Extraction des itemsets fréquents à partir de données évidentielles : application à une base de données éducationnelles

Mohamed Anis Bach Tobji *, Boutheina Ben Yaghlane**

Laboratoire LARODEC, Université de Tunis, Institut Supérieur de Gestion
41 avenue de la Liberté, Cité Bouchoucha, Le Bardo 2000, Tunisie
*anis.bach@isg.rnu.tn **boutheina.yaghlane@ihec.rnu.tn

Résumé. Dans cet article, nous étudions le problème de l'extraction des itemsets fréquents (EIF) à partir de données imparfaites, et plus particulièrement ce qu'on appelle désormais les données *évidentielles*. Une base de données évidentielle stocke en effet des données dont l'imperfection est modélisée via la théorie de l'évidence. Nous introduisons une nouvelle approche d'EIF qui se base sur une structure de données de type arbre. Cette structure est adaptée à la nature complexe des données. La technique que nous avons conçue, génère jusqu'à 50% de la totalité des itemsets fréquents lors du premier parcours de l'arbre. Elle a été appliquée sur des bases de données synthétiques ainsi que sur une base de données éducationnelles. Les expérimentations menées sur la nouvelle méthode, montrent qu'elle est plus performante en terme de temps d'exécution en comparaison avec les méthodes existantes d'EIF.

1 Introduction

L'extraction des itemsets fréquents (EIF) à partir des données (Agrawal et al., 1993) a reçu une attention particulière de la part des chercheurs puisqu'elle constitue l'étape coeur de plusieurs méthodes de fouille de données à l'instar de l'extraction des règles d'associations, la construction de certains classifieurs, l'extraction des séries temporelles et d'autres techniques de fouille de motifs (dits aussi patterns). Néanmoins, la majorité des techniques d'EIF (Agrawal et Srikant, 1994; Han et al., 2000; Lucchese et al., 2003; Zhao et Bhowmick, 2003) ne prennent pas en considération l'aspect imparfait des données générées par les applications du monde réel. Ce problème d'imperfection touche en effet plusieurs systèmes d'informations. En sciences expérimentales, les chercheurs se basent sur des expérimentations dont les données générées sont sauvegardées et par la suite traitées. Ces valeurs observées ou mesurées sont la plupart du temps imprécises ou incertaines (Kwan et al., 1996). En médecine, les docteurs se trouvent souvent dans l'obligation d'émettre un diagnostic en présence de symptômes imprécis voire incertains (Konias et al., 2005). Les données générées par certains systèmes à base de capteurs sont aussi imparfaites. Les capteurs d'un système unique peuvent produire des informations à différents niveaux de confiance. En plus, chacun d'eux peut produire une information incertaine, imprécise ou incomplète (Vaughn et al., 2005). L'imperfection des données,

et donc leur complexité, ne doit pas être un obstacle par rapport à l'extraction de connaissances et c'est d'ailleurs l'objet de ce travail.

Récemment, l'EIF à partir de bases de données imparfaites commence à susciter de l'intérêt dans la communauté de fouille de données et certains travaux sont apparus dans ce sens. Dans la plupart de ces travaux, l'imperfection des données est modélisée via la théorie des probabilités (Chui et al., 2007; Leung et al., 2007), la théorie des possibilités (Djouadi et al., 2007) ou les ensembles flous (Chen et Wei, 2002; Dubois et al., 2006; Wang et al., 2005). En dépit de l'importance de la théorie d'évidence (nommée aussi théorie de Dempster-Shafer) (Shafer, 1976), l'état de l'art en EIF à partir de bases de données évidentielles est relativement pauvre (Hewawasam et al., 2007).

Dans cet article, nous présentons notre technique d'EIF à partir de bases de données à unique attribut imparfait. Cette technique a été testée sur plusieurs bases de données générées synthétiquement, ainsi que sur une base de données éducationnelles réelles. Cette dernière est le résultat d'une enquête qui a été menée sur la qualité de la formation en informatique dans les universités tunisiennes. . Les expérimentations effectuées sur notre technique ont montré des résultats encourageants au niveau de la performance, et ce relativement aux techniques existantes (Hewawasam et al., 2007; Bach Tobji et al., 2008b). Il est à signaler que cet article est une extension de (Bach Tobji et al., 2008a) appliquée aux données éducationnelles.

Le reste de l'article est organisé comme suit : Dans la section 2, nous présentons les concepts basiques de la théorie de Dempster-Shafer. La section 3 est une présentation des bases de données évidentielles où nous introduisons la base de données éducationnelles sur laquelle nous allons appliquer notre méthode d'EIF. Dans la section 4, nous présentons les principaux concepts relatifs au modèle d'EIF à partir de données évidentielles, suivis par la méthode que nous avons conçue pour résoudre ce problème. Dans les sections 4.2.1 et 4.2.2, nous présentons respectivement la structure de données utilisées pour représenter les données à fouiller, ainsi que la technique introduite pour en extraire les itemsets fréquents. Les expérimentations que nous avons menées aussi bien sur des données synthétiques que sur des données éducationnelles sont présentées dans la section 5. Finalement, la conclusion de notre travail est donnée dans la section 6.

2 La théorie de Dempster-Shafer

2.1 Les concepts de base

La *théorie de Dempster-Shafer*, appelée aussi *théorie de l'évidence*, a été introduite par Dempster dans (Dempster, 1967) et formalisée mathématiquement par Shafer dans (Shafer, 1976). La théorie de Dempster-Shafer est souvent décrite comme une généralisation de la théorie bayésienne. Nous présentons dans cette section les concepts formels de cette théorie.

Soit $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$ un ensemble fini non-vide incluant n hypothèses exhaustives et mutuellement exclusives. Θ est appelée *cadre de discernement* ou *cadre d'intérêt* du problème posé. Le *référentiel de définition* utilisé pour évaluer la véracité d'une proposition est constitué de tous les sous-ensembles possibles de Θ, soit l'ensemble 2^{Θ}. La *fonction de masse élémentaire* m d'une hypothèse $X \subseteq \Theta$, notée $m(X)$, représente la partie du degré de croyance placée sur X et qui n'a pas été distribuée aux sous-ensembles de X. La masse de croyance

élémentaire, notée *bba* est définie par :

$$m : 2^{\Theta} \rightarrow [0, 1]$$

m satisfait deux conditions qui trouvent leur correspondance dans la théorie des probabilités à savoir :

La masse de l'ensemble vide est nulle :

$$m(\emptyset) = 0$$

La somme des masses de toutes les hypothèses correspond à l'unité :

$$\sum_{X \subseteq \Theta} m(X) = 1$$

Les sous-ensembles de Θ sur lesquels sont placées des masses strictement positives, sont appelés *éléments focaux*, leur ensemble est noté Foc. Le triplet $\{\Theta, Foc, m\}$ est appelé *corps d'évidence* noté *BoE*. La *fonction de croyance (bel)* est définie à partir de la fonction de masse m, dans la mesure où la croyance d'un événement $A \subseteq \Theta$ reflète la croyance totale assignée à A, c.-à-d., la somme des masses de tous les sous-ensembles de A.

$$bel(A) = \sum_{B \subseteq A} m(B)$$

La *fonction de plausibilité (pl)* est la fonction duale de la fonction de croyance, elle mesure l'intensité avec laquelle une proposition A peut être considérée comme vraie. La plausibilité de A est donc la somme des masses de tous les éléments focaux qui sont compatibles avec A.

$$pl(A) = \sum_{B \cap A \neq \emptyset} m(B)$$

2.2 La monotonie de la fonction de croyance

La fonction de croyance est en effet monotone. Si une proposition X est incluse dans une proposition Y, alors la croyance de Y est supérieure à celle de X. Nous verrons que cette propriété est extrêmement importante pour hausser la performance de notre technique d'EIF.

$$X \subseteq Y \Rightarrow bel(X) \leq bel(Y)$$

2.3 La règle conjonctive de combinaison

Soient m_1 et m_2 deux distributions de masse définies pour le même cadre de discernement Θ et fournies par deux corps d'évidence indépendants. La règle conjonctive de combinaison (Smets, 1998) est applicable lorsque les sources d'informations, relatives aux distributions combinées, sont fiables. La règle conjonctive de combinaison est applicable à plus de deux distributions de masse :

$$m_1 \textcircled{\cap} m_2(Z) = \sum_{X,Y \subseteq \Theta : X \cap Y = Z} m_1(X) \times m_2(Y)$$

3 Bases de données évidentielles

3.1 Définition

Une base de données évidentielles permet le stockage de données incertaines et imprécises qui sont modélisées à l'aide de la théorie de Dempster-Shafer. Une base de données évidentielles notée EDB contient n attributs et d lignes. Chaque attribut i $(1 \leq i \leq n)$ a un domaine Θ_i de valeurs discrètes. La valeur de la colonne i pour la ligne j est dite *valeur évidentielle* et notée V_{ij}. Elle correspond à une *bba* qui est définie comme suit :

Définition 1 (Valeur évidentielle) *Soit V_{ij} une valeur évidentielle qui correspond à la colonne i et à la ligne j. V_{ij} correspond à un corps d'évidence composé du cadre de discernement Θ_i (le domaine de la colonne i), de l'ensemble F_{ij} des éléments focaux et de la fonction de masse m_{ij} qui est définie comme suit :*

$$m_{ij} : 2^{\Theta_i} \rightarrow [0, 1] \textit{ avec :}$$

$$m_{ij}(\emptyset) = 0 \textit{ et } \sum_{x \subseteq \Theta_i} m_{ij}(x) = 1$$

3.2 Imperfection de données

Grâce à la théorie de Dempster-Shafer, plusieurs types d'imperfections peuvent être modélisées à travers les données sauvegardées dans une même base de données évidentielles. Ainsi, les données stockées peuvent être :

id	A	B	C
1	$A_{1(0.6)}$	B_1	$C_{2(0.2)}$
	$A_{2(0.4)}$		$\{C_1, C_2\}_{(0.8)}$
2	$A_{1(0.2)}$	B_1	$C_{1(0.5)}$
	$\{A_2, A_3\}_{(0.8)}$		$C_{2(0.5)}$

TAB. 1 – *Exemple d'une base de données évidentielle*

- *Parfaites.* Dans une base de données évidentielles, nous avons mentionné plus haut que la valeur d'un attribut est un corps d'évidence. Quand ce corps d'évidence inclut exactement un élément focal qui est singleton, et dont la masse est par conséquent égale à l'unité, nous parlons de donnée parfaite. Le tableau 1 est une base de données évidentielles exemple où la valeur de la colonne B est parfaite dans les deux lignes d'identifiants 1 et 2.
- *Probabilistes.* Quand le corps d'évidence inclut des éléments focaux singletons, là nous parlons d'une valeur probabiliste. Dans notre base de données exemple, la valeur de la colonne C dans la deuxième ligne est probabiliste.
- *Possibilistes.* Lorsque le corps d'évidence inclut des éléments focaux imbriqués, nous parlons de valeur possibiliste. En effet, la fonction π en théorie de possibilité correspond à la fonction pl en théorie de Dempster-Shafer. Dans ce cas, la valeur de l'attribut C dans la première ligne est possibiliste avec $\pi(C_1) = 0.8$ et $\pi(C_2) = 1$

– *Manquantes.* Si une la valeur d'un attribut i pour une ligne j est manquante, alors la bba V_{ij} inclut un seul élément focal qui coincide avec Θ_i, soit le domaine de l'attribut i, avec une masse égale à l'unité.
– *Évidentielles.* A l'instar de la valeur de l'attribut A dans la deuxième ligne qui n'est ni parfaite, ni probabiliste, ni possibiliste, ni manquante.

3.3 La base de données éducationnelles

Dans cette section, nous présentons la base de données éducationnelles sur laquelle nous allons appliquer notre méthode d'EIF. Ces données concernent une étude qui a été menée sur la formation universitaire des informaticiens en Tunisie. L'objectif de cette étude est de trouver des corrélations entre plusieurs variables tels que les choix d'un répondant lors de son orientation universitaire, la nature de ses études en secondaire, le degré de sa satisfaction par rapport à la formation qu'il a eue (quoique cette information s'étale sur plusieurs variables, puisque le répondant évalue la formation qu'il a eue sur chaque module, de manière indépendante), ses intentions une fois diplômé ainsi qu'une panoplie d'autres variables intéressantes. Le répondant au questionnaire sélectionne forcément pour chaque question (variable) une réponse unique parmi une liste très détaillée de choix. Ces attributs sont dits parfaits, car chaque variable prend une valeur élémentaire.

Une exception concerne cependant une question qui demande au répondant de donner un classement des modules/unités qu'il a étudiés, en fonction de ses préférences par rapport à la qualité de formation qu'il a eue pour ces modules/unités. Le répondant sélectionne entre deux et cinq choix prédéfinis qu'il ordonne selon ses préférences. A chaque fois, il peut choisir un module, ou une unité (un module étant un ensemble d'unités). A titre d'exemple, supposons qu'un étudiant apprenne trois modules pendant sa formation universitaire. Le module *SGBD* [1] qui inclut l'unité *Bases de Données* et l'unité *Administration*, le module *Conception* qui inclut l'unité *Processus Unifié* et l'unité *Merise* et enfin le module *Programmation* qui inclut l'unité *Algorithmique* et l'unité *Structures de Données Avancées*. Ceci n'est qu'un exemple très réduit par rapport au contenu réel du questionnaire, qui englobe la totalité des modules et unités enseignés. Le répondant peut alors faire ses choix entre les modules ou les unités et les classer selon un ordre de préférence.

La liste de choix sélectionnée par le répondant est convertie via la théorie de Dempster-Shafer en une *bba*. En effet, *le cadre de discernement* Θ n'est autre que l'ensemble des unités. Dans notre exemple, $\Theta = \{BD, Ad, PU, M, Al, SD\}$ [2]. *Les éléments focaux* correspondent aux choix du répondant. Un élément focal peut être un singleton dans le cas où le choix porte sur une unité, ou composé dans le cas où le choix porte sur un module. Des *masses absolues* sont affectées aux choix du répondant selon la formule suivante :

Soit c le nombre de choix sélectionnés par le répondant ($2 \leq c \leq 5$) et r_i le rang du choix i ($1 \leq i \leq c$). La masse absolue attribuée à chaque choix i, et donc élément focal, est de $c - r_i + 1$. Ces masses étant absolues, nous les normalisons en les divisant par la somme totale des masses qui n'est autre que $\sum_{1 \leq i \leq c} r_i$, et ce pour les ramener à l'intervalle [0..1]. Nous signalons qu'il existe une panoplie d'autres formules de conversion de préférences qualitatives à des bba, nous

1. Systèmes de Gestion de Bases de Données
2. BD, Ad, PU, M, Al, SD sont les abréviations de Bases de Données, Administration, Processus Unifié, Merise, Algorithmique et Structures de données Avancées.

en citons (Ben Yaghlane et al., 2006). L'exemple suivant explique la représentation évidentielle que nous avons utilisée pour formuler les réponses préférentielles des répondants en *bba*.

Exemple 1 *Un répondant définit la préférence suivante : 1. SGBD ; 2. M ; 3. Al.*

Les choix de ce répondant sont convertis en éléments focaux, soit {BD,Ad}, {M} et {Al}. Des masses absolues sont affectées proportionnellement à l'ordre de préférence aux différents éléments focaux. La masse absolue de {BD,Ad} est de $3(= 3 - 1 + 1)$*, celle de {M} est de* $2(= 3 - 2 + 1)$ *et enfin celle de {Al} est de* $1(= 3 - 3 + 1)$*. On procède par la suite à leur normalisation (en les divisant par* 6*) pour obtenir ce jeu de masses :*
m({BD,Ad})=0.5 ; m({M})=0.33 et m({Al})=0.17

Ainsi, notre base de réponses n'est autre qu'une base de données évidentielles à unique attribut imparfait, celui qui concerne les préférences définies par les répondants par rapport aux modules dont ils jugent avoir eu une excellente formation. Cette base de données éducationnelles sera utilisée pour mener des expérimentations de performance sur la méthode *FIMpED* que nous proposons dans la prochaine section.

4 Extraction des itemsets fréquents à partir de données évidentielles

Nous présentons dans cette section le modèle d'EIF à partir de données évidentielles, qui a été adapté à partir du travail de (Agrawal et al., 1993), introduit dans (Hewawasam et al., 2007) et raffiné dans (Bach Tobji et al., 2008b). Par la suite, nous présentons la nouvelle méthode conçue pour une extraction performante des itemsets fréquents à partir de données à unique attribut évidentiel.

4.1 Concepts de base

Les concepts d'*item*, d'*itemset* et de *support* sont reformulés pour tenir en compte la nature évidentielle des données, et pour produire des mesures exactes relatives à la fréquence et à la confiance des itemsets. Dans ce qui suit nous présentons formellement ces concepts :

Définition 2 (Item évidentiel) *Un item évidentiel noté* iv_i *est un élément focal dans une* bba V_{ij} *qui correspond à un attribut* i *et à une ligne* j*. Par conséquent, il est défini comme étant un sous-ensemble de* Θ_i *(*$iv_i \in 2^{\Theta_i}$*)*

Exemple 2 *Dans notre base de données exemple (tableau 1),* A_1 *ou encore* $\{A_2, A_3\}$ *sont des items évidentiels.*

Définition 3 (Itemset évidentiel) *Un itemset évidentiel est un ensemble d'items évidentiels. Les items évidentiels qui le composent correspondent naturellement à des attributs différents. Formellement, l'itemset évidentiel* X *est défini ainsi :*

$$X \in \prod_{i \in \{1,..,n\}} 2^{\Theta_i}$$

Exemple 3 *L'ensemble d'items* A_1A_3 *ne forme pas un itemset car* A_1 *et* A_2 *correspondent tous deux au même attribut* A*. Par contre,* $A_1\{C_1, C_2\}$ *est un itemset évidentiel valide.*

Un opérateur d'inclusion spécifique aux itemsets évidentiels est aussi défini :

Définition 4 (Inclusion des itemsets évidentiels) *Soient X et Y deux itemsets évidentiels. Les items de X et de Y, relatifs à l'attribut i, sont respectivement notés i_X and i_Y.*

$$X \subseteq Y \text{ si et seulement si : } \forall i_X \in X, i_X \subseteq i_Y$$

Exemple 4 *L'itemset $\{A_1, A_2, A_3\}C_2$ inclut l'itemset $\{A_1, A_2\}C_2$.*

Le fait qu'une ligne de la base puisse contenir une (ou plusieurs) valeur(s) évidentielle(s), cela veut dire que l'imperfection concerne toute la ligne, voire toute la base de données. Dans ce qui suit, nous allons définir *le corps d'évidence d'une ligne*, par la suite en déduire *le corps d'évidence de la base évidentielle*. Une fois ces concepts définis, nous allons voir comment en déduire la fonction de support relative à un itemset évidentiel.

Le corps d'évidence d'une ligne $j \in [1..d]$ est déduit des valeurs évidentielles V_{ij} qui la composent en suivant ces deux étapes :

– Étendre la bba de chaque valeur évidentielle incluse dans la ligne j au cadre de discernement global $\Theta = \prod_{i \leq n} \Theta_i$ (Elouedi et al., 2001). La bba étendue est appliquée sur l'extension cylindrique des éléments focaux. L'extension cylindrique d'un élément focal $X \subseteq \Theta_i$ est notée $X^{\Theta_i \uparrow \Theta}$avec :

$$X^{\Theta_i \uparrow \Theta} = (\Theta_1, .., \Theta_{i-1}, X, \Theta_{i+1}, .., \Theta_n)$$

Il en découle que l'extension de la bba d'une valeur évidentielle -relative à un attribut i- est notée $m_{ij}^{\Theta_i \uparrow \Theta}$ et définie comme suit :

$$m_{ij}^{\Theta_i \uparrow \Theta}(X^{\Theta_i \uparrow \Theta}) = m_{ij}(X)$$

– Déduire le corps d'évidence de la ligne j à partir des extensions des valeurs évidentielles qui la composent. En effet, les bba étendues partagent le même cadre de discernement Θ, nous pouvons donc les combiner grâce à la règle conjonctive présentée dans la section 2.3.

Définition 5 (Corps d'évidence d'une ligne) *Le cadre de discernement d'un corps d'évidence d'une ligne est le produit cartésien des domaines des attributs de la base. Il est noté $\Theta = \prod_{i \leq n} \Theta_i$. Les éléments focaux sont les sous-ensembles de Θ, et donc des vecteurs de la forme $X = \{x_1, x_2, \ldots, x_n\}$ où $x_i \subseteq \Theta_i$. La masse d'un vecteur X dans une ligne j est calculée grâce à la règle conjonctive, qui combine les extensions des bba des valeurs évidentielles qui composent j :*

$$m_j : \Theta \rightarrow [0, 1]$$

$$m_j(\emptyset) = 0$$

$$m_j(X) = \textcircled{\cap}_{x_i \in X} m_{ij}^{\Theta_i \uparrow \Theta}(x_i^{\Theta_i \uparrow \Theta})$$

Exemple 5 *Pour illustrer cette définition, nous présentons ici le corps d'évidence de la première ligne de la base de données exemple (tableau 1). Le cadre de discernement* Θ *est le produit cartésien des domaines des trois attributs qui sont* Θ_A, Θ_B *et* Θ_C. *Les éléments focaux sont les combinaisons de tous les items évidentiels de la ligne en question, et donc tous les itemsets évidentiels possibles que l'ont peut générer de la ligne, soient* $A_1B_1C_2$, $A_1B_1\{C_1, C_2\}$, $A_2B_1C_2$ *et* $A_2B_1\{C_1, C_2\}$. *Dans une première étape, nous étendons la* bba *de chaque valeur évidentielle faisant partie de la ligne* 1. *Par exemple, la* bba *étendue* m_{A1} *qui correspond à l'attribut* A *et la ligne* 1 *est :*

$$m_{A1}^{\Theta_A\uparrow\Theta}(A_1\Theta_B\Theta_C) = 0.6 \; et \; m_{A1}^{\Theta_A\uparrow\Theta}(A_2\Theta_B\Theta_C) = 0.4$$

Une fois les bba *étendues des valeurs évidentielles calculées, nous les combinons à l'aide de la règle conjonctive, nous obtenons :* $m_1(A_1B_1C_2) = 0.6\times1\times0.2 = 0.12$, $m_1(A_1B_1\{C_1, C_2\}) = 0.6 \times 1 \times 0.8 = 0.48$, $m_1(A_2B_1C_2) = 0.4 \times 1 \times 0.2 = 0.08$ *et* $m_1(A_2B_1\{C_1, C_2\}) = 0.4 \times 1 \times 0.8 = 0.32$.

Le corps d'évidence d'une ligne étant défini, nous pouvons maintenant introduire *le corps d'évidence de la base de données*, puisque par définition, une base de données est un ensemble de lignes.

Définition 6 (Corps d'évidence de la base de données) *Le corps d'évidence d'une base de données évidentielle* EDB *est composé par le cadre de discernement* Θ, *l'ensemble des éléments focaux* Foc *composé par tous les itemsets évidentiels existants dans la base, et la fonction de masse* m_{EDB}. *Soit* X *un itemset évidentiel et* d *la taille d'*EDB *:*

$$m_{EDB} : \Theta \rightarrow [0, 1] \; avec \; m_{EDB}(X) = \frac{1}{d}\sum_{j=1}^{d} m_j(X)$$

Les mesures de croyance et de plausibilité sont naturellement définies comme suit :

$$Bel_{EDB}(X) = \sum_{Y\subseteq X} m_{EDB}(Y)$$

$$Pl_{EDB}(X) = \sum_{Y\cap X\neq\emptyset} m_{EDB}(Y)$$

Exemple 6 *Dans la base de données exemple du tableau 1, la masse de l'itemset évidentiel* $\{A_2, A_3\}B_1C_2$ *est la somme de ses masses dans les lignes, divisée (pour normaliser la mesure de masse) par* $d = 2$. *Par conséquent* $m_{EDB}(\{A_2, A_3\}B_1C_2) = 0.2$. *Sa croyance dans la base est la somme des masses de tous ses sous-ensembles, soient* $\{A_2, A_3\}B_1C_2$ *et* $A_2B_1C_2$ *et donc* $Bel_{EDB}(\{A_2, A_3\}B_1C_2) = 0.24$.

La masse d'un itemset évidentiel X dans une base de données évidentielles EDB est la croyance partielle attribuée exactement à X. Par contre, la somme des masses des sous-ensembles de X reflète la croyance totale de X dans EDB qui n'est autre que la fonction de croyance $Bel_{EDB}(X)$. Le *support* de X dans EDB, soit la fréquence de X dans la base, coincide donc avec sa croyance dans le corps d'évidence d'EDB.

Définition 7 (Support d'un itemset évidentiel) *Soit X un itemset évidentiel dans EDB. Le support de X est sa croyance dans la base.*

$$Support(X) = Bel_{EDB}(X)$$

L'objectif de l'EIF, et donc de la méthode que nous introduisons dans la section 4.2, est de calculer l'ensemble F des itemsets évidentiels fréquents d'une base évidentielle EDB, avec un seuil minimum de support noté min_{supp}. Un itemset évidentiel fréquent est un itemset dont le support excède le seuil minimum de support. Soit X un itemset évidentiel et Θ le produit cartésien des domaines des attributs d'EDB. L'ensemble F est défini ainsi :

$$F = \{X \subseteq \Theta / Support(X) \geq min_{sup}\}$$

4.2 La méthode *FIMpED*

La méthode que nous introduisons est appelée *FIMpED* pour *"Frequent Itemset Mining in partially Evidential Databases"*. Elle assure une extraction performante des itemsets fréquents à partir de bases de données à unique attribut évidentiel. Ce type de bases de données est issu de plusieurs applications où une seule mesure est imparfaite (diagnostic d'un médecin, reconnaissance de cibles, mesure observée par un scientifique etc.), et à plusieurs problèmes de classifications où l'attribut classe est imparfait alors que les attributs caractéristiques ne le sont pas (Elouedi et al., 2001).

La méthode *FIMpED* procède en deux phases majeures. Dans la première phase, une structure de données de type arbre, contenant tous les enregistrements de la base, est générée. Cette structure est une adaptation de la structure arbre présentée dans (Hewawasam et al., 2007), appelée *BIT*, et initialement introduite dans (Kubat et al., 2003). Notre adaptation a été effectuée par rapport au type de base de données à unique attribut évidentiel que nous traitons. Dans la deuxième phase se produit l'extraction des itemsets fréquents à partir de la structure arbre. La technique introduite procède à son tour en deux étapes, soit l'extraction d'une part très importante (jusqu'à la moitié) de l'ensemble F en un seul parcours, et en deuxième étape à compléter l'ensemble F à partir du sous-ensemble initialement extrait. Les deux phases de *FIMpED* sont respectivement décrites en détail dans les sections 4.2.1 et 4.2.2.

4.2.1 Construction de la structure de données arbre

La structure d'arbre *BIT*, pour *"Belief Itemset Tree"*, a été introduite dans (Hewawasam et al., 2007). Le *BIT* contient $n + 1$ niveaux hiérarchiques, chacun correspondant à un attribut sachant que la racine est un noeud vide. La structure d'arbre est construite en y insérant les enregistrements de la base de données un par un, de la racine jusqu'à une feuille. Il s'agit donc d'insérer les itemsets évidentiels de notre base dans l'arbre *BIT* de telle manière que chaque itemset évidentiel distinct corresponde à un unique chemin de l'arbre. Un noeud de l'arbre *BIT* porte trois informations ; l'étiquette d'un *item évidentiel*, une valeur de masse *m*, et une valeur de croyance *bel*. La masse et la croyance que porte un noeud correspondent respectivement à la masse et à la croyance du chemin qui part de la racine jusqu'au noeud en question. Un noeud étant un item, un chemin étant constitué de plusieurs noeuds, par conséquent un chemin correspond à un ensemble d'items et donc à un itemset.

Pour expliquer l'algorithme de construction de l'arbre *BIT*, nous introduisons une deuxième base de données exemple dans le tableau 2. Remarquez que seul l'attribut C est imparfait dans cette base.

id	A	B	C
1	A_1	B_1	$C_{2(0.2)}$ $\{C_1, C_2\}_{(0.8)}$
2	A_2	B_1	$C_{1(0.3)}$ $C_{2(0.7)}$
3	A_2	B_1	$\{C_1, C_2, C_3\}$
4	A_1	B_1	$C_{2(0.5)}$ $\{C_1, C_3\}_{(0.5)}$

TAB. 2 – *Base de données à attribut évidentiel unique*

Exemple 7 *La figure 1 est l'arbre BIT correspondant à notre base exemple (tableau 2). Le corps d'évidence de la base inclut six éléments focaux qui sont* $B_1A_1C_2$, $B_1A_1\{C_1, C_2\}$, $B_1A_2C_1$,$B_1A_2C_2$,$B_1A_2\{C_1, C_2, C_3\}$ *et* $B_1A_1\{C_1, C_3\}$. *Il s'agit tout simplement des six chemins dans notre arbre* BIT.

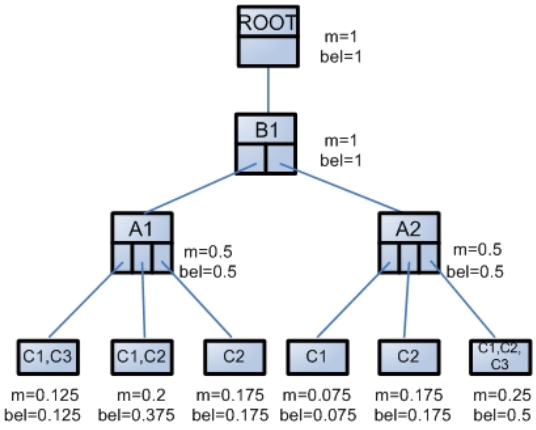

FIG. 1 – *La structure* BIT *correspondante à la base de données du tableau 2*

Remarquez que l'ordre des niveaux de l'arbre n'est pas celui des attributs dans la base (A, B puis C). Le choix n'est pas aléatoire et cela fait l'objet de la première amélioration apportée à l'arbre *BIT* initialement introduit dans (Hewawasam et al., 2007). En effet, un choix étudié de l'ordre des attributs correspondant aux niveaux de l'arbre nous permettra de réduire la taille de l'arbre et par conséquent le temps de parcours de la structure de données. Nous calculons alors pour chaque attribut i la taille de l'ensemble $Dist_i$ qui inclut les valeurs distinctes de i

dans EDB. Naturellement, $Dist_i \subseteq \Theta_i$. Ce calcul ne concerne que les attributs parfaits. L'attribut évidentiel (dans notre cas l'attribut C) n'est pas concerné par cette opération et occupera de toutes les façons le dernier niveau de l'arbre pour des raisons d'optimisation (que nous expliquerons plus tard). Ensuite les attributs i parfaits ($i \in [1..n-1]$) sont triés de manière croissante selon les tailles de leurs $Dist_i$. C'est pour cela que dans notre exemple, l'attribut B occupe le deuxième niveau (puisque $|Dist_B| = 1$) et l'attribut A occupe le deuxième niveau ($|Dist_A| = 2$). Si dans notre exemple, on inverse les niveaux A et B, nous verrons que la taille de l'arbre (mesurée par le nombre de noeuds) augmentera.

Un deuxième changement apporté à la structure initiale *BIT*, consiste à introduire l'information bel dans chaque noeud de l'arbre. Cette information sera calculée à la volée quand un élément focal est inséré dans la structure arbre. S'il existe déjà un chemin qui correspond à l'élément focal, alors les masses et les croyances de ses noeuds sont tout simplement incrémentés par sa masse, sinon un nouveau chemin est créé et les masses et les croyances de ses noeuds sont initialisés par sa masse. L'information bel fournit à n'importe quel noeud de l'arbre, la croyance de l'itemset construit par les items du chemin qui part de la racine jusqu'au noeud en question. Cette information est désormais disponible dans la prochaine phase d'extraction, elle est calculée dans (Hewawasam et al., 2007) et (Bach Tobji et al., 2008b) respectivement après la construction des structures de données *BIT* et *TIDLIST*. Elle donnera un avantage certain à notre technique. L'algorithme 1 est le pseudo-code d'insertion des éléments focaux d'une base de données évidentielles dans la structure d'arbre *BIT*. Le tableau 3 contient une description détaillée des objets et des méthodes utilisées dans les algorithmes.

Notez que le dernier item (qui est évidentiel) de chaque enregistrement correspond à une feuille dans l'arbre *BIT*. Son insertion provoque deux opérations pour calculer sa croyance (ou plus exactement la croyance du chemin qu'il achève). D'abord, nous ajoutons à sa croyance, les masses de tous les noeuds qui lui sont frères et en plus sous-ensembles (c.-à-d. qu'il inclut) (ligne 17, 18 et 19 dans l'algorithme 1). Ensuite, nous mettons à jour les croyances de ses noeuds frères dont il est sous-ensemble (c.-à-d. qui l'incluent) en leur ajoutant sa masse (lignes 21, 22 et 23 de l'algorithme 1).

L'algorithme 1 permet l'insertion d'un seul enregistrement dans l'arbre *BIT*. Cet algorithme doit être alors itéré autant de fois que d'enregistrements dans la base de données fouillée.

4.2.2 Génération des itemsets fréquents à partir de l'arbre *BIT*

Dans cette section, nous présentons en détail la technique d'EIF à partir de la structure *BIT*. L'extraction se produit en deux étapes. La première étape consiste à parcourir l'arbre une seule fois dans le but d'extraire ce qu'on appelle les itemsets *fréquents rapides*. Ces itemsets sont qualifiés de "rapides" car ils sont calculés dès le premier parcours de l'arbre. Ce calcul s'effectue ainsi :

Notre algorithme d'extraction explore la totalité des chemins de l'arbre *BIT*. Chaque chemin qui commence à la racine et qui s'arrête sur le noeud le plus profond dont le support dépasse le seuil minimum de support min_{supp} est un itemset *fréquent rapide*. Par exemple, supposons que le seuil de support est à 0.5. Les chemins les plus profonds correspondants aux itemsets fréquents rapides dans notre base de données exemple (tableau 2) sont $B_1A_2\{C_1, C_2, C_3\}$ et B_1A_1. Une fois que nous avons un ensemble d'itemsets fréquents, nous générons leurs sous-ensembles qui sont également fréquents par définition, puisque la fonction de croyance est monotone (voir la section 2.2).

Classes/Méthodes	**Description**
Item	Classe composée de trois attributs : l'**item** (une étiquette), la valeur de la **masse** (un réel) et la valeur du **support** (un réel aussi).
Node	Objet composé de quatre attributs. Le premier est un tableau noté **value** qui inclut les items. Le deuxième est aussi un tableau noté **child** qui inclut les noeuds fils (de type Node) correspondant à chaque item de **value**. Le troisième attribut est un réel qui inclut une **masse** et le quatrième est aussi un réel qui inclut le **support**.
Record	Classe composée d'un tableau d'items, et de la masse de l'enregistrement (élément focal) dans la base.
Extract(S as Set,k as Integer)	C'est une fonction qui retourne l'ensemble des itemsets de taille k inclus dans S.

TAB. 3 – *Description des classes et des méthodes utilisées dans les algorithmes*

Exemple 8 *Supposons que* $min_{supp} = 0.3$. *Dans l'arbre* BIT *de notre base exemple (tableau 2), nous générons les itemsets fréquents rapides en parcourant en profondeur notre structure de données. Ces derniers sont :* $B_1A_2\{C_1, C_2, C_3\}$, B_1A_2, $B_1A_1\{C_1, C_2\}$ *et* B_1A_1. *De plus, tous leurs sous-ensembles sont fréquents, soit :* B_1, A_1, A_2, $\{C_1, C_2, C_3\}$, $\{C_1, C_2\}$, $B_1\{C_1, C_2\}$, $A_2\{C_1, C_2, C_3\}$, $B_1\{C_1, C_2, C_3\}$ *et* $A_1\{C_1, C_2\}$. *Notez qu'un simple parcours de l'arbre nous permet d'extraire une grande partie de l'ensemble* F *qui englobe la totalité des itemsets fréquents dans* EDB *qui excèdent le seuil* min_{supp}.

L'algorithme 2 est le pseudo-code qui permet l'extraction des itemsets fréquents rapides à partir de l'arbre *BIT* dans l'ensemble FF (*Fast Frequent itemsets*). Lorsque l'ensemble FF est calculé, nous générons les sous-ensembles des éléments de FF puisqu'ils sont aussi fréquents, ils seront stockés dans FF aussi.

La deuxième étape consiste à compléter l'ensemble FF pour former l'ensemble F qui contient la totalité des itemsets fréquents dans EDB selon min_{supp}. Pour cela, nous utilisons la même technique itérative utilisée aussi bien dans (Hewawasam et al., 2007) que dans (Bach Tobji et al., 2008b), à savoir calculer les itemsets fréquents de taille k (l'ensemble F_k), générer les itemsets candidats de taille $k + 1$ à partir de F_k et parcourir l'arbre pour calculer leurs supports et n'en garder que les itemsets fréquents. A la différence des techniques classiques, *FIMpED* stocke en mémoire les itemsets fréquents rapides qui, selon nos expérimentations, peuvent aller jusqu'à 50% de la taille totale de F. Chaque fois où *FIMpED* génère l'ensemble C_k des itemsets candidats, il l'élague avec l'ensemble FF, ce qui réduit l'espace de recherche et améliore sa performance.

Exemple 9 *L'ensemble* FF *étant le suivant* $\{B_1, A_1, A_2, \{C_1, C_2, C_3\}, \{C_1, C_2\}, B_1A_2, B_1A_1, B_1\{C_1, C_2\}, A_2\{C_1, C_2, C_3\}, B_1\{C_1, C_2, C_3\}, A_1\{C_1, C_2\}, B_1A_1\{C_1, C_2\}, B_1A_2 \{C_1, C_2, C_3\}\}$. *A l'étape deux, il s'agit de compléter l'ensemble* F *de tous les itemsets fréquents dans notre base de données exemple, avec un seuil de support fixé à* 0.3. *Nous commençons par l'itération 1, pour extraire tous les items férquents. L'ensemble des candidats dans*

Algorithm 1 InsertRecord

Require: r as Record, k as Integer
Ensure: BIT as Node
if $k < n + 1$ {Traitement de tous les niveaux à l'exception du dernier} **then**
  if $BIT.value$ includes $r(k)$ in position p **then**
    Increment $BIT.value(p).mass$ and $BIT.value(p).support$ by $r.mass$
  else
    Add the item $r(k)$ in $BIT.value$ with its mass
    Add an empty Node to $BIT.child$
    Affect to p the last position of the array $BIT.value$ {p est donc la position de $r(k)$ dans le tableau *BIT.value*}
  end if
  $InsertRecord(r, k + 1, BIT.child(p))$
else
  {Traitement des items évidentiels, et donc des feuilles de l'arbre *BIT*}
  if $BIT.value$ includes $r(k)$ in position p **then**
    Increment $BIT.value(p).mass$ and $BIT.value(p).support$ by $r.mass$
  else
    Add the item $r(k)$ in $BIT.value$ with its mass
    Affect to p the last position of the array $BIT.value$
    for all item i in $BIT.value/i \subset r(k)$ **do**
      Increment $BIT.value(p).mass$ by $i.mass$ {Transfert à la volée des masses des sous-ensembles de r(k) à la croyance de r(k)}
    end for
  end if
  for all item i in $BIT.value/i \supset r(k)$ **do**
    Increment $i.support$ by $r.mass$ {Transfert à la volée de la masse de r(k) aux croyances des sur-ensembles de r(k) }
  end for
end if

cette preimère itération (noté C_1) inclut tous les items de l'arbre BIT. Or, seul l'item $\{C_1, C_3\}$ ne figure pas dans FF. En calculant son support dans l'arbre BIT (0.2), nous saurons qu'il est non-fréquent. Il n'intègrera pas l'ensemble F_1 qui inclut tous les items fréquents. $F_1 = \{B_1, A_1, A_2, \{C_1, C_2, C_3\}, \{C_1, C_2\}\}$. C_2 est l'ensemble des itemsets candidats de taille 2 (c.-à-d. composés de deux items). En appliquant la fonction Apriori_Gen sur F_1, nous obtiendrons l'ensemble C_2 suivant $\{B_1A_1$, B_1A_2, B_1C_2, $B_1\{C_1, C_2\}$, $B_1\{C_1, C_2, C_3\}$, A_1C_2, $A_1\{C_1, C_2\}$, $A_1\{C_1, C_2, C_3\}$, A_2C_2, $A_2\{C_1, C_2\}$, $A_2\{C_1, C_2, C_3\}\}$. $C_2 \setminus FF$ étant $\{B_1C_2$, A_1C_2, $A_1\{C_1, C_2, C_3\}$, A_2C_2, $A_2\{C_1, C_2\}\}$, ce seront uniquement ces itemsets dont nous vérifierons les fréquences dans la base de données. Parmi eux, seuls B_1C_2 et $A_1\{C_1, C_2, C_3\}$ sont fréquents. On en déduit que $F_2 = \{B_1A_2$, B_1A_1, $B_1\{C_1, C_2\}$, $A_2\{C_1, C_2, C_3\}$, $B_1\{C_1, C_2, C_3\}$, $A_1\{C_1, C_2\}$, B_1C_2, $A_1\{C_1, C_2, C_3\}\}$. Ensuite, nous claculons l'ensemble C_3 des itemsets candidats de taille 3, via la fonction Apriori_Gen. $C_3 = \{B_1A_2\{C_1, C_2, C_3\}$, $B_1A_1\{C_1, C_2\}$, $B_1A_1\{C_1, C_2, C_3\}\}$. Or $C_3 \setminus F$ nous donne $\{B_1A_1\{C_1, C_2, C_3\}\}$. En calculant le support de cet itemset (0.5), nous rendons compte qu'il est fréquent. Par conséquent on l'ajoute à $F_3 = \{B_1A_1\{C_1, C_2\}$, $B_1A_2\{C_1, C_2, C_3\}$, $B_1A_1\{C_1, C_2, C_3\}\}$. Ayant uniquement trois attributs dans notre base, on ne peut générer des itemsets de taille 4. L'ensemble F est donc l'union de F_1, F_2 et F_3. Remarquez comment à chaque itération i, l'élagage de l'ensemble C_i par les éléments de FF de même taille, a réduit considérablement notre espace de recherche.

L'algorithme 3 est le pseudo-code de *FIMpED* qui complète l'ensemble FF pour obtenir en fin de compte F. La fonction *Apriori_Gen*, utilisée dans notre algorithme, est présentée dans la définition 8. La figure 2 est une description simplifiée du processus global de notre technique.

Algorithm 2 GenerateFast

Require: BIT as Node, min_{supp} as Real
Ensure: FF as Set of Itemsets
for all item i in $BIT.value$ **do**
  if $i.support >= min_{supp}$ and $BIT.child$ not Null **then**
    $GenerateFast(N.child, min_{supp}, FF)$
  else
    $FF = FF \cup Path$
  end if
end for

Définition 8 (La fonction *Apriori_Gen*) *La fonction Apriori_Gen traite en entrée l'ensemble des itemsets fréquents d'une taille k, noté F_k, et produit en sortie l'ensemble des itemsets candidats de taille $k+1$, noté C_{k+1}.*

La fonction génère à partir de chaque couple (X, Y) l'itemset $Z = X \cup Y$, si $X, Y \in F_k$, et si X et Y contiennent $k-1$ items communs. En d'autre termes, l'itemset Z inclut les $k-1$ items en commun entre X et Y et aussi les deux items qui diffèrent entre X et Y.

La fonction de support étant monotone, il suffit qu'un seul sous-ensemble de Z ne soit pas fréquent (et donc n'appartient pas à F_k) pour que Z soit rejeté, puisqu'il sera sûrement non-fréquent. Cette vérification est faite pour chaque itemset Z généré à partir de chaque couple d'itemsets X et Y ayant $k-1$ items communs.

Algorithm 3 FIMpED

Require: BIT as Node, min_{supp} as Real
Ensure: F as Set of Itemsets
$GenerateFast(BIT, min_{supp}, FF)$
$k = 1$
for all item i in BIT **do**
  if $i \notin Extract(FF, 1)$ **then**
    Compute support of i in BIT
    if $i.suport >= min_{supp}$ **then**
      $F(1) = F(1) \cup i$
    end if
  end if
end for
while $F(k) \neq \emptyset$ **do**
  $C = Apriori\_Gen(F(k))$
  $k = k + 1$
  $C = C \setminus Extract(FF, k)$ {L'espace de recherche est élagué ici. C'est l'avantage procuré par *FIMpED* en comparaison avec les méthodes existantes.}
  Compute supports of all itemsets of C in BIT {Le support d'un itemset est calculé via un simple scan de l'arbre *BIT*}
  Remove infrequent itemsets according to min_{supp} from the set C
  $F(k) = C$
  $C = \emptyset$
end while
$F = F \cup FF$

5 Expérimentations

Nous présentons dans cette section les tests et expérimentations que nous avons menés sur quelques bases de données évidentielles. Nous présenterons en premier lieu les bases de données synthétiques ainsi que les paramètres selon lesquels nous les avons générées, et en

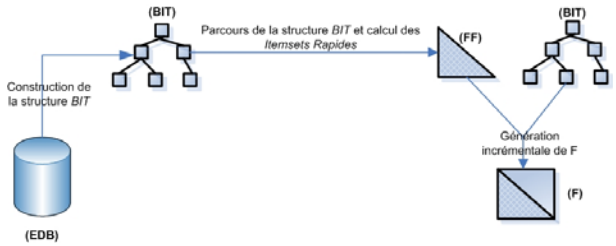

FIG. 2 – *Une description simplifiée de la méthode* FIMpED *process*

second lieu les caractéristiques de la base de données éducationnelles. Ensuite nous montrerons l'ensemble des résultats des opérations de fouille de données que nous avons exécutées sur ces données.

5.1 Les bases de données

Afin d'évaluer la performance de notre technique, nous avons généré quelques bases de données synthétiques selon certains paramètres. Il y a tout d'abord le paramètre D qui est la taille de la base, le paramètre C qui est le nombre d'attributs de la base, le paramètre I qui est la taille de l'ensemble des domaines des attributs de la base ($I = \sum_{i=1}^{C} |\Theta_i|$) et le paramètre $\%U$ qui est le pourcentage des lignes incluant des valeurs évidentielles. Il y a aussi le paramètre nef qui représente le nombre maximum d'éléments focaux dans une valeur évidentielle. Ainsi que le paramètre tef qui représente la taille maximale d'un élément focal à l'intérieur d'une valeur évidentielle. Par exemple, dans le tableau 2, le nombre maximum d'éléments focaux par valeur est $nef = 2$, alors que la taille maximale d'un élément focal est de $tef = 3$, celle de l'élément $\{C_1, C_2, C_3\}$. Ces deux paramètres étant des bornes supérieures, leurs valeurs effectives lors de la génération des lignes de la base sont choisies de manière aléatoire. En d'autres termes, si $nef = 2$ et que $tef = 3$, alors chaque fois que le générateur s'apprête à générer une valeur évidentielle de la base synthétique, il choisit aléatoirement le nombre d'éléments focaux et leurs tailles tout en respectant les limites respectives de 2 et de 3.

Nous avons généré plusieurs bases de données en variant les paramètres cités ci-dessus. Nous retenons la base $D3000I400C30\%U10$, avec $nef = 2$ et $tef = 3$, sur laquelle nous allons effectuer des opérations d'EIF via les algorithmes *FIMpED*, *FIMED* (Bach Tobji et al., 2008b) et *HPSS* (Hewawasam et al., 2007) sur un même intervalle de seuils minimums de supports.

Outre les bases de données synthétiques, nous utiliserons la base de données éducationnelles présentée dans la section 3.3 pour mener quelques expérimentations sur les méthodes citées ci-dessus. Cette base contient exactement 521 lignes, 34 attributs et 177 items. L'attribut évidentiel peut contenir de 2 jusqu'à 5 éléments focaux, chacun pouvant inclure de 1 à 5 éléments.

Rappelons que les bases de données utilisées dans les expérimentations ont toutes un unique attribut évidentiel.

5.2 Résultats des expérimentations

Nous avons implémenté *FIMpED*, *FIMED* et *HPSS* pour les tester sur nos bases synthétiques, ainsi que sur notre base de données réelles. Nous avons commencé par extraire les itemsets fréquents à partir de la base $D3000I400C30\%U10$ sur un intervalle de seuils minimums de support allant de 3% jusqu'à 20%. A noter que les paramètres nef et tef sont mis respectivement à 2 et 3. La figure 3 montre que *FIMpED* a été le plus performant, et ce grâce à l'avantage qu'il prend lorsqu'il parcourt l'arbre *BIT* pour la première fois, générant une grande partie de l'ensemble F des itemsets fréquents dans un laps de temps réduit.

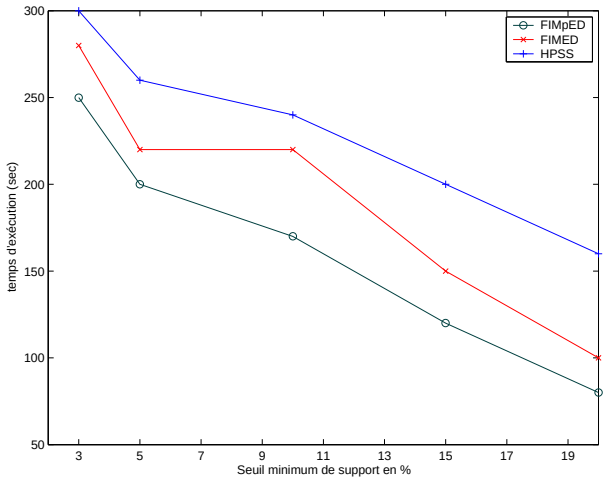

FIG. 3 – *Comparaison des performances de FIMpED, FIMED et HPSS pour la base* $D3000I400C30\%U10nef2tef3$

La même expérimentation sur notre base de données éducationnelles a pratiquement donné les mêmes résultats que nous jugeons satisfaisants. Le nombre de lignes ainsi que le nombre d'items étant relativement réduits par rapport à la base synthétique citée ci-dessus, nous avons décidé de réduire les valeurs des seuils minimums de supports d'extraction, ce qui n'a pas altéré les performances des méthodes testées. La figure 4 nous montre les résultats de cette expérimentation.

La troisième expérimentation que nous avons menée, concerne le comportement de *FIMpED* par rapport à la densité des bases fouillées. En effet, une base de données est dite *dense* si elle contient un nombre d'items réduit pour un certain nombre de lignes. Plus le nombre d'items augmente dans une base (pour un même nombre de lignes), moins elle est dense, et plus elle est *éparse*. Dans une base dense, les mêmes items (dont le nomre est réduit) se répètent dans les lignes, ce qui fait qu'on aura un nombre faible d'itemsets. Par contre, dans les bases éparses, les lignes sont beaucoup plus diversifiées, contenant plus d'items différents, ce qui fait qu'on aura un nombre elevé d'itemsets.

Dans le cadre de cette troisième expérimentation, nous avons généré cinq bases synthétiques différentes avec le même $D = 3000$, le même $C = 20$ et pour des valeurs de I allant de

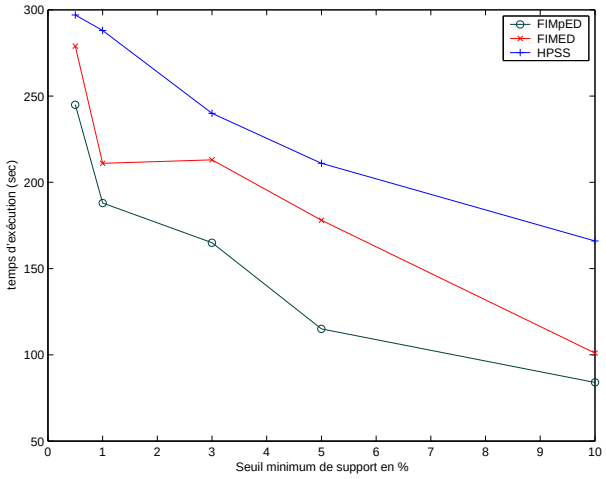

FIG. 4 – *Comparaison des performances de FIMpED, FIMED et HPSS pour la base de données éducationnelles*

100 jusqu'à 500 unités. Toutes les extractions ont été faites avec le seuil de support arbitraire de 4% par *FIMED*, *FIMpED* et *HPSS*. La figure 5 illustre les résultats de cette expérimentation.

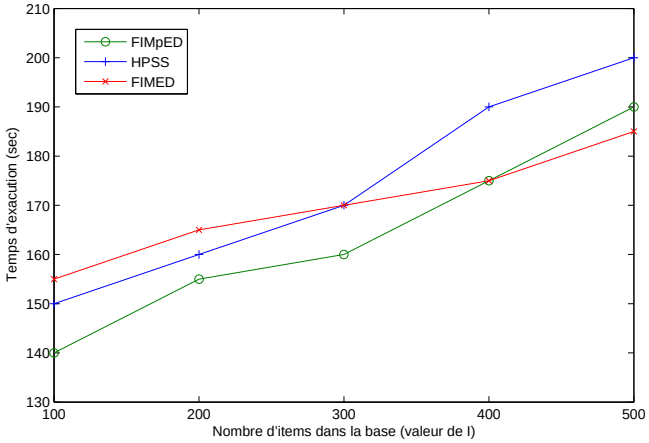

FIG. 5 – *Performance de FIMpED, FIMED et HPSS pour différentes densités de données*

Remarquez dans la figure 5 l'avantage de *HPSS* et *FIMpED*, inhérent à leur structure BIT, qu'elles ont par rapport à *FIMED* lorsque les bases fouillées sont denses. L'avantage de la structure d'arbre *BIT* utilisée, est que plus la base est dense, plus sa taille est petite, et plus *FIMpED* et *HPSS* sont performants puisqu'ils auront à parcourir moins de chemins.

Rappelons que dans une base dense, le nombre d'items est petit par rapport au nombre de lignes. La distribution aléatoire des items sur les lignes nous donne alors des enregistrements qui se "ressemblent". Cela veut dire que le nombre d'itemsets sera relativement faible. Et puisque dans la structure *BIT*, un chemin correspond à un itemset distinct, nous aurons un nombre relativement faible de chemins avec des valeurs de supports très grands (dû au nombre élevé d'occurrences des mêmes itemsets). Par contre, plus la base de données est éparse, plus le nombre d'itemsets est grand, et plus le nombre de chemins dans l'arbre *BIT* généré est grand. C'est ce qui justifie la diminution de la performance de *FIMpED* et *HPSS* lorsque la valeur de *I* augmente, en comparaison avec *FIMED* qui utilise plutôt une structure de données liste.

Passons maintenant à l'importance de l'extraction des itemsets rapides lors du premier parcours de l'arbre *BIT*. Pour ce, nous avons essayé de calculer la taille de l'ensemble FF (ensemble des itemsets fréquents rapides) par rapport à celle de F (ensemble de tous les itemsets fréquents), ainsi que le temps de calcul de FF par rapport à celui de F. Dans le tableau 4, nous présentons deux ratios calculés pour des EIFs qui correspondent à des seuils de support différents. Le premier ratio nous montre la proportion de la taille de FF par rapport à F, tandis que le deuxième nous montre la proportion du temps de calcul de FF par rapport au temps de calcul de F. Cette expérimentation a été menée sur la base de données éducationnelle. Elle montre à quel point FF est une part importante de l'ensemble F (jusqu'à 50%), et qui est calculée en un temps très réduit (jusqu'à 20 fois moins que le calcul de F). L'ensemble FF nous permet d'élaguer considérablement l'espace de recherche (l'ensemble des itemsets candidats C_i pour chaque itération i). Etant un sous-ensemble de F, plus la taille de FF est grande, plus le temps d'exécution de *FIMpED* est réduit et plus notre méthode est performante.

min_{supp}	**Size ratio**	**Time ratio**
0.5%	33%	7%
1%	37%	11%
2%	24%	5%
3%	48%	8%

TAB. 4 – *Impact du premier parcours de* FIMpED *sur sa performance*

Finalement, nous avons essayé de voir l'impact du changement des valeurs des paramètres nef et tef sur la performance de notre solution *FIMpED*. Rappelons que nef est le nombre maximum d'éléments focaux dans une valeur évidentielle, et que tef est la taille maximale que peut avoir un élément focal. Cette expérimentation a été menée une première fois en variant la valeur de nef de 3 à 10 pour une valeur fixe de $tef = 2$. La courbe à gauche dans la figure 6 présente les résultats de cette expérimentation. Par la suite, nous avons varié la valeur de tef de 3 à 10 pour une valeur fixe de $nef = 2$. La courbe de droite dans la même figure présente les résultats obtenus. Nous constatons que la performance de *FIMpED* est surtout dégradée lorsque le nombre d'éléments focaux, par valeur évidentielle, augmente. Ceci est principalement dû à l'augmentation du nombre d'itemsets évidentiels à traiter. Par contre, la taille des éléments focaux n'affecte pas considérablement la performance de l'algorithme, si l'on garde le paramètre nef à une valeur réduite. Le calcul des supports étant fait à la volée, une fois que nous disposons de l'arbre *BIT*, traiter des éléments focaux à grandes tailles ou à

petites tailles n'affecte pas les processus coûteux de génération des itemsets candidats ou de calcul de supports.

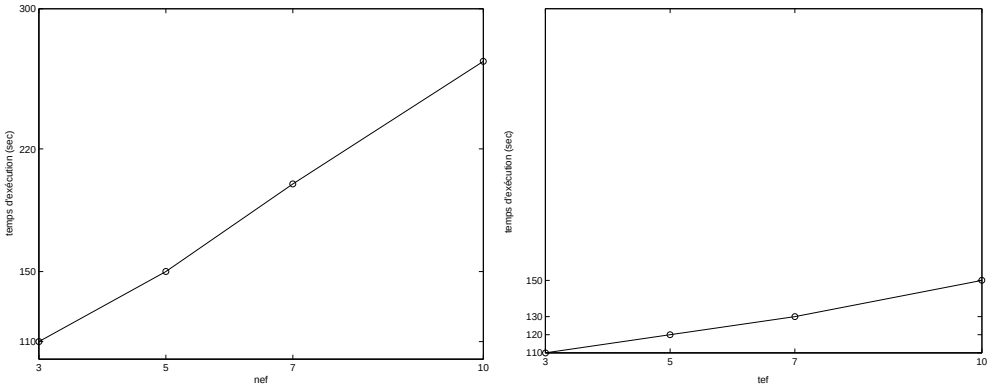

FIG. 6 – *Performance de FIMpED pour une variation des valeurs de* nef *et de* tef

6 Conclusion

Dans cet article, nous avons présenté une technique d'EIF à partir de bases de données évidentielles à unique attribut imparfait. Cette technique, appelée *FIMpED*, se base sur une structure de données arbre que nous avons adaptée à la nature évidentielle des données. Un premier parcours de l'arbre nous permet d'extraire jusqu'à 50% des itemsets fréquents ce qui nous donne un avantage certain par rapport aux méthodes existantes. Les expérimentations menées sur des données synthétiques ou encore des données réelles, ont montré que *FIMpED* est plus performant, en temps d'exécution, que des méthodes telles que *FIMED* et *HPSS*.

Une extension possible de ce travail pourrait concerner l'analyse des connaissances générées à la suite des opérations de fouille de données. Ces connaissances ne manquent pas de complexité à l'instar des données à partir desquelles elles ont été extraites. Le preneur de décision, qui veut profiter de cette base énorme de connaissances, cherche à se faire aider par de nouvelles techniques informatiques de filtrage et de sélection, capables de traiter ce type complexe de motifs générés.

Références

Agrawal, R., T. Imielinski, et A. Swami (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington DC USA, pp. 207–216.

Agrawal, R. et R. Srikant (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago de Chile, Chile, pp. 487–499.

Bach Tobji, M. A., B. Ben Yaghlane, et K. Mellouli (2008a). Frequent itemset mining from databases including one evidential attribute. In *Proceedings of the 2nd international conference on Scalable Uncertainty Management*, Napoli, Italty, pp. 19–32.

Bach Tobji, M. A., B. Ben Yaghlane, et K. Mellouli (2008b). A new algorithm for mining frequent itemsets from evidential databases. In *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Malagà, Spain, pp. 1535–1542.

Ben Yaghlane, A., T. Denoeux, et K. Mellouli (2006). Elicitation of expert opinions for constructing belief functions. In *Proceedings of the 13th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Paris, France, pp. 403–411.

Chen, G. et Q. Wei (2002). Fuzzy association rules and the extended mining algorithms. *Information Sciences-Informatics and Computer Science 147*(1-4), 201–228.

Chui, C.-K., B. Kao, et E. Hung (2007). Mining frequent itemsets from uncertain data. In *The 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 47–58.

Dempster, A. P. (1967). Upper and lower probabilities induced by a multiple valued mapping. *The Annals of Mathematical Statistics 38*(2), 325–339.

Djouadi, Y., S. Redaoui, et K. Amroun (2007). Mining association rules under imprecision and vagueness : towards a possibilistic approach. In *IEEE International Fuzzy Systems Conference*, London, UK, pp. 1–6.

Dubois, D., E. Hullermeier, et H. Prade (2006). A systematic approach to the assessment of fuzzy association rules. *Data Mining and Knowledge Discovery 13*(2), 167–192.

Elouedi, Z., K. Mellouli, et P. Smets (2001). Belief decision trees : Theoretical foundations. *International Journal of Approximate Reasoning 28*, 91–124.

Han, J., J. Pei, et Y. Yin (2000). Mining frequent patterns without candidate generation. In *Proceedings Of The 2000 ACM SIGMOD International Conference On Management Of Data*, Dallas, Texas, United States, pp. 1–12.

Hewawasam, K., K. Premaratne, et M.-L. Shyu (2007). Rule mining and classification in a situation assessment application : a belief theoretic approach for handling data imperfections. *IEEE Transactions on Systems, Man and Cybernetics 37*(6), 1446–1459.

Konias, S., I. Chouvarda, I. Vlahavas, et N. Maglaveras (2005). A novel approach for incremental uncertainty rule generation from databases with missing values handling : application to dynamic medical databases. *Medical Informatics and The Internet in Medicine 30*(3), 211–225.

Kubat, M., A. Hafez, V. V. Raghavan, J. R. Lekkala, et W. K. Chen (2003). Itemset trees for targeted tssociation querying. *IEEE Transactions on Knowledge and Data Engineering 15*(6), 1522–1534.

Kwan, S. K., F. Olken, et D. Rotem (1996). Uncertain, incomplete, and inconsistent data in scientific and statistical databases. In *Uncertainty Management in Information Systems*, pp. 127–154. Kluwer Academic Publishers.

Leung, C. K.-S., C. L. Carmichael, et B. Hao (2007). Efficient mining of frequent patterns from uncertain data. In *Seventh IEEE International Conference on Data Mining - Workshops*, pp.

489–494.

Lucchese, C., S. Orlando, R. Perego, P. Palmerini, R. Perego, et F. Silvestri (2003). kdci : A multi-strategy algorithm for mining frequent sets. In *Proceedings of the IEEE ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, Melbourne, Florida, USA.

Shafer, G. (1976). A mathematical theory of evidence. *Princeton University Press*.

Smets, P. (1998). The application of the transferable belief model to diagnostic problems. *International Journal of Intelligent Systems 13*(2-3), 127–157.

Vaughn, R. B., J. Farrell, R. Henning, M. Knepper, et K. Fox (2005). Sensor fusion and automatic vulnerability analysis. In *Proceedings Of The 4Th International Symposium On Information And Communication Technologies*, Cape Town, South Africa, pp. 230–235.

Wang, X., C. Borgelt, et R. Kruse (2005). Mining fuzzy frequent item sets. In *IEEE International Fuzzy Systems Conference*, Reno, USA, pp. 528–533.

Zhao, Q. et S. S. Bhowmick (2003). Association rule mining : A survey. Technical Report 2003116, Nanyang Technological University, Singapore.

Summary

In this paper, we tackle the problem of frequent itemset mining from imperfect databases, and particularly from what we call *evidential* databases. An evidential database stores data whose imperfection is modelled via the evidence theory. Our new method is based on a tree data structure that is suitable for the evidential nature of the data. This new technique find out until 50% of the whole of the frequent itemsets. We applied it to educational data and synthetic data. The experimentations led to compare our solution to the existing ones showed good results in term of execution time.