

Comparaison d'algorithmes de génération de concepts

Chiraz El Hog — Samir Moalla

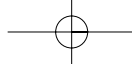
Département des Sciences de l'Informatique
Faculté des Sciences de Tunis
1060 Tunis
TUNISIE
elhog.chiraz@gmail.com, samir.moalla@fst.rnu.tn

.....
RÉSUMÉ. Différentes techniques d'analyse de données ont vu le jour au sein de la technologie de Data Mining ou Fouille de Données. L'un des problèmes majeurs des techniques qui utilisent l'Analyse Formelle de Concepts est le regroupement conceptuel des données dans un treillis de concepts, afin d'en extraire les informations implicites stockées dans des bases de données volumineuses. La détermination des éléments du treillis ainsi que les relations entre ses éléments ont fait l'objet d'une multitude de travaux visant à réduire le temps et l'espace mémoire nécessaires à ces opérations. Nous nous sommes intéressés dans cet article à la comparaison des performances de cinq algorithmes de génération de concepts, à savoir *Ganter*, *Bordat* et *Close By One* avec deux algorithmes *Concept1* et *Concept2* basés sur la théorie des graphes.

ABSTRACT. Generating concepts defined by a binary relation between a set of attributes and a set of objects is one of the important current problems encountered in Knowledge Discovery in Databases. Formal concept analysis is one of these interesting approaches used to generate lattice of concepts that is used in terms to find interesting patterns. In this paper, we present a comparison between three known concept generation algorithms e.g. *Ganter*, *Bordat* and *Close By One* and two new algorithms *Concept1* and *Concept2* based on graph theory.

MOTS-CLÉS : Fouille de données, Concepts formels, Treillis de Galois

KEYWORDS : Data Mining, Formal Concepts, Galois Lattice

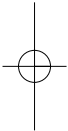
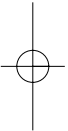


1. Introduction

Face à l'évolution des technologies d'acquisition et de stockage de données, nous assistons à la génération de très larges volumes de données qui dépassent la capacité d'analyse et d'extraction de connaissances [1]. Cependant, la grande masse de données collectées recèle des informations non exploitées mais potentiellement utiles pour les décideurs. Le Data Mining ou fouille de données, étape essentielle d'un processus d'extraction de connaissances, permet l'extraction automatique, à partir de données, d'informations ou de connaissances qui peuvent être utile au décideur final. Les informations extraites, suite à l'application d'un processus de fouille de données, peuvent se présenter sous plusieurs formes. Une des structures les plus utilisées est celle du treillis de Galois ou treillis de concepts. La popularité croissante du treillis de Galois, en tant qu'outil de Data Mining, a mis à l'ordre du jour le problème d'engendrer, le plus efficacement possible, un treillis de concepts. Ce problème a fait l'objet de plusieurs recherches à l'issue desquelles différents algorithmes ont été proposés, comparés et analysés [2, 3, 9].

C'est dans ce cadre que se situe cet article qui présente une étude comparative de cinq algorithmes de génération de concepts à savoir ceux de *Ganter*, *Bordat*, *Close By One* et deux algorithmes *Concept1* et *Concept2* basés sur la théorie des graphes. Ces deux derniers algorithmes ont été étudiés et testés en [5].

La suite de cet article est organisée comme suit : La Section 2 passe en revue les définitions et notations liées aux treillis de concepts formels. La Section 3 présente une description non exhaustive des algorithmes *Ganter*, *Bordat*, *Close By One* ainsi que les algorithmes *Concept1* et *Concept2*. Une comparaison expérimentale des performances de ces algorithmes, basée sur des bases de données synthétiques et réelles, est décrite dans la Section 4. Enfin, la Section 5 conclut cet article en présentant une synthèse de cette comparaison.



2. Définitions et notations

Dans cette partie, nous allons présenter les principales notions et définitions nécessaires à la compréhension de la suite de cet article.

Définition 1 Contexte formel [9]

Un contexte formel est un triplet $K = (O, A, R)$ où O et A sont deux ensembles finis et R une relation binaire entre O et A tel que $R \subseteq O \times A$. L'ensemble O est appelé ensemble d'objets (ou transactions) et A est appelé ensemble d'attributs (ou items).

Définition 2 Correspondance de Galois [10]

Soient $O_i \subseteq O$ et $A_i \subseteq A$, on définit f et g comme suit :

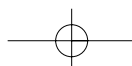
$$f : P(O) \rightarrow P(A), f(O_i) = \{a \in A / (o, a) \in R, \forall o \in O_i\} \text{ intension}$$

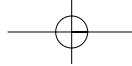
$$g : P(A) \rightarrow P(O), g(A_i) = \{o \in O / (o, a) \in R, \forall a \in A_i\} \text{ extension}$$

f et g sont deux applications monotones décroissantes et le couple (f, g) est appelé **correspondance de Galois** entre $P(O)$ et $P(A)$.

Définition 3 Concept formel (fermé, rectangle) [10]

Soient $O_i \subseteq O$ et $A_i \subseteq A$. Un concept formel de (O, A, R) est un couple (O_i, A_i) avec $O_i = g(A_i)$ et $A_i = f(O_i)$, tel que O_i est l'extension de A_i et A_i est l'intension de O_i . Le concept formel (O_i, A_i) est équivalent au produit cartésien $O_i \times A_i$ maximal pour la propriété suivante : $\forall o \in O_i, \forall a \in A_i, oRa$.





Définition 4 Treillis de concepts [3]

L'ensemble des concepts formels, extrait à partir d'un contexte formel, constitue un treillis complet quand les concepts formels sont ordonnés par inclusion des intensions ou par inclusion des extensions. Pour un contexte $K = (O, A, R)$, l'ensemble de tous les concepts formels munis de la relation d'ordre partiel forme un treillis complet appelé treillis de concepts ou treillis de Galois, noté $L(O, A, R)$.

3. Algorithmes de génération de concepts

Plusieurs travaux se sont intéressés à l'étude aussi bien théorique qu'expérimentale des algorithmes de génération de concepts tel que ceux de *Ganter*, *Bordat* et *Close By One*. Une nouvelle approche basée sur la théorie des graphes a été proposée par Sigayret [4] et a fait l'objet d'une étude approfondie et d'une expérimentation dans [5]. Deux nouveaux algorithmes de génération de concepts ainsi que des arcs du diagramme de *Hasse*, *Concept1* et *Concept2*, résultent de cette approche.

Algorithme de Bordat [6]

L'algorithme de *Bordat* construit simultanément les éléments du treillis et les arêtes de son graphe de *Hasse*. Il consiste à chercher de manière récursive descendante tous les concepts formels utilisant un parcours par niveau (en largeur) de ce graphe. En partant du supremum du treillis $A \times \emptyset$, on construit la liste des rectangles maximaux. Pour chaque élément de cette liste, étudiée séquentiellement, on cherche les rectangles maximaux qu'il couvre en se basant sur la relation de couverture de la relation d'ordre \leq . Cette méthode engendre chaque rectangle autant de fois qu'il a d'éléments immédiatement supérieurs. Ceci nécessite de conserver en mémoire la liste de tous les rectangles maximaux afin de vérifier que les rectangles trouvés n'ont pas été calculés. Toutefois, cette opération permet d'énumérer toutes les arêtes du graphe de *Hasse* du treillis.

Algorithme de Ganter [7]

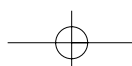
L'algorithme de *Ganter* (*Next Closure*) utilise l'ordre lexicographique sur l'ensemble d'attributs pour énumérer tous les concepts du treillis. Ceci dit, à partir d'un fermé A quelconque on détermine le plus petit fermé, selon l'ordre lexicographique, suivant A .

Algorithme Close By One [8]

Cet algorithme utilise la même stratégie de génération que celui de *Ganter* en ajoutant une structure intermédiaire pour accélérer la recherche de nouveaux concepts en se basant sur ceux déjà générés.

Algorithme Concept1 [2, 4]

Cet algorithme présente des similitudes avec celui de *Bordat*. Toutefois, *Concept1* introduit des améliorations au niveau du calcul de la couverture et du traitement des concepts déjà générés. Il s'appuie sur les résultats des travaux sur les séparateurs minimaux tels que la notion de domination et de couverture dans un graphe co-biparti associé à la relation. L'idée consiste à calculer de façon récursive la couverture de chaque concept en parcourant le treillis en profondeur à partir de l'élément bottom et de répéter ces calculs pour chaque concept généré. Selon ce processus, chaque concept sera généré autant de fois qu'il a de prédécesseurs. Pour remédier à cet inconvénient, une information supplé-



mentaire est sauvegardée, indiquant les attributs déjà utilisés vu que le treillis est construit par inclusion sur les intensions représentées par les attributs de la relation.

Algorithme Concept2 [2, 4]

Cet algorithme est basé sur le même principe de génération que l'algorithme *Concept1*. La différence réside dans les structures de données et la stratégie adoptée pour déterminer la couverture d'un concept. En effet, cet algorithme utilise une nouvelle structure de données "la table de domination", permettant de sauvegarder les informations globales sur l'ensemble des dominations de la relation de départ.

4. Comparaison des performances des algorithmes

Dans cette section, nous allons analyser les résultats des expériences que nous avons réalisées sur les algorithmes *Concept1* et *Concept2*, puis nous les comparerons aux algorithmes *Ganter*, *Bordat* et *Close By One*.

Toutes les expérimentations ont été réalisées sur un PC Centrino Duo de fréquence 1,66 GHz avec 2 Go de RAM sous l'environnement *Windows XP*. Nous avons implémenté les algorithmes *Concept1* et *Concept2* en Java, alors que *Ganter*, *Close By One* et *Bordat* sont écrits en C++. Les expérimentations ont été menées sur des données aléatoires et d'autres réelles. Pour ce faire, nous avons utilisé un générateur aléatoire de relations binaires ainsi qu'une version binarisée de bases réelles présentant des données de types différents accessibles au public depuis le site UCI Repository¹. Ces bases sont typiquement connues comme des bases denses.

4.1. Bases aléatoires

La rapidité d'exécution d'un algorithme de génération de concepts se trouve influencée par trois paramètres : le nombre d'objets de la relation, le nombre d'attributs de la relation et la densité de la relation qui représente la corrélation entre les attributs et les objets. Ce nombre moyen d'attributs par objet peut être ramené à un pourcentage de 1 sur l'ensemble de la relation. Nous avons construit nos jeux d'essai suivant cette optique. Nous avons essayé de faire ressortir l'influence de la variation de chaque paramètre sur le temps d'exécution. Pour chaque série de tests, nous avons fixé deux des paramètres et fait évoluer le troisième. Nous avons également pris en compte le fait que dans les bases de données réelles, le nombre d'objets est beaucoup plus important que le nombre d'attributs. Chaque point des courbes de la section 4.2 représente une moyenne des valeurs de 11 essais réalisés pour chaque base représentée par une relation binaire.

Série de Test 1

Pour cette série de tests, nous avons utilisé une base ayant 20 attributs et un pourcentage de densité de 40%. Nous avons étudié l'influence de la variation du nombre d'objets de cette base sur le temps d'exécution nécessaire pour l'extraction de l'ensemble des concepts pour chacun des algorithmes. Le tableau 1 présente les valeurs récupérées et la figure 1 décrit l'allure des courbes de variation du temps d'exécution en fonction de la variation du nombre d'objets.

1. <http://mlearn.ics.uci.edu/databases/>

# Objets	Concept1	Concept2	Close By One	Ganter	Bordat
50	0,57	0,70	0,04	0,04	0,06
100	1,84	2,12	0,25	0,29	0,25
200	4,50	5,88	1,10	2,00	0,60
300	7,34	10,03	2,81	6,34	0,96
400	10,59	15,16	5,46	14,51	1,46
500	14,98	21,69	9,06	27,79	1,96
1000	37,25	55,08	43,53	205,11	4,96
1500	59,12	95,76	106,31	654,98	8,32
2000	87,39	141,25	204,70	1491,93	13,12

Tableau 1. Variation du temps d'exécution en fonction du nombre d'objets

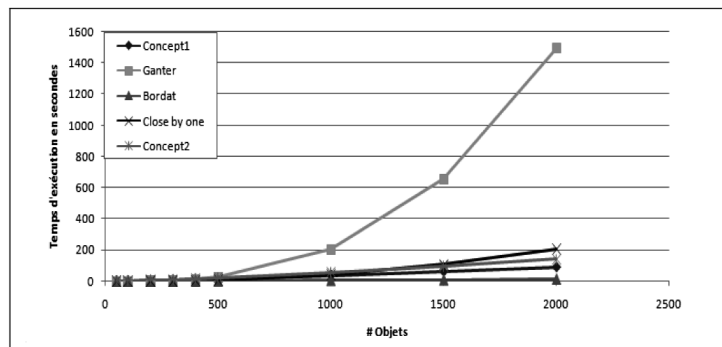


Figure 1. Variation du temps d'exécution en fonction du nombre d'objets

Nous avons constaté que, pour un nombre d'objets supérieur à 500, l'algorithme de *Ganter* prend une allure exponentielle. Les algorithmes *Bordat*, *Close By One*, *Concept1* et *Concept2* présentent la même allure et des valeurs très rapprochées avec un avantage pour celui de *Bordat*. Pour un nombre d'objets supérieur à 1500, les algorithmes *Concept1* et *Concept2*, basés sur la théorie des graphes, présentent de meilleurs résultats que l'algorithme *Close By One*. En effet, ces deux algorithmes récursifs utilisent les attributs comme intensions des concepts générés dans un parcours en profondeur. La complexité en temps d'exécution est influencée par la taille de la pile d'exécution.

Série de Test 2

Pour cette série de tests, nous avons utilisé une base ayant 1000 objets et 40% de densité. La variation du temps d'exécution en fonction du nombre d'attributs, est décrite par le tableau 2 et représentée par la figure 2.

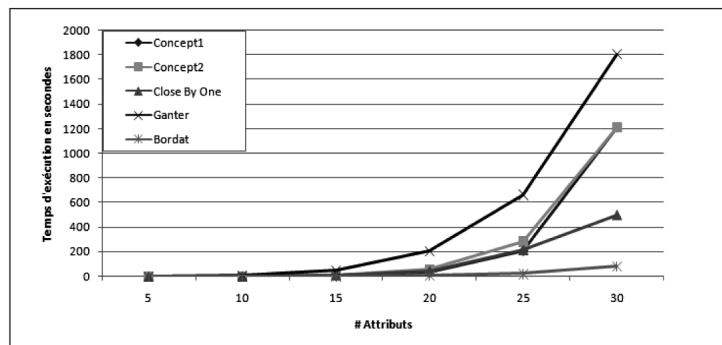


Figure 2. Variation du temps d'exécution en fonction du nombre d'attributs

# Attributs	Concept1	Concept2	Close By One	Ganter	Bordat
5	0,18	0,33	0,06	0,42	0,06
10	0,68	1,50	0,93	6,23	0,18
15	4,72	9,30	9,45	50,96	1,03
20	35,24	54,93	44,28	206,04	5,01
25	208,21	285,14	214,94	664,32	20,12

Tableau 2. Variation du temps d'exécution en fonction du nombre d'attributs

Au delà de 15 attributs, les algorithmes *Ganter*, *Close By One*, *Concept1* et *Concept2* présentent une allure exponentielle, par contre l'algorithme de *Bordat* donne les meilleurs résultats. Les algorithmes *Concept1* et *Close By One* donnent des résultats très proches ce qui rend leur courbes confondues. Ils présentent le même comportement vis-à-vis de la variation du nombre d'attributs.

Série de Test 3

Dans cette série, nous avons utilisé une base ayant 1000 objets et 20 attributs. Nous avons étudié l'influence de la variation de la densité de cette base sur le temps d'exécution nécessaire pour l'extraction de l'ensemble des concepts. Le tableau 3 présente les valeurs obtenues et la figure 3 décrit l'allure des courbes de variation du temps d'exécution en fonction de la variation de la densité.

Densité	Concept1	Concept2	Close By One	Ganter	Bordat
10%	0,99	2,96	0,59	3,23	0,08
20%	2,88	7,36	2,71	15,26	0,31
30%	9,77	19,65	11,36	57,20	1,20
40%	35,39	55,07	44,97	212,55	5,01
50%	126,74	157,81	163,39	748,50	23,04
60%	425,49	431,42	573,17	2374,36	107,56
70%	1356,73	1067,25	1413,70	6595,01	483,57
80%	3670,90	2018,76	1848,68	//	1409,37
85%	5888,98	2752,48	1797,71	//	2304,70

// Pas de résultat au bout de 24h

Tableau 3. Variation du temps d'exécution en fonction de la densité

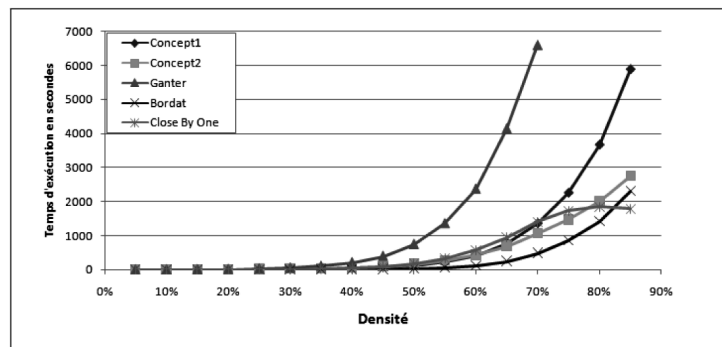


Figure 3. Variation du temps d'exécution en fonction de la densité

Une valeur élevée de la densité de la relation a une grande influence sur le comportement de l'algorithme de *Bordat* qui a présenté les meilleurs résultats au cours des deux séries de tests précédentes. En effet, d'après les courbes de la figure 3 nous remarquons que cet algorithme présente la même allure que celle de l'algorithme *Concept1*. Tous deux

sont moins performants que l'algorithme *Concept2* pour une valeur de densité supérieure à 55%. L'algorithme de *Ganter* est le plus sensible à la variation de la densité. Sa courbe représentative prend une allure exponentielle pour une valeur de densité supérieure à 40%.

Les séries de tests réalisées sur des relations binaires, générées aléatoirement, sont en faveur de l'algorithme de *Bordat* pour des nombres élevés d'attributs et en faveur de l'algorithme *Concept1* pour des valeurs élevées de densité. Pour des valeurs élevées de nombre d'objets, les algorithmes *Bordat*, *Close By One*, *Concept1* et *Concept2* donnent des résultats similaires et des courbes de même allure. L'algorithme de *Ganter* présente toujours une plus grande sensibilité à des valeurs élevées des trois paramètres étudiés : densité, nombre d'objets et nombre d'attributs. Ces remarques ont été par la suite prouvées sur des données réelles.

4.2. Bases réelles

Dans cette section, nous allons étudier le comportement des 5 algorithmes vis à vis des bases réelles de l'UCI repository, qui représentent des bases denses. Les caractéristiques de ces bases ainsi que les résultats des tests effectués sont présentés par le tableau suivant. Les résultats sur des bases réelles présentés dans le tableau 4 et la figure 4 confirment

	Base	# Objets	# Attributs	Concept1	Concept2	Ganter	Close By One	Bordat
DB1	Monks-1	432	19	2,87	5,29	5,23	2,90	0,32
DB2	Monks-3	432	19	2,66	4,95	4,87	2,56	0,34
DB3	House-votes-84	435	18	11,30	8,02	9,23	3,32	1,90
DB4	Balance-scale	625	23	2,28	5,65	4,62	2,01	0,17
DB5	Tic-Tac-Toe	958	29	122,37	140,23	208,50	10,30	6,11
DB6	Car	1728	21	10,26	24,62	135,14	54,20	0,73
DB7	Nursery	12960	31	1329,92	2401,09	//	//	//

//. Pas de résultat au bout de 24h

Tableau 4. Temps de génération des concepts pour des bases réelles

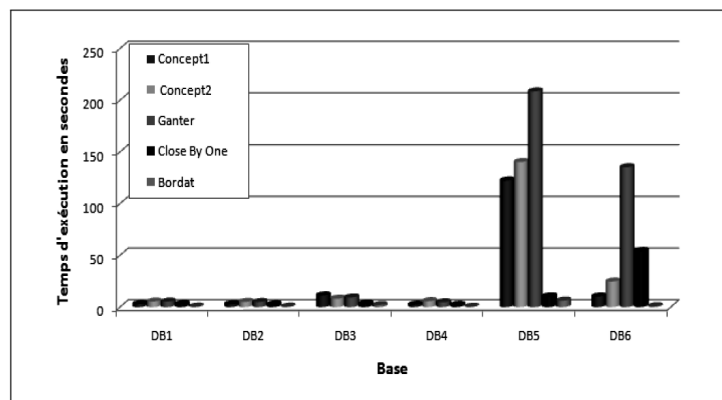


Figure 4. Variation du temps d'exécution en fonction du nombre d'objets pour des bases réelles

les résultats trouvés précédemment pour les bases synthétiques. L'algorithme de *Bordat* présente les meilleurs résultats suivi des algorithmes *Concept1* et *Concept2* pour les bases ayant un nombre d'objets dépassant 1200. Pour ces valeurs, l'algorithme *Close By One*

perd son avantage par rapport aux deux algorithmes basés sur la théorie des graphes. Pour la base *Nursery* ayant un nombre d'objets égal à 12960 et un nombre d'attributs de 31 nous n'avons pas pu générer la liste des concepts au bout de 24h utilisant les algorithmes *Ganter*, *Close By One* et *Bordat*. Par contre, nous avons pu récupérer la liste des concepts avec les algorithmes *Concept1* et *Concept2*. L'approche algorithmique se basant sur les graphes présente alors de meilleurs résultats dans le cas des bases denses ayant un nombre d'objets et un nombre d'attributs très importants.

5. Conclusion

Le choix d'un algorithme de génération de concepts est élément fondamental dans un processus de fouille de données basé sur l'analyse de concepts formels. Pour cela, nous avons, dans cet article, comparé plusieurs algorithmes de construction d'un treillis de concepts.

Les résultats des tests que nous avons réalisés montrent que l'algorithme de *Bordat* donne des meilleures performances pour des bases éparses de petites tailles. Pour des bases denses, les algorithmes *Concept1* et *Concept2* s'avèrent les plus adéquats, car ils donnent de meilleurs résultats aussi bien pour la génération de concepts que pour la détermination des éléments du diagramme de Hasse. Les algorithmes basés sur les graphes sont plus adaptés à des bases de données denses de grande taille.

L'algorithme de *Ganter* lui s'avère plus en adéquation avec des bases de petites tailles et de densités faibles. Mais, il donne de mauvaises performances pour les valeurs élevées de l'un des paramètres suivants : nombre d'objets, nombre d'attributs ou densité.

6. Bibliographie

- [1] Y. BASTIDE, N. PASQUIER, « Mining frequent patterns with counting inference », *Conference on Knowledge Discovery and Data Mining, Boston, Massachusetts USA*, 2000.
- [2] S. KUZNETSOV, S. OBIEDKOV, « Comparing Performance of Algorithms for Generating Concept Lattices », *Journal of Experimental Theoretical Artificial Intelligence*, April 2002.
- [3] B. GANTER AND R. WILLE, « Formal Concept Analysis », *Springer-Verlag*, 1999
- [4] A. SIGAYRET, « Data Mining : une approche par les graphes », *Thèse Ecole Doctorale Sciences pour l'ingénieur de Clermont Ferrand, Université Blaise Pascal- Clermont II*, Décembre 2002.
- [5] C. EL HOG, « Génération des concepts : approche basée sur les graphes », *Mastère Ecole Doctorale en Informatique, Faculté des Sciences de Tunis*, Janvier 2008.
- [6] J.P. BORDAT, « Calcul pratique du treillis de galois d'une correspondance », *Mathématiques et Sciences Humaines, tome 96*, 1986.
- [7] B. GANTER, « Two basic algorithms in concept analysis », *Preprint 831, Technische Hochschule Darmstadt*, 1984.
- [8] A. BERRY, J.P. BORDAT, A. SIGAYRET, « A local approach to concept generation », January 2006.
- [9] G. BRIKHOFF, « Lattice Theory », *American Society Providence RI*, 1976.
- [10] E. M. NGUIFO, « Fouille de Données et Treillis de Galois », *Tutoriel EGC'04 - Clermont-Ferrand*, 2004.