How well go Lattice Algorithms on currently used Machine Learning TestBeds ?

Huaiguo Fu, Engelbert Mephu Nguifo

CRIL-CNRS FRE2499, Université d'Artois
Rue de l'université SP 16, 62307 Lens cedex. France
{fu,mephu}@cril.univ-artois.fr

Abstract. Many research papers in classification or association rules increase the interest of Concept lattices structures for data mining (DM) and machine learning (ML). To increase the efficiency of concept lattice-based algorithms in ML, it is necessary to make use of an efficient algorithm to build concept lattices. In fact, more than ten algorithms for generating concept lattices were published. As real data sets for data mining are very large, concept lattice structure suffers from its complexity issues on such data. The efficiency and performance of concept lattices algorithms are very different from one to another. So we need to compare the existing lattice algorithms with large data. We implemented the four first algorithms in Java environment and compared these algorithms on about 30 datasets of the UCI repository that are well established to be used to compare ML algorithms. Preliminary results give preference to Ganter's algorithm, and then to Bordat's algorithm, which do not fit well with the recommendations of Kuznetsov and Obiedkov. Furthermore, we analyzed the duality of the lattice-based algorithms.

1 Introduction

Concept is an important and basic means of knowledge representation, since it represents abstraction and generalization of objects. A concept defines a subset of objects which shares some common attributes or properties. Concept lattice structure [Ganter and Wille, 1999] has shown to be an effective tool for data analysis, knowledge discovery, and information retrieval, etc [Mephu Nguifo *et al.*, 2002]. It shows how objects can be hierarchically grouped together according to their common attributes. Researchers of different domains study it in theory and application of data analysis and formal knowledge representation etc.

Several algorithms are proposed to build concepts or concept lattices of a context : Bordat [Bordat, 1986], Ganter (NextClosure) [Ganter, 1984], Chein [Chein, 1969], Norris [Norris, 1978], Godin [Godin *et al.*, 1995], Nourine [Nourine and Raynaud, 1999], Carpineto [Carpineto and Romano, 1996], and Valtchev[Valtchev and Missaoui, 2001], etc. Some algorithms can generate also diagram graphs of concept lattices. The performance of the lattice algorithm is very important for its application to data mining (DM). In fact real data sets for DM are very large, e.g. the customer data of a company. In the worst case, the generation of lattice nodes increases exponentially. The efficiency of concept lattice algorithms are different from one to another. So we need to compare the existing lattice algorithms with large data and make use of an efficient algorithm to satisfy the mining and learning task and to increase the efficiency of concept lattice-based algorithms in real applications.

Different works on comparison of lattice algorithms have been done. Guénoche [Guénoche, 1990] reviewed four algorithms : Chein, Norris, Ganter and Bordat. This is the first review of lattice algorithms, he pointed out theoretical complexity, but there is no experimental test for these algorithms. Godin et al. [Godin and Chau, 1998] presented incremental algorithms for updating the concept lattice and corresponding graph. Results of empirical tests were given in order to compare the performance of the incremental algorithms to three other batch algorithms : Bordat, Ganter, Chein. The test data is small and randomly generated. Kuznetsov et al. [Kuznetsov and Obiedkov, 2002] compared, both theoretically and experimentally, performance of ten well-known algorithms for constructing concept lattices. The authors considered that Godin was suitable for small and sparse context, Bordat should be used for contexts of average density, and Norris, CBO and Ganter should be used for dense contexts. The algorithms were compared on different randomly generated contexts using the density/sparness, and on one real dataset (SPECT heart database) of the UCI repository. The test data is small and randomly generated, only one real dataset is used.

If the experimental datasets are too small or random, it's not easy to appraise the performance of these algorithms for DM. So in order to analyze and compare concept lattices algorithms, we use a publicly available database [Blake *et al.*, 1998] which are often used in order to compare machine learning (ML) algorithms. Even if it is not demonstrated that this database which contains more than forty datasets is representative of practical applications, it is well established that these testbeds should be used to measure efficiency issues of a new ML algorithm. So it's necessary to show how concept lattice algorithms fits in such data. Conclusions could help to build efficient ML algorithm based on concept lattice.

When generating concepts, lattice algorithm focusses on objects or attributes. So if the number of objects is greater than the number of attributes, it might be interesting to build the concept node based on the minimum number between objects and attributes. We propose a new definition : dual algorithm, which consists of applying an algorithm to the same context by inverting rows and columns. The duality of lattice algorithm is considered in our comparison of lattice algorithms. The difference between algorithm and its dual algorithm is described.

Our goal is to describe preliminary results of our implementation (using JAVA on a pentiumIII 450 computer with 128MB RAM) of the four first published algorithms (Chein, Norris, Ganter and Bordat) and their dual algorithms for generating concept lattices on about 30 datasets of the UCI repository. We test also these algorithms in the worst case. Preliminary results give preference to Ganter's algorithm, and then to Bordat's algorithm, which do not fit well with recommendations of Kuznetsov and Obiedkov's.

This work is also motivated by the fact that there are no free available platforms with different concept lattice algorithms for researchers. This justifies our choice of JAVA for implementation.

The rest of this paper is organized as follows : we introduce the notion of concept lattice in section 2 ; In section 3 we present the four lattice algorithms used for our experiment ; In section 4 experimental comparisons are discussed.

2 Concept lattice

The theoretical foundation of concept lattice relies on the mathematical lattice theory [Birkhoff, 1967]. Concept lattice is used to represent the order relation of concepts.

Definition 2.1 *A* **context** *is defined by a triple $(G; M; R)$, where G and M are two sets, and R is a relation between G and M. The elements of G are called objects, while the elements of M are called attributes.*

For example, Figure 1 represents a context. $G(1,2,3,4,5,6,7,8)$ is the object set, and $M(a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8)$ is the attribute set. The crosses in the table describe the relation R between G and M, which means that an object verifies an attribute.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
1	×	×					×	
2	×	×					×	×
3	×	×	×				×	×
4	×		×				×	×
5	×	×		×		×		
6	×	×	×	×		×		
7	×		×	×	×			
8	×		×	×		×		

FIG. 1 – An example of context (G, M, R).

Definition 2.2 *Given a subset $A \subseteq G$ of objects from a context $(G; M; R)$, we define an operator that produces the set A' of their common attributes for every set $A \subseteq G$ of objects to know which attributes from M are common to all these objects :*

$A' := \{m \in M \mid gRm \text{ for all } g \in A\}$.

Dually, we define B' for subset of attributes $B \subseteq M$, B' denotes the set consisting of those objects in G that have all the attributes from B :

$B' := \{g \in G \mid gRm \text{ for all } m \in B\}$.

These two operators are called the **Galois connection** *of $(G; M; R)$. These operators are used to determine a formal concept.*

So if B is an attribute subset, then B' is an object subset, and then $(B')'$ is an attribute subset. We have : $B \subseteq M \Rightarrow B'' \subseteq M$. Correspondingly, for object subset A, we have : $A \subseteq G \Rightarrow A'' \subseteq G$.

Definition 2.3 *A* **formal concept** *of the context (G, M, R) is a pair (A, B) with $A \subseteq G$, $B \subseteq M, A = B'$ and $B = A'$. A is called extent, B is called intent.*

Definition 2.4 *If (A_1, B_1) and (A_2, B_2) are concepts, $A_1 \subseteq A_2$ (or $B_2 \subseteq B_1$), then we say that there is a hierarchical order between (A_1, B_1) and (A_2, B_2).*

All concepts with the hierarchical order of concepts form a complete lattice called **concept lattice**.

3 Concept lattice algorithms

Concept lattice algorithm plays an essential role for the application of concept lattice. More than ten algorithms for generating concept lattices were published. We select the four first published algorithms (Chein, Norris, Ganter and Bordat) to compare and analyze them on different aspects. The other algorithms are very often improvements of one of these four algorithms.

3.1 Algorithm type

Generally, concept lattice algorithms are divided into two types : batch algorithms and incremental algorithms. Batch algorithms construct completely the lattice from scratch when adding a new object or attribute, while incremental ones update lattice structure when adding a new object.

For example, algorithms of Bordat, Ganter, Chein, Lindig and Nourine are batch algorithms. There are three ways to generate concepts with batch algorithms :

- **Descending :** such as Bordat's algorithm, from the top concept, we build the maximal rectangles. The algorithm repeats the same process to generate the other subnodes.
- **Ascending :** We can generate concepts below, and then spread super-node, such as Chein algorithm.
- **Enumeration :** algorithm enumerates all the nodes of the lattice according to a certain order. For example, Ganter's algorithm uses lexicographical order to enumerate the nodes.

There are some incremental algorithms such as the algorithms of Norris, Godin, Capineto, Dowling and Valtchev. The idea of these algorithms is that the new object makes intersection with all the concepts in the lattice to update lattice structure.

3.2 The principle of the algorithms

3.2.1 Chein's algorithm

Chein's algorithm [Chein, 1969] builds concepts in a bottom-up manner. It repeats the following iterative method at every stage k.

For each object g_i, $(g_i,(g_i'))$ is considered as first layer L_1. L_k is the set of the rectangles of layer k. An arbitrary element of L_k is (G_i,G_i'). From L_k, we build the layer L_{k+1}. For every two elements of L_k : (G_i, G_i') and (G_j, G_j'), if $G_i' \cap G_j' \notin L_{k+1}$, then $(G_i \cup G_j, G_i' \cap G_j')$ is an element of L_{k+1}. Otherwise, merge all pairs that have the same $G_i' \cap G_j'$ as an element of L_{k+1}.

At the end we delete L_k's element whose attribute set is the same as L_{k+1}'s element.

3.2.2 Norris' algorithm

Norris' algorithm [Norris, 1978] is an incremental algorithm. For the context (G, M, R), when we add each objects g_k, the concepts set of this level L_K is generated from L_{K-1} in the same way. For the first object, L_1 contains only (g_1, g'_1).

Adding one object g_{k+1} to L_K, we can build L_{k+1}. $\forall(G_i,\ G'_i) \in L_K$. If $G'_i \subset g'_{k+1}$, then $(G_i \cup (g_{k+1}), G'_i) \in L_{k+1}$.

Otherwise, $(G_i, G'_i) \in L_{k+1}$, and furthermore we add $(G_i \cup (g_{k+1}),\ G'_i \cap (g'_{k+1}))$ to L_{k+1} if $(G_i,\ (g'_{k+1}) \cap G'_i)$ is maximum.

After examination of all the rectangles, if g'_{k+1} is maximum, we add the $(g_{k+1},\ g'_{k+1})$ in L_{k+1}.

3.2.3 Ganter's algorithm (NextClosure algorithm)

The principle of NextClosure algorithm [Ganter, 1984] uses the characteristic vector which represents arbitrary subsets A of M, to enumerate all concepts of $(G; M; R)$. Given $A \subseteq M$, $M = \{a_1, a_2, \ldots, a_i, \ldots, a_{m-1}, a_m\}$, $A \to A''$ is the closure operator. The lectically smallest attribute subset is $\emptyset''$. The NextClosure algorithm proved that if we know an arbitrary attribute subset A, the next concept (the smallest one of all concepts that is larger than A) with respect to the lexicographical order is $A \oplus a_i$, where $\oplus$ is defined by

$A \oplus a_i = (A \cap (a_1, a_2, \ldots, a_{i-1}) \cup \{a_i\})''$

$A \subseteq M$ and $a_i \in M$, a_i being the largest element of M with $A < A \oplus a_i$ by lexicographical order.

In other words, for $a_i \in M \backslash A$, from the largest element to smaller one of $M \backslash A$, we calculate $A \oplus a_i$, until we find the first time $A < A \oplus a_i$, then $A \oplus a_i$ is the next concept.

3.2.4 Bordat's algorithm

The Bordat's algorithm [Bordat, 1986] searches all concepts hierarchically and builds the concept lattices (Hasse diagram). It uses a top-down strategy, and is a level-wise algorithm. Its principle is first to find all the maximal object subsets of G, then to build the corresponding concepts, and finally to find the maximal object subsets of the object subsets found above. So there are clear hierarchical relations within all concepts of a context, so that we can generate concept lattices.

Bordat's algorithm doesn't only generate all concepts but also it builds links between these nodes. This procedure increases the time cost. So it needs large memory.

3.3 Dual algorithm

Analyzing the four algorithms, we find that one algorithm can focuss on objects or attributes. The performances of an algorithm can be different according to the number of objects and attributes. So every lattice algorithm can be described or implemented by focussing on objects or attributes. We propose a new definition : dual algorithm.

Definition 3.1 *A* **dual algorithm** *of concept lattice is an algorithm which can be applied to the same context by focussing either on objects or on attributes.*

In other words : we can use the same algorithm from two directions (objects (set) or attributes(set)) to generate the concept lattice. Two dual algorithms are usually considered to be the same, and we can get the same concept lattice with two dual algorithms. In fact, the idea of the algorithms is the same, but the time cost of algorithm isn't frequently identical.

Proposition 3.1 *The time cost of a dual algorithm for a context is equivalent to the time cost of original algorithm for dual context.*

A dual context of a context is obtained by inverting rows and columns.

4 Experimental comparison

The four algorithms and their corresponding dual ones are implemented in Java environment and are available through request. These algorithms are tested on a Pentium III 450, 128 MB RAM. In our experiment, we compared these algorithms on about 30 datasets of the UCI repository and on the worst cases.

4.1 Test on ML benchmarks

4.1.1 Benchmark databases

Real data for our experiment come from ML benchmarks : UCI repository. We have got about 30 databases to build binary contexts (see table 1). The biggest context has 67557 objects and 126 attributes. This is not as huge as on real databases. However it's larger than datasets used by Kuznetsov et al. [Kuznetsov and Obiedkov, 2002] and Godin et al. [Godin and Chau, 1998] in their experiments.

These datasets are ordered by the number of concepts. For two datasets (kr-vs-kp and connect-4), we didn't get the number of concepts with these algorithms in our computer, as they fail due to the lack of memory.

4.1.2 Running time of the 4 first algorithms

We tested every context with the four first algorithms. Figure 2 shows the running time results. Analyzing the experimental results, Ganter and Bordat algorithms are faster than others. Bordat's algorithm not only generates the nodes of the lattice but also it builds links between these nodes. So if we want really to compare the three others to Bordat's algorithm, it would be necessary to build their links between nodes.

4.1.3 Running time of the 4 dual algorithms

We consider that the performance is different between one algorithm and its dual algorithm. So we implement each algorithm and its dual algorithm to focus respectively on objects or attributes. The experimental results (see table 2) show that the

DataSet	ID	Objects	Attributes	Concepts
shuttle-landing-control	d03	15	24	52
adult-stretch	d01	20	10	89
lenses	d02	24	12	128
zoo	d07	101	28	377
hayes-roth	d06	132	18	380
servo	d09	167	19	432
SPECT_train	d04	80	23	909
post-operative	d05	90	25	1521
balance-scale	d18	625	23	2104
flare1	d17	323	32	2608
flare2	d21	1066	32	2987
soybean-small	d10	47	79	3253
monks-3	d14	432	19	3959
monks-1	d16	432	19	4463
monks-2	d15	432	19	5427
car	d22	1728	21	7999
breast-cancer-wisconsin	d25	699	110	9860
house-votes-84	d13	435	18	10642
SPECT_test	d11	187	23	14532
SPECT_two	d30	267	23	21548
audiology.standardized	d08	26	110	30401
tic-tac-toe	d20	958	29	59503
nursery	d27	12960	31	147577
lung-cancer	d12	32	228	186092
agaricus-lepiota	d28	8124	124	227594
promoters	d19	106	228	304385
soybean-large	d23	307	133	806030
dermatogogy	d24	366	130	1484088
kr-vs-kp	d26	3196	75	/
connect-4	d29	67557	126	/

TAB. 1 – The datasets of UCI repository ordered by the number of concepts. / means that the programs fail to generate all concepts.

performance of two dual algorithms are very different. For example, we have tested Ganter's algorithm and its dual algorithm for the dataset Flare2, and time cost can be 100 times different. So the difference between algorithm and its dual algorithm is marked, to show that we must consider duality when comparing lattice algorithms.

Figure 3 shows performance of the four algorithms and their dual algorithms. We can see that Ganter's algorithm runs faster than others. Figure 4 shows an important conclusion : Ganter's algorithm has the best performance when it focusses on the smallest number of objects or attributes. For example, for dataset d08, it has 26 objects and 110 attributes, the number of objects is smaller than attributes, so dual algorithm that focusses on objects is faster than that focussing on attributes. With the real database, the number of attributes is often smaller than that of objects. Ganter's algorithm works faster than others in this case. Ganter algorithm should search all closures using the smallest number between attributes or objects. This is the consequence of Ganter's algorithm since it explores almost all the subsets of the set of attributes or objects.

This is not the case with the three other algorithms. Norris' algorithm and its dual algorithm have the most difference. But Bordat's algorithm have little difference with its dual algorithm. It is not possible to infer from the analysis of the code of the 3 other algorithms that they should be used by focussing on the smallest number of attributes or objects. And the experiment seems to confirm that.

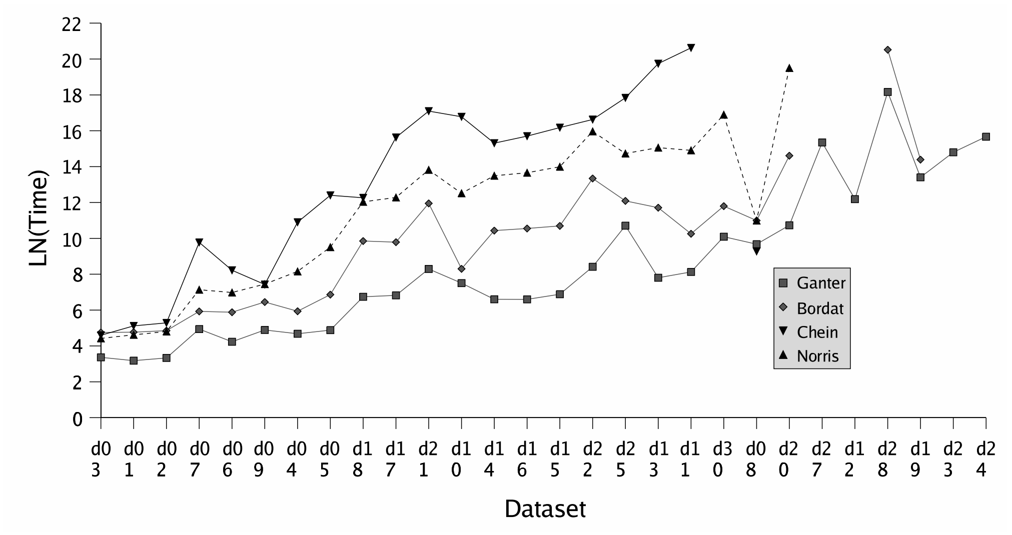

FIG. 2 – Performance (in ms) of the 4 first lattice algorithms on UCI datasets.

4.2 Test in the worst case

Definition 4.1 *A context in the* **worst case** *is the case where the sizes of* G *and* M *are equal to* n*, and each attribute is verified by* $n-1$ *different objects, each object possesses* $n-1$ *different attributes.*

We have tested four algorithms in the worst case. This particular case generates with context of size n (number of lines and number of columns) : 2^n nodes for concept lattice. The results (see figure 5) show that Ganter's algorithm is the best in worst case. It succeeds in computing some large data that were impossible to be computed with other algorithms. For example : the worst case with 20 attributes (2^{20} concepts) is very hard to compute with other algorithms, but Ganter's algorithm can build the concept lattice for this context. However each algorithm fails in building lattice nodes for context with more than 22 attributes.

5 Conclusion

The concept lattice algorithm to generate concepts or diagram graph is considered important in theory and for its application. We need algorithms of high level performance to satisfy the mining and learning task. Four algorithms are analyzed and are compared in this paper, of course, this work will be extended to other lattice algorithms. We use real dataset and worst cases datasets to test four algorithms in Java environment, the analysis shows that algorithms of Ganter and then Bordat are faster than others. Ganter's algorithm is the best for large and dense data. Bordat's algorithm can be used to generate the line diagram if the computer has enough memory.

ID	**Ganter**	**G-dual**	**Bordat**	**B-dual**	**Chein**	**C-dual**	**Norris**	**N-dual**
d03	29	37	116	167	99	194	83	119
d01	24	43	118	224	167	183	101	123
d02	28	83	128	204	198	232	122	140
d07	140	589	376	754	17607	14652	1255	1787
d06	69	707	359	660	3724	1462	1070	543
d09	133	1120	633	968	1681	1061	1713	478
d04	108	507	378	646	54283	80217	3467	3297
d05	132	1084	957	1554	243564	146154	13457	6620
d18	845	54825	18986	37388	211849	63231	168423	9986
d17	916	26149	17798	31592	6146858	3562383	214944	46890
d21	4000	395911	154984	778299	26633173	12807020	1002709	147099
d10	1819	1924	4027	7143	19392256	12515300	271203	58280
d14	737	41453	34005	18126	4457627	1272022	721953	34865
d16	734	45055	38153	20997	6583877	1376935	853978	48280
d15	975	50789	44152	24404	10623849	1904268	1190532	66662
d22	4516	1229305	623112	483612	16677919	2555755	8566689	145633
d25	44748	560389	177929	713596	55932215	34140525	2485251	703593
d13	2450	131862	121844	69740	375882938	154525507	3461533	463109
d11	3382	28432	28459	19540	/	/	1982204	1059784
d30	24144	102877	132799	41304	/	/	21695260	/
d08	15922	5288	59384	161548	10672	/	59192	10914983
d20	45681	1767983	2211559	472985	/	/	294433666	8667683
d27	4627030	/	/	/	/	/	/	/
d12	196487	75805	/	496263	/	/	/	/
d28	77183183	/	/	/	/	/	/	/
d19	663052	552271	1774334	1469483	/	/	/	/
d23	2676454	14959171	/	/	/	/	/	/
d24	6367387	/	/	/	/	/	/	/
d26	/	/	/	/	/	/	/	/
d29	/	/	/	/	/	/	/	/

TAB. 2 – The results of running time (in milliseconds) of lattice algorithms for real data. / means that the programs fails to generate all concepts.

In this paper we discuss for the first time dual algorithm for concept lattices. The difference between algorithm and its dual algorithm is presented. We should consider duality when comparing lattice algorithms.

Comparing our results with the platform of Kuznetsov and Obiedkov may be interesting and help researchers in Formal Concept Analysis in their implementations of lattice structure.

Even if this work shows performance of concept lattices algorithm in ML benchmarks, it is based on the generation of the whole concepts and not on the way ML algorithms are designed. In fact, not all the concepts are relevant for learning task. Thus the way to reach quickly to the relevant concepts should be take into account for the learning algorithm to be more efficient, and the choice of an efficient concept lattice algorithm should not omit that. We believe that it is not enough if we apply simply these four algorithms to deal with enormous data. We consider that we need to develop faster algorithm, or to improve existing algorithms, to raise the efficiency of the application of concept lattice [Fu and Mephu Nguifo, 2003, Valtchev *et al.*, 2002]. For example, we can use parallel method to improve the performance of the application of concept lattice. We can also apply fuzzy logic to build flexible lattice.

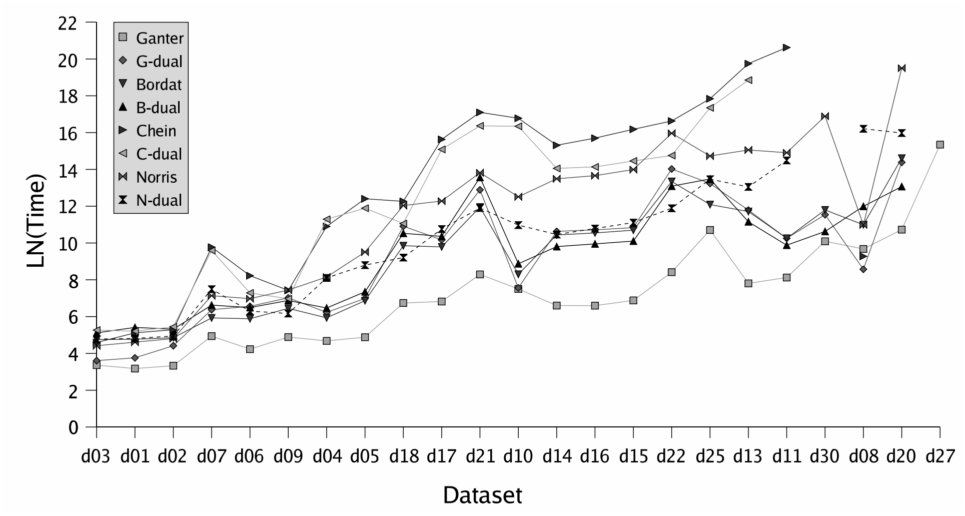

FIG. 3 – Performance of lattice algorithms and their dual algorithms.

Acknowledgements

We are grateful to anonymous reviewers for helpful comments and to Jean Jacques Givry for english proofreading. This research benefits from the support of IUT de Lens, Université d'Artois, CNRS and the region Nord/Pas de calais.

References

[Birkhoff, 1967] G. Birkhoff. *Lattice Theory.* American Mathematical Society, Providence, RI, 3rd edition, 1967.

[Blake *et al.*, 1998] C. Blake, E. Keogh, and C. Merz. UCI repository of machine learning databases, 1998. http ://www.ics.uci.edu/~mlearn/MLRepository.html.

[Bordat, 1986] J. Bordat. Calcul pratique du treillis de galois d'une correspondance. *Mathématiques, Informatiques et Sciences Humaines*, 24(94) :31–47, 1986.

[Carpineto and Romano, 1996] C. Carpineto and G. Romano. A Lattice Conceptual Clustering System and its Application to Browsing Retrieval. *Machine Learning*, 24 :95–122, 1996.

[Chein, 1969] M. Chein. Algorithme de recherche des sous-matrice premières d'une matrice. *Bulletin Math. de la Soc. Sci. de la R.S. de Roumanie*, 61(1), 1969. Tome 13.

[Fu and Mephu Nguifo, 2003] Huaiguo Fu and Engelbert Mephu Nguifo. Partitioning large data to scale up lattice-based algorithm. In *Proceedings of ICTAI03*, Sacramento, CA, November 2003. IEEE Computer Press.

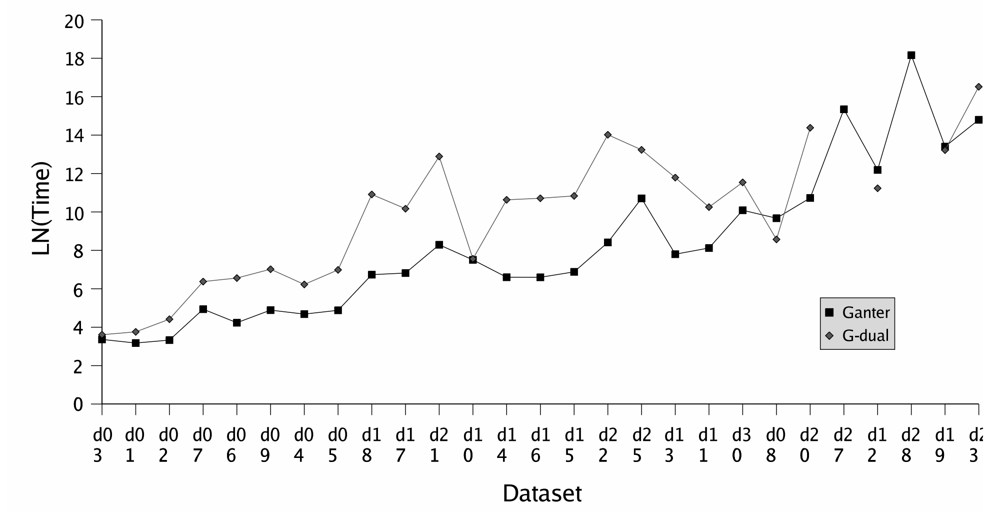

FIG. 4 – Performance of comparison on UCI datasets with Ganter algorithms.

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

[Ganter, 1984] B. Ganter. Two basic algorithms in concept analysis. Technical Report 831, Technische Hochschule, Darmstadt, Germany, 1984. preprint.

[Godin and Chau, 1998] Robert Godin and Thuy-Tien Chau. Comparaison d'algorithmes de construction de hiérarchies de classes. Technical report, Université de Québec, 1998.

[Godin *et al.*, 1995] R. Godin, G. Mineau, R. Missaoui, and H. Mili. Méthodes de classification conceptuelle basées sur les treillis de Galois et applications. *Revue d'intelligence artificielle*, 9(2) :105–137, 1995.

[Guénoche, 1990] A. Guénoche. Construction du treillis de galois d'une relation binaire. *Mathématiques et sciences humaines*, 109, 1990.

[Kuznetsov and Obiedkov, 2002] S. Kuznetsov and S. Obiedkov. Comparing performance of algorithms for generating concept lattices. *JETAI Special Issue on Concept Lattice for KDD*, 14(2/3) :189–216, 2002.

[Mephu Nguifo *et al.*, 2002] E. Mephu Nguifo, M. Liquiere, and V. Duquenne. *JETAI Special Issue on Concept Lattice for KDD.* Taylor and Francis, 2002.

[Norris, 1978] E.M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine Math. Pures et Appl.*, XXIII(2) :243–250, 1978.

[Nourine and Raynaud, 1999] L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71 :199–204, 1999.

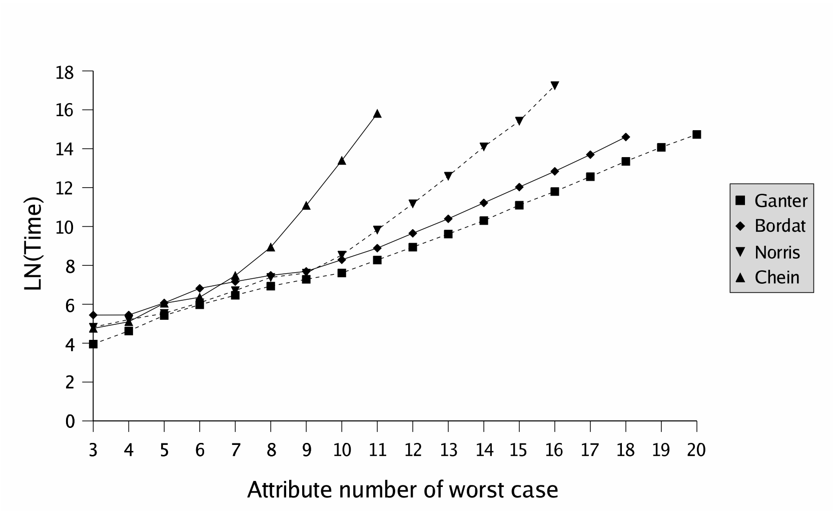

FIG. 5 – Performance of comparison on worst case data.

[Valtchev and Missaoui, 2001] Petko Valtchev and Rokia Missaoui. Building galois (concept) lattices from parts : Generalizing the incremental approach. In *Proceedings of the ICCS 2001, LNCS 2120*, pages 290–303. Springer Verlag, 2001.

[Valtchev *et al.*, 2002] Petko Valtchev, Rokia Missaoui, and P. Lebrun. A partition-based approach towards constructing galois (concept) lattices. *Discrete Mathematics*, 256(3) :801–829, 2002.

Résumé

Plusieurs travaux en classification et en recherche de règles d'association montrent l'intérêt des treillis de concepts pour la fouille de données et l'apprentissage automatique. Pour accroître l'efficacité des algorithmes d'apprentissage qui sont basés sur cette structure, il est nécessaire d'utiliser des algorithmes rapides de génération de concepts. Plus d'une dizaine d'algorithmes ont été publiés sur le treillis de concepts. Ces algorithmes présentent une complexité exponentielle, et sont donc difficilement utilisables sur des données volumineuses. En outre les temps de calcul sont très variables d'un algorithme à l'autre, en fonction aussi de la nature du contexte. Nous avons examiné l'efficacité des 4 premiers algorithmes de génération de concepts, sur les données de la base UCI couramment utilisée pour tester les algorithmes d'apprentissage. Les résultats préliminaires montrent que l'algorithme de Ganter est meilleur que celui de Bordat, lui même meilleur que les 2 autres. Ce constat diffère des conclusions des comparaisons effectuées par Kuznetsov et Obiedkov sur des données aléatoires de petite taille. Nous avons aussi examiné l'effet de la dualité du contexte (transposé) avec ces algorithmes.