Université d'Angers

M2 Intelligence Décisionnelle

Recherche de motifs fréquents par algorithmes évolutionnaires

Auteur :
Ugo Rayer

Encadrants :
Benoit Da Mota
Béatrice Duval
David Lesaint

20 avril 2017

Remerciements

Avant toute chose, je tiens à remercier l'ensemble des personnes qui ont participé, de près ou de loin, à la réalisation de ce stage et à l'écriture de ce rapport.

Des remerciements spéciaux sont adressés d'une part au Laboratoire d'Etude et de Recherche en Informatique d'Angers pour son accueil et d'autre part au projet GRIOTE de la région Pays de la Loire qui a financé cette étude. Ensuite, je tiens à remercier chaleureusement Madame Duval et Messieurs Da Mota et Lesaint pour la qualité de leur encadrement et les différentes remarques qu'ils m'ont prodiguées pendant ces quelques mois.

Enfin, un grand merci à Josépha pour ses précieux conseils d'écriture malgré l'incompréhension générale des sujets abordés.

Table des matières

Chapitre 1

Introduction

Le calcul des motifs fréquents est une notion essentielle dans de nombreux domaines liés à la découverte et l'extraction de connaissances. Initiallement introduit par Agrawal et al. dans [**?**], ces motifs étaient alors utilisés pour l'établissement de règles d'associations visant à caractériser les habitudes d'achats de clients d'un supermarché. Par exemple, une règle de la forme "Pain & Beurre $\Rightarrow$ Jambon (75%)" signifie que 3 clients sur 4 achetant du pain et du beurre achètent également du jambon. Le calcul de telles règles se fait en deux étapes, dont la principale (i.e. disposant de la plus grosse complexité calculatoire) correspond au calcul des motifs fréquents. Depuis son introduction, le problème du calcul des motifs fréquents a été très largement étudié et appliqué à de nombreux domaines comme en bio-informatique, en cybersécurité et bien évidemment en marketing.

L'avènement de l'ère du Big Data a fait rentrer le problème de calcul des motifs fréquents dans une nouvelle dimension. En effet, le volume de données produites dans tous les domaines a cru de manière vertigineuse ces dernières années, rendant l'extraction de connaissances d'autant plus nécessaire et délicate. De fait, la problématique du passage à l'échelle des méthodes exactes est devenue primordiale. Bien que divers efforts en ce sens aient été faits au travers de nombreuses publications, ils se concentrent généralement sur l'optimisation et la parallélisation des méthodes existantes.

Les algorithmes évolutionnaires font partie des techniques de résolution de problèmes combinatoires appelées méta-heuristiques. Les méta-heuristiques regroupent un ensemble de méthodes approchées à la résolution de problème combinatoire. Elles sont naturellement envisagée lorsque la complexité du problème étudié ne permet pas l'usage de méthodes exactes (aussi bien en temps qu'en espace). Le principe des algorithmes évolutionnaires est de manipuler un ensemble d'individus représentant chacun une solution au problème étudié. A chaque itération, certains individus sont modifiés via des opérateurs de croisement et de mutation. Chaque individu est évalué au regard d'une fonction à optimiser dépendant du problème étudié.

Ainsi, nous formaliserons le problème de calcul de motifs fréquents dans la section suivante avant d'effectuer un état de l'art des méthodes existantes en

section 3. Le chapitre 4 sera dédié à la présentation de notre méthode dont nous présenterons les résultats en section 5. Le dernier chapitre sera consacré à la conclusion de cette étude et à une ouverture vers de futurs travaux.

Chapitre 2

Le problème du calcul motifs fréquents

2.1 Définition du problème

La définition la plus courante du problème de calcul des motifs fréquents se fait par la théorie des ensembles. Nous verrons cependant dans le chapitre suivant qu'il peut également être décrit par la théorie des graphes. Nous présenterons dans un premier temps un cadre formel nécessaire à la définition du problème que nous illustrerons ensuite. Enfin, nous introduirons quelques propriétés dérivés de la définition du problème.

2.1.1 Formalisation

Soit $\mathcal{I}$ un ensemble de *symboles* appelées **items**. Quelque soit $I \subseteq \mathcal{I}$, I est un *motif* appelé **itemset**.

Soit $\mathcal{T} = \{ t_1, \ldots, t_n \}$ un ensemble de **transactions**. Chaque élément t_i est un couple $< tid, I >$ où tid est l'identifiant de la transaction et $I \subseteq \mathcal{I}$. $\mathcal{T}$ est communément appelé **base de transactions**.

Pour tout itemset $I \subseteq \mathcal{I}$ la **couverture** de I par $\mathcal{T}$ est définie par :

$$\mathbf{cover}_{\mathcal{T}}(I) = \{ t \in \mathcal{T} |\ I \subseteq t \}$$

La cardinalité de la couverture d'un itemset I par $\mathcal{T}$

$$\mathbf{sup}_{\mathcal{T}}(I) = |\ \mathbf{cover}_{\mathcal{T}}(I)\ |$$

est appelée **support** de I. Etant donné un support minimal $minsup$ l'ensemble des **itemsets** (i.e. motifs) **fréquents** est défini par :

$$\mathbf{F}_{\mathcal{T},minsup} = \{ I \subseteq \mathcal{I} |\ sup_{\mathcal{T}}(I) \geq minsup \}$$

Le problème du **calcul des itemsets fréquents** (**FIM** - *Frequent Itemsets Mining*) est, étant donné une base de transactions $\mathcal{T}$ et un support minimal *minsup* de calculer l'ensemble F des itemsets fréquents. Comme F. Boden le remarque dans [**?**], bien qu'historiquement définie comme une valeur relative et donc asujettie à un support minimal défini dans l'intervalle [0,1], le support est de nos jours mesuré de manière absolue. Si nécessaire, nous y ferons référence sous la notion de fréquence :

$$\mathbf{Freq}_{\mathcal{T}}(I) = \frac{|\ \mathbf{cover}_{\mathcal{T}}(I)|}{|\mathcal{T}|}$$

avec *minfreq* simplement définie par $\frac{minsup}{|\mathcal{T}|}$.

2.1.2 Exemple

Situons nous dans le contexte de la définition historique de problème et considérons la base de transactions suivante (que nous conserverons tout au long de cet article). Le tableau 1 décrit chaque transaction par : un identifiant, une liste d'objets achetés et la liste des items fréquents vis à vis d'un support minimal de 3.

ID transaction	Objets achetés	Items Fréquents Ordonnés
100	f, c, a, d, g, i m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

TABLE 2.1 – Base de transactions exemple

Une fois calculé, l'ensemble des itemsets fréquents vis à vis de ce jeu de données $\mathcal{T}$ est l'ensemble $\mathbf{F}_{\mathcal{T},3}$ = { (f :4), (c :4), (a :3), (b :3), (m :3), (p :3), (fc :3), (fa :3), (fm :3), (cm :3), (cp :3), (ca :3), (am :3), (fca :3), (fcm :3), (fam :3), (cam :3), (fcam :3) }.

2.1.3 Complexité et Propriétés

La complexité du problème vient d'une part du nombre d'itemsets à considérer en fonction du nombre d'objets et d'autre part de nombre de transactions dans la base. En effet, soit n objets fréquents dans la base, il y a alors 2^n itemsets possibles. D'autre part, le calcul du support d'un itemset se fait au travers de l'ensemble des transactions. L'efficacité d'une méthode à résoudre ce problème se fera donc par sa capacité à explorer intelligement l'espace des 2^n itemsets et par son efficacité à calculer le support d'un itemset vis à vis de $|\mathcal{T}|$.
Différents théorèmes et propriétés issus de l'étude de ce problème sont utilisés dans les méthodes proposées pour le résoudre. Nous présentons les propriétés liées à la monotonie du support d'un ensemble et renvoyons le lecteur vers [**?**] et [**?**] pour plus de détails.

Monotonie du support. Soit une base de transactions $\mathcal{T} sur \mathcal{I}$ et soient X, $Y \subseteq \mathcal{I}$ deux itemsets. Alors,

$$X \subseteq Y \Rightarrow \mathbf{sup}_{\mathcal{T}}(Y) \leq \mathbf{sup}_{\mathcal{T}}(X)$$

Cette propriété nous permet de dire que si un k-itemset X (i.e. un itemset comprenant k objets) est fréquent, alors l'ensemble Y des k-1-itemsets $\subset X$ est fréquent. Par exemple, (fca :3) est fréquent, de même que (fc :3), (fa :3) et (ca :3). Nous pouvons de manière duale dire que si un k- itemset X est non-fréquent, alors aucun des k+1-itemset Y tel que $X \subset Y$ n'est fréquent. Ces deux observations sont à la base des différents sens de parcours de l'espace de recherche des 2^n itemsets possibles dans la plupart des algorithmes.

2.2 Problème des itemsets clos maximaux

Afin de réduire l'espace de recherche, il a été proposé de contraindre la recherche à l'ensemble des itemsets clos. Un k-itemset$X$ fréquent est clos si $\mathbf{sup}_{\mathcal{T}}(X) > \mathbf{sup}_{\mathcal{T}}(Y)$ $\forall$ Y tel que $X \subset Y$. Un itemset clos est maximal si aucun de ses surensembles n'est fréquent. Ainsi dans notre exemple, l'ensemble des itemsets clos est $\mathbf{C}_{\mathcal{T},3}$ = { (f :4), (c :4), (b :3), (cp :3), (fcam :3) } et l'ensemble des itemsets clos maximaux est $\mathbf{MC}_{\mathcal{T},3}$ = $\mathbf{C}_{\mathcal{T},3}-$ { (f :4), (c :4) }. La figure suivante représente le treillis complet des différents itemsets possibles sur les objets fréquents. Y figurent d'une part les itemsets fréquents en vert, d'autre part les itemsets clos en jaune et enfin les itemsets clos maximaux en rouge. Pour plus de lisibilité, les itemsets non-fréquents ne figurent pas dans le treillis.

Zaki et al. et Pas. et al. prouvent dans [**?**] et [**?**] que l'intégralité des itemsets fréquents peut être dérivée de l'ensemble des itemsets clos maximaux. Ils proposent alors d'adapter les méthodes existantes pour le calcul des itemsets clos maximaux.

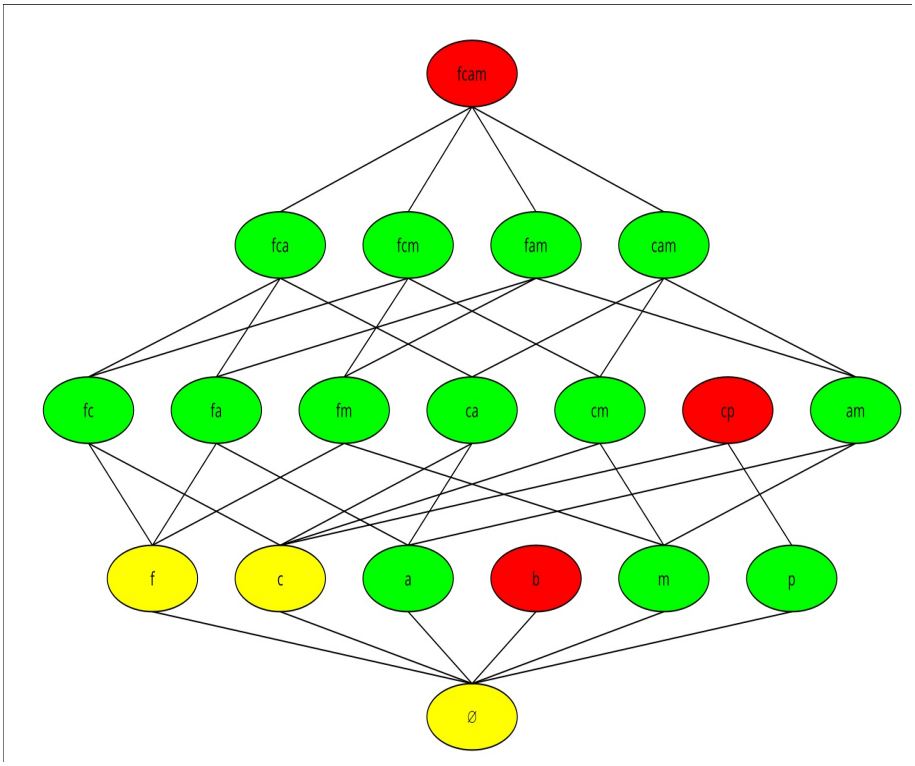

FIGURE 2.1 – Treillis des itemsets fréquents, clos et maximaux de notre exemple

2.3 Itemset et classification

Dans cette section nous redéfinissons le problème à considérer. Nous l'étudierons ensuite sous l'aspect d'un problème combinatoire solvable par recherche locale. Enfin nous l'aborderons via une approche évolutionnaire.

2.3.1 Définition du problème

Soit $\mathcal{T}$ une base de transactions définie sur $\mathcal{N}$ items. En plus d'un identifiant et de ses items, chaque transaction possède une classe (positive ou négative). Le problème consiste à trouver parmi $\mathcal{T}$ un itemset X tel que le support de X dans la classe positive soit supérieur à un seuil minimal *minsup* **et** que le support de X dans la classe négative soit inférieur à un seuil maximal *maxsup*.

Formellement, soit X = $\{i_1, ..., i_k\}$ un k-itemset quelconque. Alors, soient sup_+(X) et sup_-(X) les deux fonctions suivantes :

$$sup_+(\mathrm{X}) = \{t_j \mid t_j \in \mathcal{T} \text{ et } \forall i \in X, t_{i,j} = 1 \text{ et } t_j \in C^+\}$$
$$sup_-(\mathrm{X}) = \{t_j \mid t_j \in \mathcal{T} \text{ et } \forall i \in X, t_{i,j} = 1 \text{ et } t_j \in C^-\}$$

X est un itemset solution si et seulement si les propriétés suivantes sont vérifiées :

$$|sup_+(\mathrm{X})| > minsup$$
$$|sup_-(\mathrm{X})| < maxsup$$

2.3.2 Représentation et évaluation d'une solution

Le problème étant de trouver un itemset maximisant la couverture de la classe positive tout en minimisant son support dans la classe négative, une solution du problème est simplement un bitset S de $\mathcal{N}$ bits tel que $\forall i \in [1..\mathcal{N}] S_i = 1$ si l'item i appartient à la solution, $S_i = 0$ sinon.
L'évaluation d'une solution doit permettre de quantifier la corrélation des différents items dans leur pouvoir de classification de la base de transactions. Diverses mesures existent et ont été proposées et largement étudiées dans la littérature (citation TAN02, GEN06, ...).
Dans une problématique de classification, de nombreuses mesures sont basées sur les 4 ensembles décrivant la matrice de confusion des performances d'un classifieur :

- Vrai positif (TP) : $\forall i \in [1..\mathcal{N}]\ S_i = 1 et t_i = 1\ \&\ t \in C^+$
- Faux Positif (FP) : $\exists i \in [1..\mathcal{N}]$ tel que $S_i = 1, t_i = 0\ \&\ t \in C^+$
- Vrai Négatif (TN) : $\exists i \in [1..\mathcal{N}]$ tel que $S_i = 1, t_i = 0\ \&\ t \in C^-$
- Faux Négatif (FN) : $\forall i \in [1..\mathcal{N}]\ S_i = 1, t_i = 1\ \&\ t \in C^-$

Le tableau suivant récapitule l'appartenance d'une transaction t à un des quatre ensembles :

	$S \subseteq t$	$S \not\subseteq t$
Classe $t = 1$	$t \in TP$	$t \in FP$
Classe $t = 0$	$t \in FN$	$t \in TN$

La précision d'une telle classification correspond au rapport entre le nombre de réponses pertinentes données (TP) et le nombre de réponses données (TP+FP). Le rappel correspond quant à lui au rapport entre le nombre de réponses pertinentes données (TP) et le nombre de réponses pertinantes existant (TP+FN).

$$\text{- Precision} = \frac{TP}{TP+FP}$$
$$\text{- Rappel} = \frac{TP}{TP+FN}$$

La *F Mesure* permet d'aggréger rappel et précision en accordant un poids équivalent à chaque mesure. Cette une version simplifié de la F_β *Mesure* définie par la relation suivante :

$$\text{- } F_\beta = (1+\beta^2) * \frac{Precision*Rappel}{\beta^2*Precision+Rappel}$$

Dans notre cas, où rappel et précision sont équitablement pondérés, la mesure peut se simplifier comme suit :

$$\text{- } F_1 = \frac{2*TP}{2*TP+FN+FP}$$

Toutefois, cette mesure ne permet pas d'introduire les deux seuils définissant les contraintes de notre problème. De plus, le problème que nous tentons de résoudre peut être vu comme un problème d'optimisation sous contraintes et donc défini comme suit :

$$\begin{gathered}\text{Maximize } \sum (X_i = 1)\ \forall i \in \mathcal{I} \text{ s.c.} \\ VP(X) \geq minsup \\ FN(X) \leq maxsup\end{gathered}$$

Enfin, comme le rappellent C. Dhaenens et L. Jourdan dans *Metaheuristics for Big Data*, le choix dans la mesure de qualité est fortement corrélée au contexte d'application et peut donc être utilisée comme fonction objectif par une méthode d'optimisation. Dans ce sens et traitant le problème d'un point de vue générique, nous proposons donc d'évaluer une solution X de k items de manière générique en aggrégeant simplement notre objectif avec nos deux contraintes équitablement pondérées via la formule suivante :

$$\mathcal{F}(X) = k * (\frac{1}{2} * \frac{TP}{minsup} + \frac{1}{2} * \frac{maxsup}{FN}\ )$$

Le calcul de cette évaluation nécessite un scan complet du jeux de données afin de calculer la cardinalité des ensembles TP et FN intervenant dans la fonction $\mathcal{F}$. Nous verrons ensuite que nous pourrons profiter de ce scan pour la définition du voisinage.

2.3.3 Relation de voisinage

Après avoir défini la fonction d'évaluation d'un individu, il est nécessaire pour pouvoir mettre en place ni'mporte quel mécanisme de recherche de définir la relation de voisinage qui va nous permettre de nous déplacer au sein de l'espace de recherche. Différents points sont à considérer pour la définition d'un voisinage pertinent. En premier lieu, il est important de noter qu'il est difficile de défnir un voisinage pertinent dans le cadre d'une approche de type générique

(sans information du contexte). Ensuite, en raison de la complexité de la méthode d'évaluation (nécessitant un scan complet du jeu de données) il est important de considérer la taille du voisinage comme un facteur important. En effet, un voisinage trop grand engendrera un coût calculatoire trop important et par conséquent une baisse de performance, alors qu'un voisinage trop petit ne permettra pas un déplacement efficace dans l'espace de recherche.

Pour l'ensemble de ces raisons et pour ne pas bloquer sur une relation de voisinage informée, nous nous contenterons pour l'instant d'utiliser un voisinage simple et relativement restreint. De fait, une solution S' appartiendra au voisinage de la solution courante S s'il est différent de S à un item prêt. Formellement, le voisinage d'une solution est défini comme suit :

$$\mathcal{N}(S) = \{S' \text{ tq } |S'| = |S|+1 \text{ et } \forall i \in S, i \in S' \} \cup \{S' \text{ tq } |S'| = |S|-1 \text{ et } \forall i \in S', i \in S\}$$

Ce voisinage est de taille fixe égale à la taille de $\mathcal{I}$. En effet, chaque item a soit la possibilité d'être ajouté ou bien d'être enlevé de la solution courante. Nous pourrons éventuellement contraindre plus spécifiquement ce voisinage en tentant de spécifier les items pouvant être ajoutés/supprimés en fonction d'informations extraites lors de l'évaluation de la solution courante.

2.3.4 Recherche locale

Le premier algorithme que nous avons implémenté est une recherche tabou itérée. Plus précisément, nous commençons par choisir une solution initiale aléatoire puis nous effectuons de manière itérée plusieurs recherches tabous jusqu'à ce qu'un des deux critères d'arrêts soit atteint. Le premier est un temps d'exécution alloué à l'algorithme et le second est un nombre maximale de recherches n'aboutissant pas à une amélioration de la meilleure solution trouvée jusqu'ici.

A chaque itération, la solution courante est remplacée par la meilleure solution voisine, i.e. celle qui maximise la fonction d'évaluation. Il est par conséquent possible qu'aucun voisin n'améliore celle-ci, permettant par conséquent de sortir d'un optimum local. La mécanique tabou consiste ici à interdire à un item ayant été ajouté ou supprimé récemment de subir l'opération inverse pendant un nombre fixé d'itérations appelé *tabu tenure*. Un critère d'aspiration est toutefois indispensable dans le cas où un item interdit améliore la meilleure

solution globale. L'algorithme est présenté ci-après.

Algorithm 1: Tabu Search

Input : TDB $\mathcal{T}$, tabu tenure tt, total time TT, nombre max sans amélioration β
Output: Solution S_b

```
1  begin
2      Solution S = randomInit();
3      Solution S_b = S;
4      d = 0; iteration = 0;
5      repeat
6          Solution S' = BestNeighboor(S);
7          MaJTabuList(S', S, tt + iteration);
8          S ⟵ S';
9          if Evaluation(S') > Evaluation(S_b) then
10             S_b = S;
11             d = 0;
12         end
13         else
14             d = d + 1;
15         end
16         iteration = iteration + 1;
17     until d == β // timeLeft == 0;
18 end
```

tid	A	B	C	D	E	Classe
1	1	0	1	1	0	+
2	0	1	1	0	1	+
3	1	0	1	1	0	+
4	1	0	1	1	1	+
5	0	1	0	0	1	-
6	1	0	1	1	0	-
7	0	1	1	0	1	-
8	1	0	0	1	1	-