LOCAL-GENERATOR : "diviser pour régner" pour l'extraction des traverses minimales d'un hypergraphe

M. Nidhal Jelassi* **, Christine Largeron*, Sadok Ben Yahia**

* Université Jean Monnet, Saint-Etienne, France.
nidhal.jelassi, christine.largeron@univ-st-etienne.fr
**Faculté des Sciences de Tunis, Tunis, Tunisia.
nidhal.jelassi, sadok.benyahia@fst.rnu.tn

Résumé. Du fait qu'elles apportent des solutions dans de nombreuses applications, les traverses minimales des hypergraphes ne cessent de susciter l'intérêt de la communauté scientifique et le développement d'algorithmes pour les calculer. Dans cet article, nous présentons une nouvelle approche pour l'optimisation de l'extraction des traverses minimales basée sur les notions d'hypergraphe partiel et de traverses minimales locales selon une stratégie *diviser pour régner*. Nous introduisons aussi un nouvel algorithme, appelé LOCAL-GENERATOR pour le calcul des traverses minimales. Les expérimentations effectuées sur divers jeux de données ont montré l'intérêt de notre approche, notamment sur les hypergraphes ayant un nombre de transversalité élevé et renfermant un nombre très important de traverses minimales.
Mots-clés: Hypergraphe, traverse minimale, nombre de transversalité, hypergraphe partiel

1 Introduction

Bien que réputé comme singulièrement difficile et considéré comme NP-complet, le problème du calcul des traverses minimales d'un hypergraphe a suscité l'intérêt de la communauté scientifique (Berge (1989); Kavvadias et Stavropoulos (2005); Hébert et al. (2007); Murakami et Uno (2013); Toda (2013)). Cet intérêt pour les traverses minimales est dû au fait qu'elles présentent une solution pour de nombreuses applications dans des domaines variés tel que la cryptographie, le web sémantique, l'e-commerce, etc. (Hagen (2008)).

La principale difficulté que pose l'extraction des traverses minimales réside dans le nombre exponentiel de ces dernières, même quand l'hypergraphe d'entrée est simple. À titre d'exemple, considérons l'hypergraphe $H = (\mathcal{X}, \xi)$ avec l'ensemble des sommets $\mathcal{X}$ = (x_1, x_2, ..., x_{2n}) et l'ensemble des hyperarêtes $\xi = (\{x_1, x_2\}, \{x_3, x_4\}, \ldots, \{x_{2n-1}, x_{2n}\})$. Le nombre de traverses minimales est égal à 2^n tandis que le nombre de sommets est égal à $2n$.

Les algorithmes d'extraction des traverses minimales les plus performants sont des améliorations de l'algorithme de Berge (1989). Ce dernier traite les hyperarêtes une à une en calculant à chaque itération *i* les traverses minimales de l'hypergraphe constitué par les *i-èmes* hyperarêtes considérées. Avec pour objectif d'optimiser le calcul des traverses minimales, notre approche repose sur cette idée en usant du paradigme "*diviser pour régner*". Le principe consiste

à réduire ce nombre d'itérations en partitionnant l'hypergraphe en un nombre précis d'hypergraphes partiels, équivalent au nombre de transversalité de l'hypergraphe d'entrée H. A partir de chaque hypergraphe partiel H_i, nous calculons alors ce que nous appelons *les traverses minimales locales* à H_i. Le produit cartésien de ces traverses minimales locales correspondra alors à un ensemble de traverses de H qui seront soumises à une vérification de la minimalité pour être considérées comme des traverses minimales. En outre, pour un hypergraphe H avec un nombre de transversalité égal à k, le fait de partitionner H en k hypergraphes H_i permet d'éliminer le test de la minimalité pour les ensembles de sommets de taille k qui seront considérés comme traverses minimales de H, sans aucun autre calcul supplémentaire.

Cet article est organisé comme suit : dans la section 2, nous reviendrons sur différents algorithmes, proposés dans la littérature, pour calculer les traverses minimales d'un hypergraphe. Ensuite, dans la section 3, nous rappelons des notions-clés issues de la théorie des graphes et de la fouille de données, que nous combinerons pour introduire notre approche basée sur la notion de traverse minimale locale et d'hypergraphe partiel. La section 4 présentera notre approche pour le calcul des traverses minimales à travers un algorithme original, appelé LOCAL-GENERATOR. Enfin, une étude expérimentale sur des hypergraphes aléatoires sera décrite dans la section 5.

2 Etat de l'art

Berge a été le premier à proposer un algorithme dédié à l'extraction des traverses minimales mais cette solution est impraticable sur des hypergraphes de grande taille car elle nécessite le stockage des traverses minimales temporaires, calculées après chaque ajout d'une nouvelle hyperarête. En effet, l'algorithme de Berge commence par calculer l'ensemble des traverses minimales de la première hyperarête avant de le mettre à jour en ajoutant incrémentalement les autres hyperarêtes, une à une. D'après Hagen (2008) et Takata (2007), l'algorithme de Berge présente une complexité super-polynomiale en la taille de l'entrée et de la sortie. Récemment, Boros et al. (2008) ont prouvé que le temps d'exécution de l'algorithme avait une borne supérieure sub-exponentielle.

Plusieurs travaux ont ensuite tenté d'améliorer l'algorithme de Berge, parmi lesquels on peut citer ceux de Bailey et al. (2003), Dong et Li (2005) et Kavvadias et Stavropoulos (2005). L'approche la plus performante est celle de ces derniers qui, pour remédier à la principale lacune de l'algorithme pionner, à savoir une consommation excessive de mémoire, ont proposé une technique consistant à combiner les sommets qui appartiennent aux mêmes hyperarêtes à chaque itération de l'algorithme. Alors que l'algorithme de Berge effectue un parcours en largeur de l'arbre des traverses minimales, Kavvadias *et al.* proposent un parcours en profondeur de l'hypergraphe d'entrée en introduisant la notion de *noeud généralisé*, qui représentent un ensemble de sommets appartenant aux mêmes hyperarêtes et qui est mis à jour. Enfin, notons que dans cet algorithme, la vérification de la minimalité est effectuée à travers la notion de *noeud approprié*.

En l'assimilant à un problème de dualisation de fonctions booléennes et monotones, Fredman et Khachiyan ont proposé un algorithme incrémental pour l'extraction des traverses minimales en vérifiant si deux formules f et g sont mutuellement duales, (*i.e.* si $f(x) = \bar{g}(\bar{x})$). Cette approche a été reprise, de différentes manières, par Eiter et Gootlob ou encore par Boros (Eiter et al. (2002); Boros et al. (2003)). La version de ce dernier généralise le principe de la

dualisation dans la mesure où les formules ne traitent plus que des variables booléennes mais manipulent, désormais, aussi des variables entières bornées.

Adoptant un parcours en largeur d'abord et opérant par niveau, l'algorithme MTMINER de Hébert et al. (2007) met à contribution les récents travaux sur l'extraction des motifs. Il commence par générer et tester les candidats de taille 1, i.e., les sommets, avant de générer à chaque niveau un nouveau candidat X et calculer l'ensemble $G(X)$ des hyperarêtes qui ne renferment aucun sommet de X. En ce sens, si le cardinal de $G(X)$ est nul, le candidat X est considéré comme une traverse qui est minimale si elle vérifie la propriété d'anti-monotonie de la minimalité dans les classes d'équivalence. Outre cette stratégie d'élagage, Hebert *et al.* introduisent une deuxième propriété selon laquelle aucun sur-ensemble d'une traverse minimale ne peut représenter, à son tour, une traverse minimale.

Les algorithmes MMCS et RS de type SHD ont été proposés plus récemment par Murakami et Uno. Ils adopent une stratégie en profondeur et sont basés, respectivement, sur le principe du branch-and-bound et sur celui du reverse-search. Les auteurs exploitent aussi une nouvelle technique pour vérifier la minimalité des traverses calculées au moyen d'*hyperarête(s) critique(s)* (Murakami et Uno (2013)). Actuellement, ces algorithmes sont considérés dans la littérature comme les plus efficaces aussi bien en termes de temps d'exécution qu'en consommation mémoire, notamment sur des hypergraphes de grande taille.

Le dernier algorithme en date dédié au calcul des traverses minimales, est celui de Toda. Il fait appel à des structures de données compressés BDD (diagramme de décision binaire [1]) et ZDD (zéro diagramme de décision binaire supprimée [2]). Ces structures sont, respectivement, une représentation graphique des fonctions booléennes et une variété de BDD spécialisées pour les bases éparses. Les expérimentation de Toda (Toda (2013)) ont monté que son algorithme est hautement compétitif avec les algorithmes existants, en particulier avec les algorithmes SHD.

Enfin, signalons que d'autres travaux se sont intéressés au calcul des traverses minimales approchées dont l'objectif est de produire un ensemble de traverses minimales de façon à ce que chacune n'intersecte qu'un ensemble donné d'hyperarêtes $\xi' \subset \xi$ (Ruchkys et Song (2002); Abreu et van Gemund (2009); Durand et Quafafou (2013)).

Notre approche, qui repose sur le paradigme "diviser pour régner", est une extension de l'algorithme de Berge. Alors que dans ce dernier, ainsi que dans les améliorations qui en ont été proposées, l'idée est de traiter les hyperarêtes une à une, dans cet article, nous proposons de traiter les hyperarêtes ensemble par ensemble. L'hypergraphe d'entrée H se trouve alors partitionné en un nombre d'hypergraphes partiels égal au nombre de transversalité k de H. Chaque hypergraphe partiel renferme des traverses minimales locales et le produit cartésien, combiné à un test de la minimalité, permet de retrouver l'ensemble des traverses minimales de H. Le choix du nombre d'hypergraphes partiels n'est pas arbitraire puisqu'il garantit que les traverses dont la taille est égale à k peuvent être directement considérées comme des traverses minimales de H.

3 Définitions et notations

Dans cette section, nous proposons de présenter des définitions et notations que nous utiliserons tout au long des sections suivantes. Pour aboutir à notre approche d'extraction des

1. binary decision diagram
2. zero-suppressed binary decision diagrams

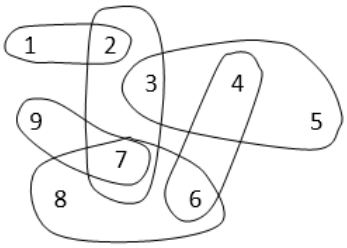
traverses minimales, basée sur la notion des traverse minimale locale, nous avons combiné des concepts de la théorie des hypergraphes (hypergraphe, traverse, nombre de transversalité) avec d'autres de la fouille de données (ensemble essentiel, support). Le nombre de transversalité d'un hypergraphe est la notion-clé de cette section, autour de laquelle est bâtie notre approche.

Définition 1 HYPERGRAPHE SIMPLE *(Berge (1989))*
Soit $H = (\mathcal{X}, \xi)$ avec $\mathcal{X}$ = $\{x_1, x_2, \ldots, x_n\}$ un ensemble fini d'éléments et ξ = $\{e_1, e_2, \ldots, e_m\}$ une famille de parties de $\mathcal{X}$. H constitue un hypergraphe simple sur $\mathcal{X}$ si :

1. $e_i \neq \emptyset, i \in \{1,\ldots, m\}$;
2. $\bigcup_{i=1,\ldots,m} e_i = \mathcal{X}$;
3. $\forall\, e_i \in \xi$ *et* $e_j \in \xi$, $e_i \subseteq e_j \Rightarrow i = j$.

Les éléments de $\mathcal{X}$ sont appelés sommets et ceux de ξ hyperarêtes de l'hypergraphe et, dans la suite, nous ne considérerons que des hypergraphes simples.

Exemple 1 *La figure 1 illustre un hypergraphe simple $H = (\mathcal{X}, \xi)$ tel que $\mathcal{X} = (1, 2, 3, 4, 5, 6, 7, 8, 9)$ et ξ = $\{$ e_1, e_2, e_3, e_4, e_5, e_6 $\}$ avec e_1 = {1, 2}, e_2 = {2, 3, 7}, e_3 = {3, 4, 5}, e_4 {4, 6}, e_5 = {6, 7, 8} et e_6 = {7, 9}.*



	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
{1, 2}	1	1	0	0	0	0	0	0	0
{2, 3, 7}	0	1	1	0	0	0	1	0	0
{3, 4, 5}	0	0	1	1	1	0	0	0	0
{4, 6}	0	0	0	1	0	1	0	0	0
{6, 7, 8}	0	0	0	0	0	1	1	1	0
{7, 9}	0	0	0	0	0	0	1	0	1

FIG. 1: Un exemple d'hypergraphe $H = (\mathcal{X}, \xi)$ et la matrice d'incidence IM_H correspondante

Définition 2 TRAVERSE MINIMALE ET NOMBRE DE TRANSVERSALITÉ *(Berge (1989))*
Soit un hypergraphe $H = (\mathcal{X}, \xi)$ tel que $\mathcal{X}$ est l'ensemble de sommets et ξ = $\{e_1, e_2, \ldots, e_m\}$ est l'ensemble des hyperarêtes de H. $T \subset \mathcal{X}$ est une traverse de H si $T \bigcap e_i \neq \emptyset\ \forall i = 1, \ldots, m$. γ_H désigne l'ensemble des traverses définies sur H. Une traverse T de γ_H est dite minimale *si $\nexists T_0 \subset T$ s.t. $T_0 \in \gamma_H$. On notera $\mathcal{M}_H$, l'ensemble des traverses minimales définies sur H.*

Le nombre minimum de sommets d'une traverse est appelé le nombre de transversalité *de l'hypergraphe H et on le désigne par : $\tau(H)$ = min {$|T|$, s.t. $T \in \mathcal{M}_H$}. De plus, d'après Berge (1989), $\mathcal{M}_H$ représente les hyperarêtes de l'hypergraphe transversal correspondant à H et défini sur $\mathcal{X}$.*

Exemple 2 *Dans l'exemple illustratif de la figure 1, l'ensemble $\mathcal{M}_H$ de l'hypergraphe est :* $\{$ $\{1, 4, 7\}$, $\{2, 4, 7\}$, $\{2, 5, 6, 7\}$, $\{2, 5, 6, 9\}$, $\{2, 4, 8, 9\}$, $\{2, 4, 6, 9\}$, $\{2, 3, 6, 7\}$, $\{2, 3, 6, 9\}$, $\{1, 5, 6, 7\}$, $\{1, 3, 6, 7\}$, $\{1, 3, 6, 9\}$ *et* $\{1, 3, 4, 8, 9\}$.

Un hypergraphe H peut être représenté par une matrice d'incidence IM_H définie par un triplet $(\xi, \mathcal{X}, \mathcal{R})$ où $\mathcal{R} \subseteq \xi \times \mathcal{X}$ est une relation binaire entre les hyperarêtes et les sommets. Cette matrice d'incidence IM_H, associée à l'hypergraphe $H = (\mathcal{X}, \xi)$, est définie par : $\forall x_i \in \mathcal{X}$ et $\forall e_j \in \xi$, $\mathcal{R}(e_j, x_i) = 1$ si $x_i \in e_j$ et $\mathcal{R}(e_j, x_i) = 0$ sinon. La Table de la figure 1 représente la matrice d'incidence associée à l'hypergraphe H.

Définition 3 SUPPORT D'UN ENSEMBLE DE SOMMETS
Soit l'hypergraphe $H = (\mathcal{X}, \xi)$ et X un sous-ensemble de $\mathcal{X}$. Nous définissons $Supp(X)$ comme le nombre d'hyperarêtes de H, renfermant au moins un élément de X :
$Supp(X) = |\{e \in \xi | \exists x \in X \wedge \mathcal{R}(e, x) = 1\}|$

Ainsi, l'ensemble X peut être vu comme une disjonction de sommets $(x_1 \vee x_2 \vee \ldots \vee x_n)$ tel que la présence d'un seul sommet de X suffit à affirmer que X satisfait une hyperarête donnée, indépendamment des autres sommets.

Définition 4 ENSEMBLE ESSENTIEL DE SOMMETS *(Casali et al. (2005)) Soit l'hypergraphe $H = (\mathcal{X}, \xi)$ et $X \subseteq \mathcal{X}$. X représente un ensemble essentiel de sommets si et seulement si :*
$Supp(X) > \max\{Supp(X \backslash x) \mid \forall x \in X\}$.

Il est important de souligner que les ensembles essentiels, extraits à partir d'une matrice d'incidence, vérifient la propriété d'idéal ordre, *i.e*, si X est un ensemble essentiel, alors $\forall Y \subset X$, Y est aussi un ensemble essentiel. De plus, la notion de traverse peut être redéfinie par le biais du support d'un ensemble de sommets et de la notion d'ensemble essentiel, selon la proposition 1.

Proposition 1 TRAVERSE MINIMALE
Un sous-ensemble de sommets $X \subseteq \mathcal{X}$ est une traverse minimale de l'hypergraphe H, si X est essentiel et si son support est égal au nombre des hyperarêtes de H, autrement dit, X est un ensemble essentiel tel que $Supp(X)$=$|\xi|$.

Preuve 1 *Soit X un ensemble essentiel de sommets tel que $Supp(X)$=$|\xi|$. Par conséquent, $X \bigcap e_i \neq \emptyset \ \forall e_i \in \xi, i = 1, \ldots, m$. Donc, d'après la définition 2, X est une traverse. La minimalité de X tient à son "essentialité". En effet, puisque X est essentiel, alors son support est strictement supérieur à celui de ses sous-ensembles directs. Par conséquent, $\nexists X_1 \subset X$ s.t. $Supp(X_1)$=$|\xi|$. X est donc une traverse minimale.*

Définition 5 UNION ET PRODUIT CARTÉSIEN *(Berge (1989))*
Soit $H = (\mathcal{X}, \xi)$ et $G = (\mathcal{X}', \xi')$ deux hypergraphes tels que $\xi = \{\xi_1, \xi_2, \ldots, \xi_m\}$ et $\xi' = \{\xi'_1, \xi'_2, \ldots, \xi'_m\}$.

$H \cup G$ représente l'union de H et G. Le résultat de cette union est un hypergraphe dont l'ensemble des sommets est constitué de ceux de H et de G, et l'ensemble des hyperarêtes contient celles de H et G, qui par souci de simplification sera aussi noté $H \cup G$:

$H \cup G = (\mathcal{X} \cup \mathcal{X}', \xi_1, \xi_2, \ldots, \xi_m, \xi'_1, \xi'_2, \ldots, \xi'_m)$

$H \times G$ représente le produit cartésien des deux hypergraphes dont le résultat est un hypergraphe dont l'ensemble des sommets contient ceux des deux hypergraphes. Quant à l'ensemble des hyperarêtes, il est aussi noté $H \times G$ et est égal au produit cartésien de ξ et de ξ' autrement dit à l'union de tous les couples possibles d'hyperarêtes tels que le premier élément appartient à ξ et le deuxième à ξ' :

$H \times G = (\mathcal{X} \cup \mathcal{X}', \xi_i \cup \xi'_j, i = 1, \ldots, m, j = 1, \ldots, m')$

Proposition 2 *(Kavvadias et Stavropoulos (2005))*
Soient H et G deux hypergraphes simples. Les traverses minimales de l'hypergraphe $H \cup G$ sont des couples, minimaux au sens de l'inclusion, générés par le produit cartésien des ensembles de traverses minimales de H et de G :
$\mathcal{M}_{H \cup G}$ = *Min{* $\mathcal{M}_H \times \mathcal{M}_G$ *}.*

Définition 6 HYPERGRAPHE PARTIEL*(Berge (1989))*
Un hypergraphe partiel H' est la restriction d'un hypergraphe H à un sous-ensemble d'hyperarêtes ξ' incluses dans ξ et aux sommets contenus dans ces hyperarêtes.

Dans le cadre de cet article, nous proposons d'étendre la proposition 2 en considérant plus de deux hypergraphes. Plus précisément, à partir d'un hypergraphe H=($\mathcal{X}$, ξ), dont le nombre de transversalité $\tau(H)$ est égal à k, et d'une traverse minimale $T = \{x_1, x_2, \ldots, x_k\}$ de $\mathcal{M}_H$ de taille k dont les sommets sont numérotés par ordre de support décroissant, nous proposons de construire k hypergraphes partiels H_i= ($\mathcal{X}_i$, ξ_i), $i = 1, \ldots, k$ tels que :

- $\xi_1 = \{e \in \xi \mid x_1 \in e\}$
- $\mathcal{X}_1 = \{x \in \mathcal{X} \mid x \in e, \forall e \in \xi_1 \}$
- $\xi_i = \{e \in \xi - \bigcup_{j=1}^{i-1} \xi_j \mid x_i \in e\}$
- $\mathcal{X}_i = \{x \in \mathcal{X} \mid x \in e, \forall e \in \xi_i \}$

On peut remarquer que les hypergraphes partiels H_i vérifient de façon évidente les propriétés suivantes :

- $\xi_i \subseteq \xi$
- $\bigcup_{i=1}^{k} \xi_i = \xi$.
- $\nexists e \in \xi$ tel que $e \in \xi_i \cap \xi_j$, $i \neq j$.

Les traverses minimales de l'hypergraphe partiel H_i sont appelées *traverses minimales locales* à H_i et leur ensemble est noté par $\mathcal{M}_{H_i}$.

Exemple 3 *L'hypergraphe de l'exemple illustratif de la Figure 1 a un nombre de transversalité égal à 3. Sachant que H possède 2 traverses minimales de cardinalité minimale égale à 3, $\{1, 4, 7\}$ et $\{2, 4, 7\}$. Prenons, par exemple, $\{1, 4, 7\}$. Après avoir ordonné les trois sommets le composant, selon un ordre décroissant de support, nous obtenons les trois hypergraphes partiels, présentés par la Figure 2, tel que H_1 ne contient que les hyperarêtes auxquelles appartient le sommet 7 (dont le support est égal à 3), H_2 ne contient que celles auxquelles appartient 4 (dont le support est égal à 2) et $H3$ contient les hyperarêtes restantes, i.e., celles qui renferment le sommet 1. Il importe de noter qu'en choisissant $\{2, 4, 7\}$, au lieu de $\{1, 4, 7\}$, le résultat reste le même.*

4 Traverses minimales locales : approche et algorithme

Optimiser le calcul de ces traverses minimales revient donc principalement à réduire le nombre de candidats traités. Ceci passe par la réduction de la taille de l'hypergraphe d'entrée. L'approche proposée dans cet article consiste à construire, à partir de l'hypergraphe d'entrée

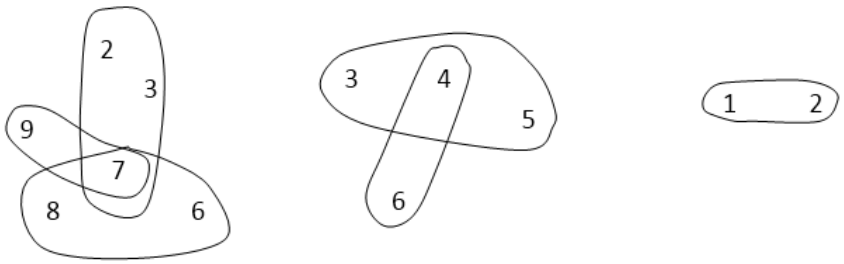


FIG. 2: Les 3 hypergraphes partiels dérivés de H : H_1, H_2 et $H3$

H, k hypergraphes partiels (H_1, H_2, ..., H_k) tel que k correspond au nombre de transversalité de H. Le calcul de l'ensemble des traverses minimales locales, $\mathcal{M}_{H_i}$ de chaque hypergraphe partiel H_i s'en trouve amélioré puisque la taille de H_i est relativement petite par rapport à celle de H. Ainsi, nous proposons d'effectuer l'union des hypergraphes partiels de façon à déterminer l'ensemble des traverses minimales $\mathcal{M}_H$ de H à partir des k-uplets,minimaux au sens de l'inclusion, issus du produits cartésien des ensembles de traverses minimales locales déterminées pour les hypergraphes partiels $\mathcal{M}_{H_i}$. Dans ce qui suit, nous présentons l'algorithme LOCAL-GENERATOR dédié au calcul des traverses minimales et basé essentiellement sur les notions de nombre de transversalité et d'hypergraphe partiel.

4.1 L'algorithme LOCAL-GENERATOR

L'algorithme LOCAL-GENERATOR, dont le pseudo-code, est décrit par l'Algorithme 1 prend en entrée une matrice d'incidence (correspondant à l'hypergraphe d'entrée) et fournit en sortie l'ensemble des traverses minimales. On suppose que les sommets de l'hypergraphe sont triés par ordre lexicographique. LOCAL-GENERATOR démarre par un appel à la fonction GETMINTRANSVERSALITY, dont le pseudo-code est décrit par l'Algorithme 2. Cette fonction recherche une traverse minimale dont la taille est minimale, égale au nombre de transversalité de l'hypergraphe. Pour ce faire, la fonction parcourt les sommets, un par un. Pour chaque élément x_i de $\mathcal{X}$, GETMINTRANSVERSALITY supprime de la matrice d'incidence IM_H les hyperarêtes de ξ qui contiennent x_i. La fonction prend ensuite le sommet, différent de x_i, ayant le plus grand support dans IM_H et réactualise la matrice de la même manière, i.e., en retirant les hyperarêtes contenant ce dernier sommet. Le traitement, entre la ligne 11 et la ligne 15, s'arrête dès que IM_H se vide complètement de toute hyperarête. Le vecteur T_{tmp} sert à stocker les sommets supprimés, un à un à partir de X, pour aboutir à $\xi' = \emptyset$. Si le nombre de sommets de T_{tmp} est le plus petit, obtenu jusque-là dans la fonction, la cardinalité de T_{tmp} est stockée dans min et les éléments de T_{tmp} sont copiés dans T. Le traitement est répété pour tous les sommets de X. A chaque itération de la boucle de la ligne 5, l'ensemble ξ' est réinitialisé avec les éléments de ξ et T_{tmp} est vidé des éléments qu'il renferme. Au final, GETMINTRANSVERSALITY retourne la suite de sommets qui a permis de "vider" la matrice d'incidence en un nombre minimum d'étapes. Cette suite-là, contenue dans T_{tmp}, représente une traverse minimale dont la taille est nécessairement égale au nombre de transversalité de l'hypergraphe d'entrée H.

Une fois le nombre de transversalité calculé et une traverse minimale de taille minimale renvoyée par la fonction GETMINTRANSVERSALITY, l'algorithme LOCAL-GENERATOR

Algorithme 1 : LOCAL-GENERATOR

Entrées : Une matrice d'incidence IM_H associée à $H = (\mathcal{X}, \xi)$
Sorties : $\mathcal{M}_H$, ensemble des traverses minimales de H

1 **début**
2 initialiser(T : vecteur) ;
3 T := GETMINTRANSVERSALITY(IM_H) ;
4 $k = T$.size() ;
5 $i = 1$;
6 **tant que** $i \leq k$ **faire**
7 $\xi_i = \{e \in \xi \mid T[i] \in e\}$;
8 $\mathcal{X}_i = \mathcal{X} \cap \xi_i$;
9 $\mathcal{M}_{H_i}$ = COMPUTE_TM(IM_{H_i}) ;
10 $i = i + 1$;
11 $\gamma_H = \mathcal{M}_{H_1} \times \mathcal{M}_{H_2} \times \ldots \times \mathcal{M}_{H_k}$;
12 **pour chaque** $X \in \gamma_H$ **faire**
13 **si** $\mid X \mid = k$ **alors**
14 $\mathcal{M}_H = \mathcal{M}_H \cup \{X\}$;
15 **sinon**
16 **si** $\nexists\, x \in X : Supp(X) = Supp(X \backslash x)$ **alors**
17 $\mathcal{M}_H = \mathcal{M}_H \cup \{X\}$;
18 **retourner** *($\mathcal{M}_H$)*
19 **fin**

construit, à partir de la matrice d'incidence IM_H, k hypergraphes partiels et fait appel à la fonction COMPUTE_TM pour construire leurs traverses minimales locales, stockées dans $\mathcal{M}_{H_i}$ (ligne 9). Cette fonction [3] prend en entrée une matrice d'incidence associée à un hypergraphe partiel H_i de H et calcule, par niveaux, l'ensemble des traverses minimales locales à H_i selon la définition 1. A la fin de la boucle de la ligne 6, LOCAL-GENERATOR a déjà calculé les ensembles des traverses minimales locales. Le produit cartésien (ligne 11) de ces ensembles $\mathcal{M}_{H_i}$, permet de construire l'ensemble γ_H. Chaque élément de γ_H issu de ce produit cartésien représente une traverse. Il reste à vérifier sa minimalité. Un des intérêts de notre décomposition de l'hypergraphe initial est d'éviter de tester la minimalité des éléments de γ_H dont la cardinalité est égale à k. En effet, ces derniers représentent des traverses minimales de H puisqu'il ne peut pas exister une traverse minimale de taille inférieure au nombre de transversalité de H. Pour les traverses de taille supérieure à k, LOCAL-GENERATOR teste la minimalité (lignes $15 - 16$) suivant la Proposition 1. Si le support d'un candidat X est strictement supérieur au maximum des supports de ses sous-ensembles directs alors X est une traverse minimale et est ajouté à $\mathcal{M}_H$.

3. Dans les expérimentations, nous avons utilisé l'algorithme MTMINER pour implémenter cette fonction.

Algorithme 2 : GETMINTRANSVERSALITY

Entrées : Matrice d'incidence IM_H associée à H
Sorties : T : Vecteur contenant une plus petite traverse minimale

```
1  initialiser(T_tmp : vecteur) ;
2  initialiser(T : vecteur) ;
3  T = T_tmp = ∅ ;
4  min = | ξ | ;
5  foreach x_i ∈ X do
6      T_tmp.clear() ;
7      T_tmp.add(x_i) ;
8      ξ' = ξ ;
9      ξ' = ξ' - | {e ∈ ξ' | x_i ∈ e} | ;
10     while ξ' ≠ ∅ do
11         Max − item = x_j ∈ X s.t. | {e ∈ ξ' | x_j ∈ e} | = max
           {| {e ∈ ξ' | x_l ∈ e} |, x_l ∈ X} ;
12         T_tmp.add(Max − item) ;
13         ξ' = ξ' - | {e ∈ ξ' | Max − item ∈ e} | ;
14     if min > | T_tmp | then
15         min = | T_tmp | ;
16         T = T_tmp ;
17 retourner (T)
```

5 Etude Expérimentale

Différentes expérimentations ont été menées sur des jeux de données variés afin d'évaluer l'algorithme LOCAL-GENERATOR. Le premier lot de jeux de données considérés dans cette étude expérimentale comporte des hypergraphes générés à partir des bases de données "*Accidents*" [4] et "*Connect-4*" [5] alors que le deuxième lot contient des hypergraphes aléatoires générés, à travers le "*random hypergraph generator*" implementé par Boros et *al.* (Boros et al. (2003)), en fonction du nombre de sommets, du nombre d'hyperarêtes et de la taille minimale des hyperarêtes.

	$\|\mathcal{X}\|$	$\|\xi\|$	$\tau(H)$	$\|\mathcal{M}_H\|$	SHD	KS	LOCAL-GENERATOR
Accidents1	81	990	1	1.961	0.301	8.620	6.519
Accidents2	336	10968	2	17.486	2.787	-	11.073
Connect-Win	79	12800	2	4.587.967	88.491	-	200.501

TAB. 1: Caractéristiques et temps de traitement des hypergraphes Accidents et Connect

4. *http ://archive.ics.uci.edu/ml*
5. *http ://fimi.cs.helsinki.fi/data/*

Le tableau 1 détaille les caractéristiques de chacun des hypergraphes du premier lot considéré. La première et la seconde colonne correspond, respectivement, au nombre de sommets et au nombre d'hyperarêtes des différents hypergraphes. La troisième colonne indique le nombre de transversalité alors que la quatrième colonne indique le nombre de traverses minimales que renferme chaque hypergraphe.

Ces trois hypergraphes ont été traités par trois algorithmes : l'algorithme SHD de Murakami et Uno (2013) l'algorithme KS de Kavvadias et Stavropoulos (2005) et notre algorithme LOCAL-GENERATOR. Le tableau 1 récapitule aussi les temps d'exécution de chaque algorithme sur chaque hypergraphe. L'algorithme SHD étant déjà le plus rapide parmi tout ceux proposés dans la littérature, il l'est aussi sur ces trois jeux de données. LOCAL-GENERATOR est moins rapide alors que KS ne parvient pas à traiter les hypergraphes *Accidents2* et *Connect-Win*. Le fait que LOCAL-GENERATOR ait des temps de traitement plus grands que SHD s'explique par le faible nombre de transversalité des 3 hypergraphes qui varie entre 1 et 2 comme le montre le tableau 1. Dans ce cas, le partitionnement de l'hypergraphe d'entrée en hypergraphes partiels, ne permet pas d'optimiser convenablement le calcul des traverses minimales. La stratégie "diviser pour régner" n'est pas pertinente lorsque la taille de la plus petite traverse minimale, d'un hypergraphe donné, est très petite. Extraire directement les traverses minimales sur l'hypergraphe considéré s'avère plus judicieux que de passer par les traverses minimales locales.

	$\lvert\mathcal{X}\rvert$	$\lvert\xi\rvert$	$\tau(H)$	$\lvert\mathcal{M}_H\rvert$	SHD	KS	LOCAL-GENERATOR
$H1$	78	50	8	3.179.102.150	1920.337	3911.431	1500.269
$H2$	95	51	9	5.040.431.550	3608.182	-	2399.088
$H3$	119	91	4	4.186.560.000	3115.226	-	1918.101
$H4$	159	142	20	7.158.203.125	5509.455	-	4275.364

TAB. 2: Caractéristiques et temps de traitement des hypergraphes aléatoires

Le Tableau 2 récapitule les caractéristiques des différents hypergraphes que nous avons générés. Ces données synthétiques ont été générés en fonction des probabilités minimale et maximale d'appartenance d'un sommet aux hyperarêtes dans l'hypergraphe. Si les nombres de sommets et d'hyperarêtes ne sont pas très élévés, ces hypergraphes renferment néanmoins un très grand nombre de traverses minimales qui varie entre $7.158.203.125$ et $3.179.102.150$. Le nombre de transversalité, τ(H), est aussi élevé en comparaison avec les hypergraphes du Tableau 1. Ceci favorise donc notre approche puisque les hypergraphes d'entrée sont partitionnés en un nombre important de petits hypergraphes partiels et, de ce fait, les traverses minimales de taille égale à τ(H) y sont plus nombreuses épargnant ainsi à notre algorithme le test de la minimalité.

Les temps d'exécution, récapitulés dans le tableau 2 montrent que l'algorithme LOCAL-GENERATOR présente des temps plus intéressants que ceux obtenus par les algorithmes KS et SHD. Notons que l'algorithme de Kavvadias et Stavropoulos (2005) ne parvient à extraire les traverses minimales que sur $H1$. Les temps d'exécution supérieurs à 1500 secondes peuvent s'expliquer par le nombre élevé de traverses minimales calculées. L'écart entre LOCAL-GENER-ATOR et SHD varie entre 420 et 1234 secondes. La différence de performances entre les tableaux 1 et 2 permet de dresser un profil des types d'hypergraphes sur lesquels notre approche est plus performante. En effet, LOCAL-GENERATOR présente des temps intéressants

dès que la taille des plus petites traverses minimales de l'hypergraphe d'entrée est élevée ce qui lui permet de partitionner l'hypergraphe en plusieurs hypergraphes partiels.

De plus, le nombre de traverses minimales est très important. Sur des hypergraphes renfermant peu de traverses minimales, LOCAL-GENERATOR peine à se montrer efficace puisque le nombre de traverses minimales devient négligeable par rapport au nombre de candidats traités et testés. Cette constatation est confirmée par le tableau 2 en observant les hypergraphes $H1$ et $H4$. Quand le nombre de traverses minimales a pratiquement doublé, l'écart en secondes est passé de 420 à 1234 entre les deux algorithmes.

6 Conclusion

Dans cet article, nous avons introduit une nouvelle approche pour le calcul des traverses minimales d'un hypergraphe. Cette approche repose sur le paradigme "*diviser pour régner*" afin de partitionner l'hypergraphe d'entrée en hypergraphes partiels, en fonction du nombre de transversalité. Le calcul des traverses minimales locales, correspondantes à ces hypergraphes partiels, permet de retrouver l'ensemble des traverses minimales à travers un produit cartésien combiné à un test de la minimalité. Ceci nous a permis d'introduire un nouvel algorithme LOCAL-GENERATOR pour l'extraction des traverses minimales. L'étude expérimentale a confirmé l'intérêt de notre approche sur un type précis d'hypergrpahes renfermant des propriétés données. Les résultats obtenus nous incitent, par ailleurs, à réfléchir à la mise en place d'un algorithme hybride qui s'auto-adapte à la valeur du nombre de transversalité, i.e., si ce dernier est jugé petit, alors on évitera de générer les hypergraphes partiels. Par ailleurs, nous chercherons à explorer l'espace de recherche en profondeur d'abord dont l'efficacité par rapport à l'exploration par niveaux a été montré.

Remerciements

Ce travail est partiellement soutenu par St-Etienne Metropole (http ://www.agglo-st-etienne .fr/) et le projet Utique CMCU 11G1417

Références

Abreu, R. et A. J. C. van Gemund (2009). A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *Proceedings of the Eighth Symposium on Abstraction Reformulation, and Approximation (SARA'09)*, California, Etats-Unis.

Bailey, J., T. Manoukian, et K. Ramamohanarao (2003). A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM '03)*, Washington, USA, pp. 485–488.

Berge, C. (1989). *Hypergraphs : Combinatorics of Finite Sets* (3rd ed.). North-Holland.

Boros, E., K. Elbassioni, V. Gurvich, et L. Khachiyan (2003). An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals. In *Proceedings of*

11th Annual European Symposium on Algorithms (ESA 2003), Amsterdam, Netherlands, pp. 556–567.

Boros, E., K. Elbassioni, et K. Makino (2008). On Berge multiplication for monotone boolean dualization. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, ICALP '08, pp. 48–59.

Casali, A., R. Cicchetti, et L. Lakhal (2005). Essential patterns : A perfect cover of frequent patterns. In *Proceedings of the 7th International Conference on DaWaK*, Copenhagen, Denmark, pp. 428–437.

Dong, G. et J. Li (2005). Mining border descriptions of emerging patterns from dataset pairs. *Knowledge and Information Systems*, 178–202.

Durand, N. et M. Quafafou (2013). Approximation de bordures de motifs fréquents par le calcul de traverses minimales approchées d'hypergraphes. In *Actes de la 13ème Conférence Francophone sur l'Apprentissage Automatique (CAp 2013)*, Lille, France, pp. to appear.

Eiter, T., G. Gottlob, et K. Makino (2002). New results on monotone dualization and generating hypergraph transversals. In *SIAM Journal On Computing*, pp. 14–22.

Hagen, M. (2008). *Algorithmic and Computational Complexity Issues of MONET*. Phd dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena.

Hébert, C., A. Bretto, et B. Crémilleux (2007). A data mining formalization to improve hypergraph minimal transversal computation. *Fundamdamenta Informaticae 80*(4), 415–433.

Kavvadias, D. J. et E. C. Stavropoulos (2005). An efficient algorithm for the transversal hypergraph generation. *Journal of Graph Algorithms and Applications 9*(2), 239–264.

Murakami, K. et T. Uno (2013). Efficient algorithms for dualizing large-scale hypergraphs. In *Proceedings of the 15th Meeting on Algorithm Engineering and Experiments (ALENEX'13)*, New Orleans, USA, pp. 1–13.

Ruchkys, D. et S. Song (2002). A parallel approximation hitting set algorithm for gene expression analysis. In *Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'02)*, Vitoria, Espirito Santo, Brazil, pp. 75–81.

Takata, K. (2007). A worst-case analysis of the sequential method to list the minimal hitting sets of a hypergraph. *SIAM Journal on Discrete Mathematics 21*(4), 936–946.

Toda, T. (2013). Hypergraph transversal computation with binary decision diagrams. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, Rome, Italy, pp. 91–102.

Summary

Generating minimal transversals from a hypergraph is a problem which is known to be particularly difficult with a large number of applications in both theoretical and applied Computer Science and several works have been proposed, in the literature to solve it. In this paper, we propose a novel approach to optimize the extraction of minimal transversals which is based on both the partial hypergraph and local minimal transversal notions according *divide and conquer* strategy. We introduce a new algorithm, called LOCAL-GENERATOR which computes efficiently minimal transversals and the experiments carried out on several types of hypergraphs showed that LOCAL-GENERATOR obtains an interesting performances.