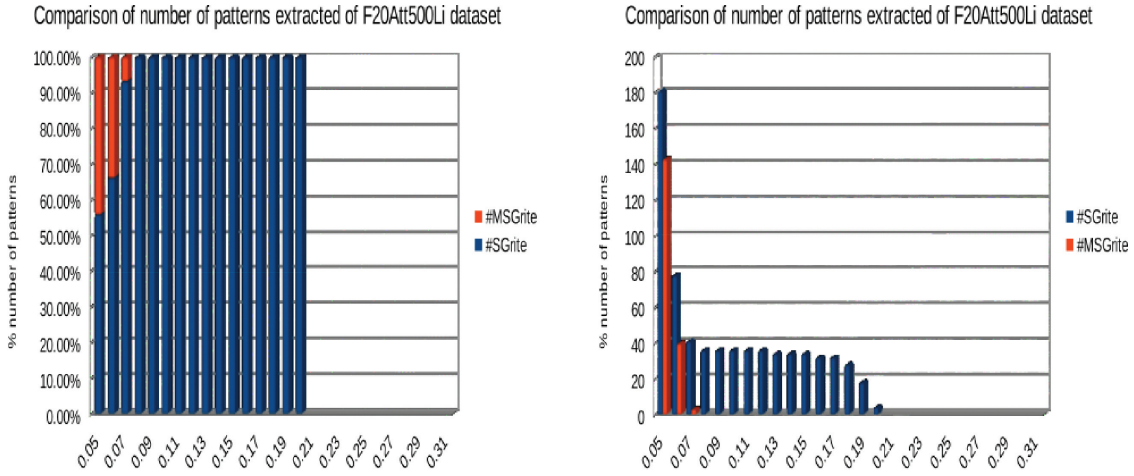


Figure 8. Different CPU times (Tr. (Resp. It.) Denotes Transactions (resp. Items)) forF20Att500Li 500 Tr. et 20 It.



(a) exp. 1 Data set F20Att500Li. (b) exp. 2 Data set F20Att500Li.

Figure 9. Experimentation of data set F20Att500Li on number gradual patterns extracted.

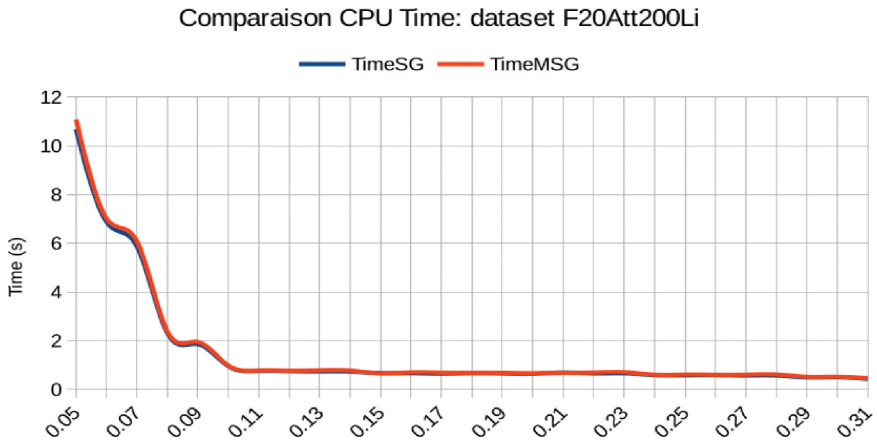
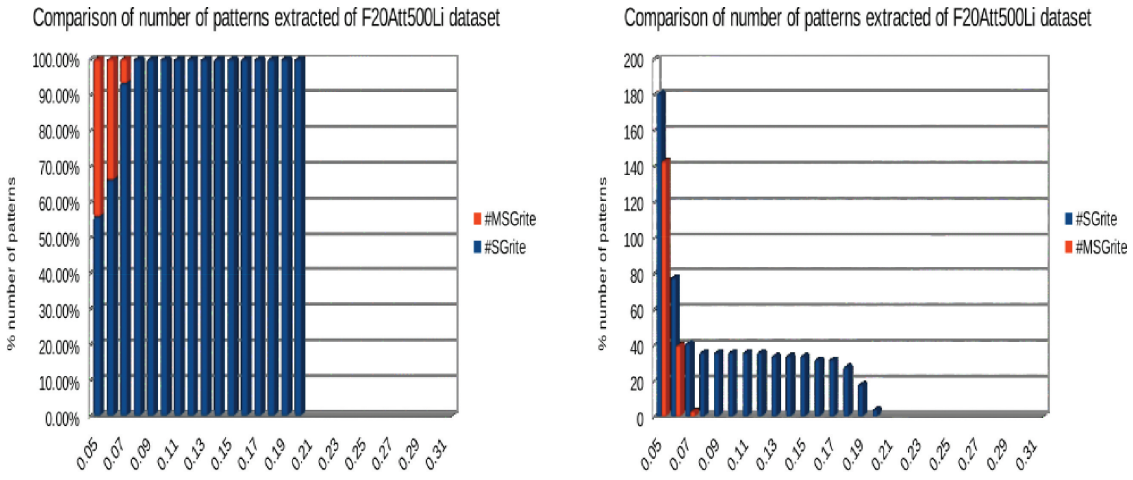


Figure 10. Different CPU times (Tr. (Resp. It.) Denotes Transactions (resp. Items)) forF20Att200Li 200 Tr. et 20 It.



(a) exp. 1 Data set F20Att200Li. (b) exp. 2 Data set F20Att200Li.

Figure 11. Experimentation of data set F20Att200Li on number gradual patterns extracted.

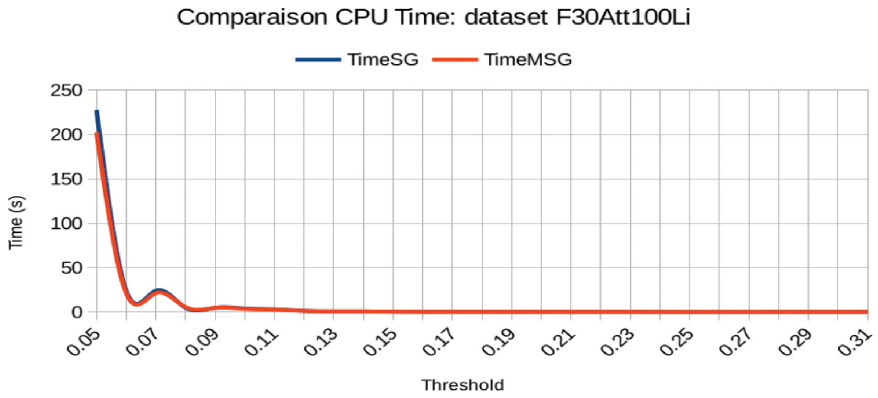
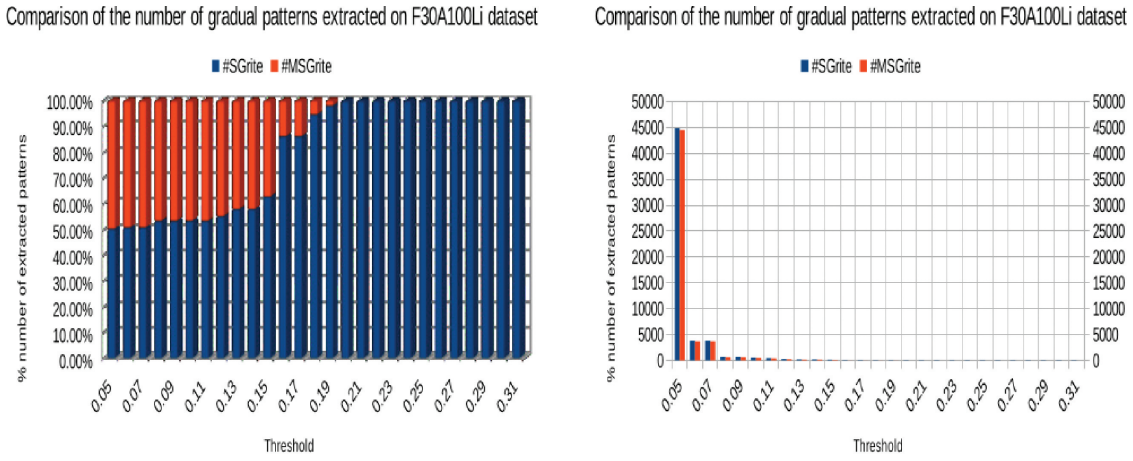


Figure 12. Different CPU times (Tr. (Resp. It.) Denotes Transactions (resp. Items)) forF30Att100Li 100 Tr. et 30 It.



(a) exp. 1 Data set F30Att100Li. (b) exp. 2 Data set F30Att100Li.

Figure 13. Number of gradual frequent patterns extracted for the F30Att100Li dataset.

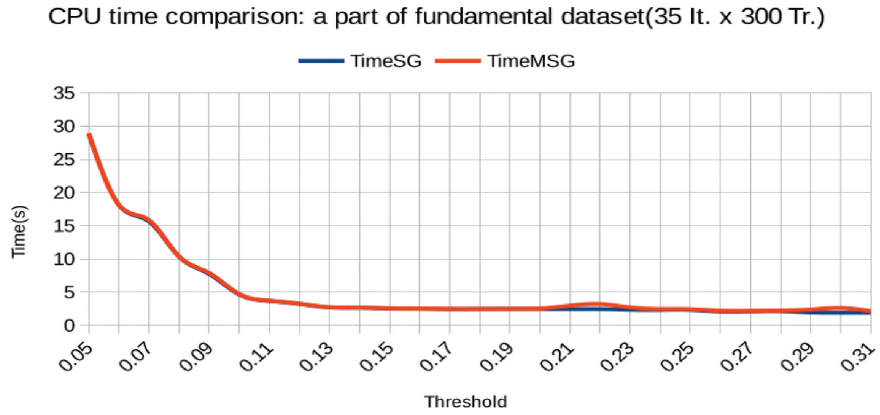
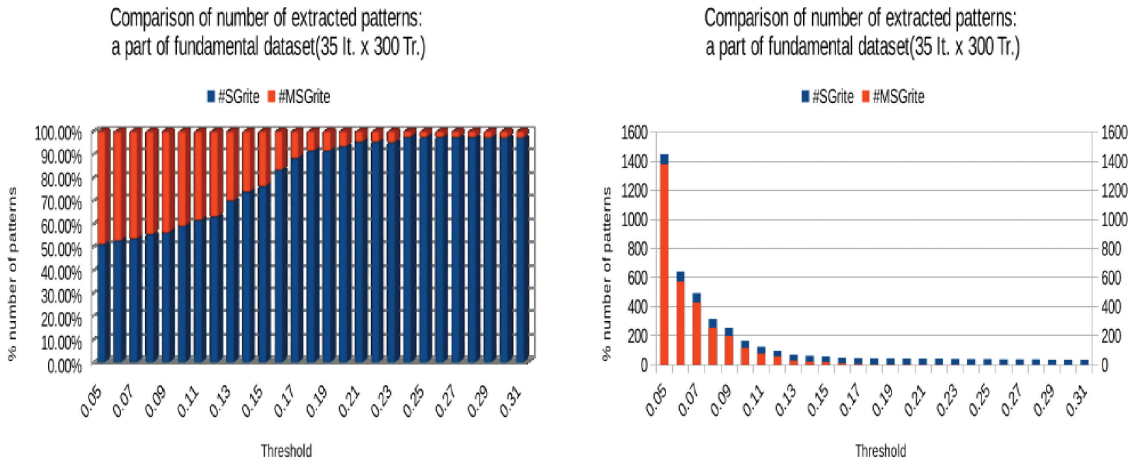


Figure 14. Different CPU times (Tr. (Resp. It.) Denotes Transactions (resp. Items)) data set fundamental, 300 Tr. et 35 It.



(a) exp. 1 Data set fundmental. (b) exp. 2 Data set fundamental.

Figure 15. Experimentation of data set fundamental on number gradual patterns extracted.

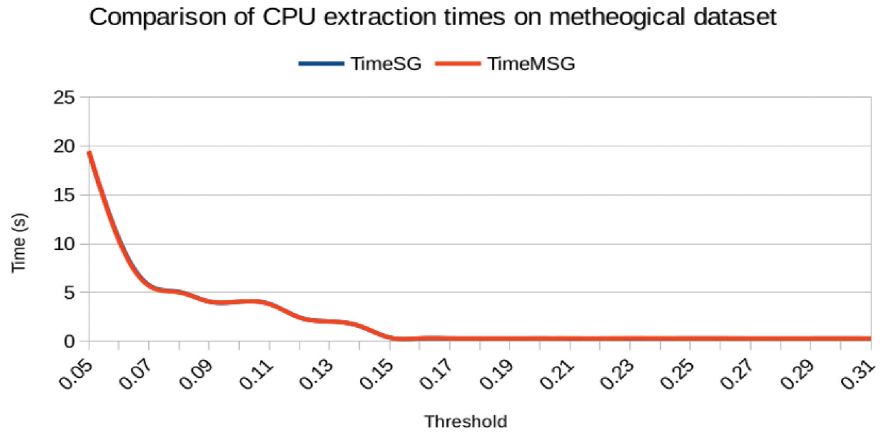
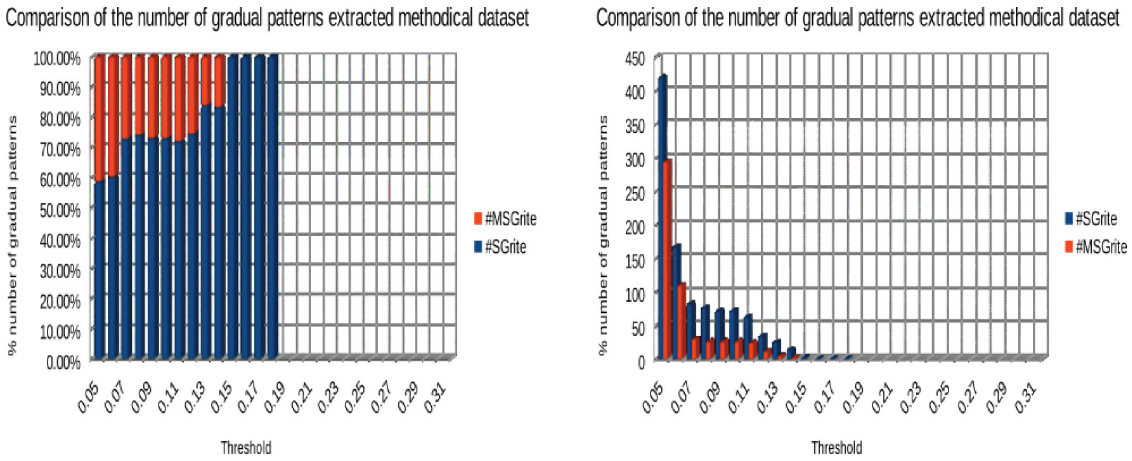


Figure 16. Comparison of execution times on meteorological data made up of 516 trans-actions and 26 items.



(a) exp. 1 Data set meteorological. (b) exp. 2 Data set meteorological.

Figure 17. Experimentation of data set meteorological on number gradual patterns ex-tracted.

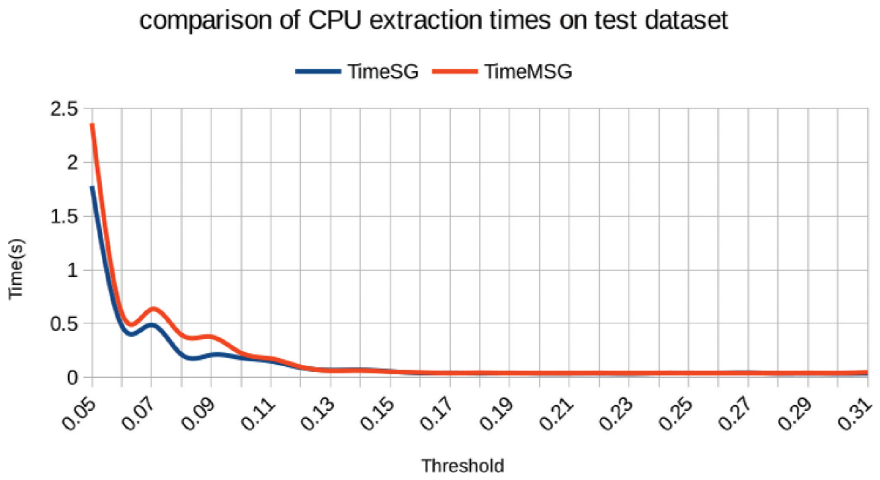
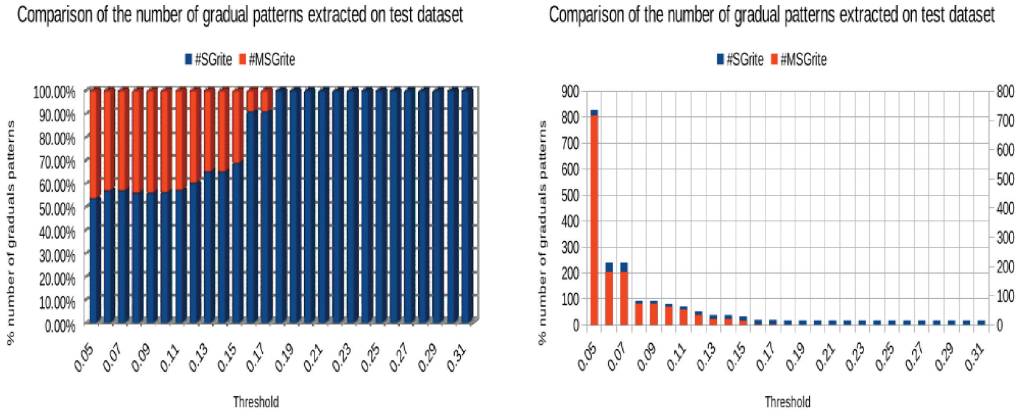


Figure 18. Comparison of execution times on test data made up of 100 tr. and 10 it.



(a) exp. 1 Data set test. (b) exp. 2 Data set test.

Figure 19. Number of gradual frequent patterns extracted for the test dataset.

6 Conclusion

In this paper, we have presented an approach to improve the performance of algorithms for discovering frequent and maximal gradual patterns by reducing by half both the search space, and the load of the computation of gradual supports on the large dataset. Various experiments carried out on various types of well-known datasets confirm the efficiency of the proposed approach. In future work, we will consider bigger datasets and explore the possibilities of distributed processing.

References

[1] A. Oudni, Fouille de données par extraction de motifs graduels: contextualisation et enrichissement, Ph.D. thesis, Université Pierre et Marie Curie-Paris VI, 2014.

[2] C. C. Aggarwal, Data mining: the textbook, Springer, 2015.

[3] S. Ayouni, Etude et extraction de regles graduelles floues: définition d'algorithmes efficaces, Ph.D. thesis, Université Montpellier, 2012.

[4] B. Négrevergne, A. Termier, M. Rousset, J. Méhaut, Para miner: a generic pattern mining algorithm for multi-core architectures, Data Min. Knowl. Discov. 28 (2014) 593–633. doi:10.1007/s10618-013-0313-2.

[5] T. D. Clémentin, T. F. L. Cabrel, K. E. Belise, A novel algorithm for extracting frequent gradual patterns, Machine Learning with Applications 5 (2021) 100068. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000347 doi: https://doi.org/10.1016/j.mlwa.2021.100068.

[6] T. Ngo, V. Georgescu, A. Laurent, T. Libourel, G. Mercier, Mining spatial gradual patterns: Application to measurement of potentially avoidable hospitalizations, in: A. M. Tjoa, L. Bellatreche, S. Biffl, J. van Leeuwen, J. Wiedermann (Eds.), SOFSEM 2018: Theory and Practice of Computer Science, volume 10706, Springer International Publishing, Cham, 2018, pp. 596–608. doi:10.1007/978-3-319-73117-9_42, series Title: Lecture Notes in Computer Science.

[7] D. Owuor, A. Laurent, J. Orero, Mining fuzzy-temporal gradual patterns, in: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2019, pp. 1–6. doi:10.1109/FUZZIEEE.2019.8858883, iSSN: 1558-4739.

[8] F. Shah, A. Castelltort, A. Laurent, Handling missing values for mining gradual patterns from NoSQL graph databases, Future Generation Computer Systems 111 (2020) 523–538. doi:10.1016/j.future.2019.10.004.

[9] L. Di Jorio, Recherche de motifs graduels et application aux données médicales, Ph.D. thesis, Montpellier 2, 2010.

[10] A. Laurent, M.-J. Lesot, M. Rifqi, Extraction de motifs graduels par corrélations d'ordres induits, Rencontres sur la Logique Floue et ses Applications, LFA'2010 (2010).

[11] L. Di-Jorio, A. Laurent, M. Teisseire, Mining frequent gradual itemsets from large databases, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 297–308.

[12] E. Hüllermeier, Association rules for expressing gradual dependencies, in: T. Elomaa, H. Mannila, H. Toivonen (Eds.), Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19–23, 2002, Proceedings, volume 2431 of Lecture Notes in Computer Science, Springer, 2002, pp. 200–211. doi:10.1007/3-540-45681-3_17.

[13] F. Berzal, J. C. Cubero, D. Sánchez, M. A. V. Miranda, J. Serrano, An alternative approach to discover gradual dependencies, Int. J. Uncertain. Fuzziness Knowl. Based Syst. 15 (2007) 559–570. doi:10.1142/S021848850700487X.

[14] C. Marsala, A. Laurent, M.-J. Lesot, M. Rifqi, A. Castelltort, Discovering ordinal attributes through gradual patterns, morphological filters and rank discrimination measures, in: D. Ciucci, G. Pasi, B. Vantaggi (Eds.), Scalable Uncertainty Management, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2018, pp. 152–163. doi:10.1007/978-3-030-00461-3_11.

[15] Y. S. Aryadinata, Y. Lin, C. Barcellos, A. Laurent, T. Libourel, Mining epidemiological dengue fever data from brazil: A gradual pattern based geographical information system, in: A. Laurent, O. Strauss, B. Bouchon-Meunier, R. R. Yager (Eds.), Information Processing and Management of Uncertainty in Knowledge-Based Systems, Communications in Computer and Information Science, Springer International Publishing, Cham, 2014, pp. 414–423. doi:10.1007/978-3-319-08855-6_42.

[16] T. Djamégni Clémentin, T. Fotso Laurent Cabrel, K. E. Belise, Un nouvel algorithme d'extraction des motifs graduels appelé Sgrite, in: CARI 2020 – Colloque Africain sur la Recherche en Informatique et en Mathémathiques Appliquées, Thiès, Senegal, 2020. URL: https://hal.archives-ouvertes.fr/hal-02925778.