Université Tunis El Manar
Faculté des Sciences de Tunis

Université Jean Monnet de Saint--Etienne
Laboratoire Hubert Curien

THÈSE

présentée en vue de l'obtention du

Diplôme de Docteur en Informatique

par

Mohamed Nidhal JELASSI

Etude, représentation et applications des traverses minimales d'un hypergraphe

Soutenue publiquement le 08 Décembre 2014
à la Faculté des Sciences de Tunis

Membres du jury :

Rapporteur	-	**M. Jean-Marc PETIT, Professeur, INSA, Lyon (France)**
Rapporteur	-	**M. Mohamed QUAFAFOU, Professeur, UAM, Marseille (France)**
Examinateur	-	**M. Faouzi BEN CHARRADA, Maître de Conférences, FST, Université Tunis El Manar**
Examinateur	-	**Mme Sawssen KRICHEN, Maître de conférences, ISG, Université de Tunis**
Examinateur	-	**M. François JACQUENET, Professeur, UJM, Saint-Etienne (France)**
Directeur de thèse	-	**M. Sadok BEN YAHIA, Professeur, FST, Université Tunis El Manar**
Directeur de thèse	-	**Mme Christine LARGERON, Professeur, UJM, Saint-Etienne (France)**

Préparée sous convention de cotutelle UTM-FST (Tunisie) – UJM-Saint-Etienne (France)

Remerciements

Je tiens tout d'abord à remercier Messieurs Jean-Marc Petit, Professeur à l'Institut National des Sciences Appliquées de Lyon, et Mohamed Quafafou, Professeur à l'Université d'Aix-Marseille, d'avoir accepté d'être les rapporteurs de mon travail de thèse. Leurs lectures minutieuses et leurs remarques pertinentes m'ont permis d'améliorer la qualité de ce manuscrit. Plus généralement, je remercie l'ensemble du jury, notamment Messieurs François Jacquenet, Professeur à l'Université Jean Monnet de Saint-Etienne, Faouzi Ben Charrada, Maître de Conférences HDR à la Faculté des Sciences de Tunis, ainsi que Madame Sawssen Krichen, Maître de Conférences HDR à l'Institut Supérieur de Gestion, qui ont tout de suite accepté d'être examinateurs et juger mon travail.

Je tiens à adresser mes vifs remerciements et ma profonde reconnaissance à Madame Christine Largeron, Professeur a l'Université Jean Monnet de Saint-Etienne, et Monsieur Sadok Ben Yahia, Professeur à la Faculté des Sciences de Tunis, pour avoir dirigé mon travail, pour leurs conseils et disponibilités, pour leurs encouragements et leurs soutiens continus. L'intérêt qu'ils ont manifesté pour mon travail, leurs suggestions, et leurs remarques ont été d'une importance capitale. Sous leur direction, j'ai appris davantage de rigueur, de sens critique et de discipline.

Un merci spécial également à mon frère Nader Jelassi. Ta présence à mes côtés a été précieuse et vitale, notamment lors de nos séjours en France. Nos longues discussions, en arpentant les rues de Clermont et Saint-Etienne, ont été importantes pour l'avancement de cette thèse. Merci d'avoir toujours été là pour moi.

Un immense merci aussi pour mon compagnon de route, mon ami et mon frère Aymen Sellaouti. Nous nous sommes rencontrés lors de notre première année universitaire et nous ne nous sommes plus quittés, gravissant les échelons ensemble. Je te remercie pour tes encouragements, tes conseils et le soutien que tu m'as apporté tout au long de ces années.

Enfin, je remercie tous les membres des laboratoires Lipah pour ces années passées ensemble ainsi que ceux du laboratoire Hubert Curien pour l'accueil chaleureux qui m'est réservé lors de mes différents séjours à Saint-Etienne.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction générale

La théorie des hypergraphes se propose de généraliser la théorie des graphes en introduisant le concept d'hyperarête. Une traverse est définie comme un ensemble de sommets qui intersecte toutes les hyperarêtes d'un hypergraphe et elle est dite minimale si elle l'est au sens de l'inclusion. Le problème de l'extraction des traverses minimales d'un hypergraphe est connu comme étant particulièrement difficile dans la mesure où premièrement, il est connu pour être coNP-complet malgré que sa complexité exacte est une question qui reste toujours ouverte, et deuxièmement de tous les algorithmes qui se sont attachés à calculer les traverses minimales, aucun n'a, à ce jour, une complexité théorique polynomiale en la taille de l'entrée et de la sortie, sauf pour des hypergraphes bien particuliers. De plus, l'un des verrous scientifiques les plus difficiles à affronter, en matière d'extraction des traverses minimales, est le nombre de traverses minimales à explorer qui peut être très élevé même pour des hypergraphes simples.

Pour autant, l'intérêt pour l'extraction des traverses minimales est en nette croissance due principalement aux solutions qu'elles offrent dans divers domaines d'application comme les bases de données, l'intelligence artificielle, l'e-commerce, le web sémantique, etc. Par ailleurs, la fouille de données arrive maintenant à maturité sur les contextes d'extraction classiques pour lesquels les algorithmes ont été mis au point. Ainsi, par exemple, les bases de données commerciales, qui décrivent les achats réalisés par des millions de clients sur des milliers de références sont désor-

mais parfaitement exploitées par les spécialistes à l'aide de méthodes fondées sur les règles d'association. Ces techniques se popularisent aussi dans les domaines médicaux, économiques ou industriels.

Dans ce travail de thèse, nous focalisons notre intérêt sur les liens, déjà prouvés dans la littérature, entre la fouille de données et la théorie des hypergraphes pour redéfinir les notions d'hypergraphe et de traverse minimale. L'adaptation des techniques de la fouille de données, conjuguée à l'exploitation de certaines propriétés des hypergraphes, présente une voie intéressante pour la mise en place d'un cadre méthodologique pour l'optimisation de l'extraction des traverses minimales.

Avec l'emergence du Web 2.0 et des réseaux sociaux, nous avons tout d'abord été amenés à mettre à profit les traverses minimales pour la recherche d'information au sein ces systèmes communautaires. Ceci nous a conduit à proposer une modélisation originale d'un réseau social sous forme d'hypergraphe, particulièrement utile et adapté au cas où on ne dispose pas de toutes les relations entre les individus du réseau social considéré. A partir de cet hypergraphe, nous nous sommes intéressés à une classe particulière de traverses minimales, appelée *traverse minimale multi-membres*, dans le contexte des systèmes communautaires. Un protocole expérimental basé sur des jeux de données du monde réel a confirmé l'intérêt de cette approche. Ce faisant, nous avons été confrontés au problème du nombre important de traverses minimales à extraire même pour un hypergraphe simple. Pour le résoudre, nous préconisons de représenter cet ensemble de manière concise et exacte en exploitant l'irrédondance de l'information dans les hypergraphes. De ce fait, notre travail consiste en la representation de l'ensemble des traverses minimales par un sous-ensemble succinct, composé de traverses minimales irrédondantes. L'introduction d'une mesure d'évaluation, appelée *taux de compacité*, nous permettra de calculer le pourcentage de traverses minimales pouvant être déduite directement à partir de l'ensemble des traverses minimales irrédondantes. Nous avons illustré l'intérêt de cette représentation

concise et exacte des traverses minimales pour résoudre le problème de l'inférence des dépendances fonctionnelles dans le but de calculer la couverture minimale d'une relation donnée.

Par ailleurs, afin d'optimiser le calcul des traverses minimales d'un hypergraphe, nous avons proposé de décomposer l'hypergraphe d'entrée en un nombre d'hypergraphes partiels égal au nombre de transversalité de l'hypergraphe initial. A partir de ces hypergraphes partiels, nous calculons leurs *traverses minimales locales* respectives, dont le produit cartésien nous fournira un ensemble de traverses de l'hypergraphe. Les tests de minimalités permettent, ensuite, de ne garder que les traverses minimales. Le principal intérêt de cette approche est que ces tests sont inutiles pour les traverses dont la taille est égal au nombre de transversalité de l'hypergraphe d'entrée et dont nous sommes sûrs qu'ils sont minimales.

Le présent mémoire, décrivant le travail réalisé au cours de cette thèse, est composé de cinq chapitres.

Le **premier chapitre** introduit le contexte de nos recherches et présente les concepts de base que nous utiliserons dans la suite. La diversité des domaines d'application des traverses minimales est aussi mise en avant dans ce chapitre avec des points d'orgue pour les domaines ayant fait l'objet de nombreux travaux dans la littérature.

Le **deuxième chapitre** présente l'état de l'art et notamment les différents algorithmes, proposés dans la littérature, pour l'extraction des traverses minimales d'un hypergraphe. Ce chapitre mettra en évidence les différentes approches, techniques et autres stratégies utilisés pour présenter des solutions à cette problématique ainsi qu'une étude comparative de ces algorithmes. Cette synthèse permet aussi de situer nos contributions par rapport aux travaux antérieurs.

Notre première contribution est introduite dans le **troisième chapitre**. Nous présentons les *traverses minimales multi-membres* (TMM), qui représentent une "sous-classe" des traverses minimales d'un hypergraphe. Ces TMM sont les plus petites traverses minimales, vérifiant une propriété de recouvrement, d'un hypergraphe d'entrée dont chaque hyperarête représente une communauté d'un réseau social donné. Un algorithme performant d'extraction de ces TMM, décrit dans ce chapitre, repose sur le calcul du nombre de transversalité de l'hypergraphe d'entrée. De plus, une application sur des jeux de données du monde réel est décrite et interprétée pour mettre en exergue l'intérêt des TMM.

Dans le **quatrième chapitre**, nous présentons notre deuxième contribution qui consiste en la representation concise et exacte de l'ensemble des traverses minimales. Cette représentation exploite l'irrédondance de l'information dans les hypergraphes, qui nous a conduit à définir et à mettre en place un cadre méthodologique pour le calcul des traverses minimales irrédondantes. Nous montrons, ensuite, l'intérêt de notre representation concise et exacte des traverses minimales à travers la résolution du problème de l'inférence des dépendances fonctionnelles. Les traverses minimales ayant déjà été utilisées pour optimiser le processus de calcul de la couverture minimale de toutes les dépendances fonctionnelles d'une relation donnée, nous proposons d'utiliser les traverses minimales irrédondantes pour réduire la taille de cette couverture.

Dans le **cinquième chapitre**, nous nous intéressons à l'extraction de toutes les traverses minimales. Étant donné que le nombre de ces dernières peu être exponentiel en la taille de l'hypergraphe, nous proposons d'optimiser leur calcul en décomposant l'hypergraphe d'entrée en des hypergraphes partiels. Nous choisissons un nombre

d'hypergraphes partiels égal au nombre de transversalité de l'hypergraphe d'entrée afin d'éliminer des tests de minimalité et d'optimiser les temps de traitement nécessaires au calcul de toutes les traverses minimales. A partir de chaque hypergraphe partiel, un ensemble de traverses minimales (dites *locales*) est calculé et un produit cartésien de toutes les traverses minimales locales permet de retrouver les traverses minimales de l'hypergraphe initial, suivant une stratégie "*diviser pour régner*".

Chapitre 1

Contexte et concepts de base

1.1 Introduction

Les hypergraphes généralisent la notion de graphe en définissant des hyperarêtes qui contiennent des familles de sommets, contrairement aux arêtes classiques qui ne joignent que deux sommets. D'un point de vue théorique, les hypergraphes permettent de généraliser certains théorèmes de graphes, voire d'en factoriser plusieurs en un seul. D'un point de vue pratique, ils sont parfois préférés aux graphes puisqu'ils modélisent mieux certains types de contraintes. Dans ce chapitre, nous présentons quelques définitions essentielles sur les hypergraphes et les traverses minimales nécessaires à l'introduction de la problématique de l'extraction des traverses minimales, en se basant essentiellement sur les définitions proposées par Berge dans [Ber89]. Ensuite, nous passons en revue le large éventail des domaines d'application des traverses minimales.

1.2 Préliminaires

La théorie des hypergraphes se propose de généraliser la théorie des graphes en introduisant le concept d'hyperarête. Dans un hypergraphe où chaque hyperarête peut contenir plusieurs sommets, une traverse minimale correspond à un sous-ensemble de sommets qui intersecte toutes les hyperarêtes d'un hypergraphe, en étant minimal au sens de l'inclusion.

1.2.1 Hypergraphes

Un hypergraphe $H = (\mathcal{X}, \xi)$ est donc constitué de deux ensembles $\mathcal{X}$ et ξ, et est défini suivant la Définition 1.

Définition 1 HYPERGRAPHE *[Ber89]*
Soit le couple $H = (\mathcal{X}, \xi)$ avec $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ un ensemble fini et $\xi = \{e_1, e_2, \ldots, e_m\}$ une famille de parties de $\mathcal{X}$. H constitue un hypergraphe sur $\mathcal{X}$ si :

1. $e_i \neq \emptyset, i \in \{1, \ldots, m\}$;
2. $\bigcup_{i=1,\ldots,m} e_i = \mathcal{X}$.

Les éléments x_i de $\mathcal{X}$ sont appelés sommets de l'hypergraphe et les éléments e_i de ξ sont appelés hyperarêtes de l'hypergraphe.

Un hypergraphe est dit d'ordre n si $\mid \mathcal{X} \mid = n$ et la taille d'un hypergraphe est égale au nombre d'occurrences des sommets dans ses hyperarêtes.

Exemple 1 *La Figure 1.1 illustre un hypergraphe $H = (\mathcal{X}, \xi)$ d'ordre 8 et de taille 15 tel que $\mathcal{X} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ et $\xi = \{\{1, 2\}, \{2, 3, 7\}, \{3, 4, 5\}, \{4, 6\}, \{6, 7, 8\}, \{7\}\}$.*

Définition 2 HYPERGRAPHE SIMPLE
$H = (\mathcal{X}, \xi)$ est dit hypergraphe simple *si pour tout $e_i \in \xi$ et $e_j \in \xi$, alors $e_i \subseteq e_j \Rightarrow$*

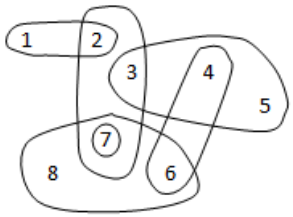


FIGURE 1.1 – Exemple d'un hypergraphe

$i = j$, i.e, aucune hyperarête de H ne renferme une autre hyperarête, sinon, H est dit hypergraphe multiple.

Ainsi, la définition des hypergraphes englobe celle des graphes. En effet, un graphe simple est un hypergraphe simple dont toutes les hyperarêtes sont de cardinalité 2, i.e., $\mid e_i \mid = 2 \; \forall \; e_i \in \xi$.

Propriété 1 *[Ber89]*

Tout hypergraphe simple H d'ordre n vérifie :

$$\sum_{e \in \xi} \binom{n}{\mid e \mid}^{-1} \preceq 1.$$

De plus, le nombre d'arêtes vérifie :

$$\mid \xi \mid \preceq \binom{n}{\lfloor n/2 \rfloor}$$

Définition 3 SOUS-HYPERGRAPHE *[Ber89]*

Soit l'hypergraphe $H = (\mathcal{X}, \xi)$ et $\mathcal{Y} \subseteq \mathcal{X}$, nous appelons sous-hypergraphe de H tout hypergraphe $H^{\mathcal{Y}} = (\mathcal{Y}, \xi^{\mathcal{Y}})$ engendré par $\mathcal{Y}$, tel que $\xi^{\mathcal{Y}} = \{ \; e_i^{\mathcal{Y}} = e_i \cap \mathcal{Y} \mid e_i \in \xi$ et $e_i^{\mathcal{Y}} \cap \mathcal{Y} \neq \emptyset \}$

Définition 4 HYPERGRAPHE PARTIEL *[Ber89]*

Soit l'hypergraphe $H = (\mathcal{X}, \xi)$ et $\xi' \subset \xi$, nous appelons hypergraphe partiel de H

tout hypergraphe $H' = (\mathcal{X}', \xi')$ *engendré par* ξ'*, tel que* $\mathcal{X}' = \bigcup_{e' \in \xi'} e'$*.* H' *est alors la restriction de l'hypergraphe* H *à un sous-ensemble d'hyperarêtes* ξ' *inclus dans* ξ *et aux sommets inclus dans* $\mathcal{X}'$*.*

Le rang *r(H)* d'un hypergraphe H est, le nombre maximum de sommets d'une hyperarête et est défini par *r(H)* = $\max\{|e_i|, \forall\ e_i \in \xi\}$. Inversement, l'anti-rang *ar(H)* désigne le nombre minimum de sommets d'une hyperarête, i.e, *ar(H)* = $\min\{|\ e_i\ |, \forall\ e_i \in \xi\}$ [Ber89] [EG95]. Trivialement, *ar(H)* $\leq$ *r(H)*. Si le rang et l'anti-rang d'un hypergraphe H sont égaux, alors H est dit *uniforme*. Tout hypergraphe uniforme est simple.

Exemple 2 *Considérons l'hypergraphe* H *de la Figure 1.1. Nous avons* $\mathrm{r}(H) = 3$ *et* $\mathrm{ar}(H) = 1$*. Par conséquent,* H *n'est pas uniforme.*

De plus, H est dit *n-uniforme* si H est un hypergraphe simple uniforme de rang n [EG95]. En ce sens, tout graphe est un hypergraphe uniforme de rang égal à 2.

Un hypergraphe est dit "intersectant" si aucun couple de ses hyperarêtes n'est disjoint, i.e., $\forall e_1, e_2 \in \xi$, $e_1 \cap e_2 \neq \emptyset$ [EG95].

Dans un hypergraphe, deux sommets x_i et x_j sont dits *adjacents* s'il existe une hyperarête e_i qui les contient tous les deux ; deux hyperarêtes e_i et e_j sont dites adjacentes si leur intersection est non vide.

Un hypergraphe $H = (\mathcal{X}, \xi)$ peut être représenté par une matrice d'incidence, notée IM_H, définie comme suit :

$$IM_H[e_i, x_j] \begin{cases} = 1 \quad si & x_j \in e_i \\ = 0 \quad sinon & \end{cases}$$

Définition 5 HYPERGRAPHE DUAL

A tout hypergraphe $H = (\mathcal{X}, \xi)$*, nous pouvons faire correspondre un hypergraphe* H^*

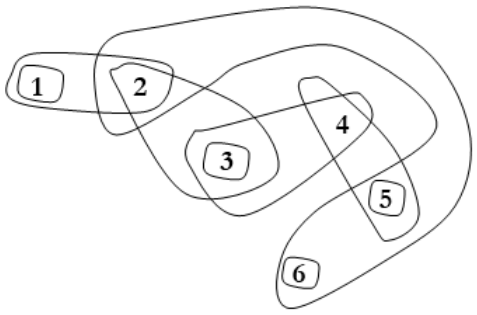


FIGURE 1.2 – Hypergraphe dual

= ($\mathcal{X}^*$, ξ^*) tel que $\mathcal{X}^* = \xi$ et $\xi^* = \mathcal{X}$. Les sommets x_1^*, x_2^*, ..., x_m^* représentent les hyperarêtes de H et les hyperarêtes e_1^*, e_2^*, ..., e_n^* représentent les sommets x_1, x_2, ..., x_n de H, où :

$$X_j = \{E_i \mid i \preceq m, e_i \ni x_j\} \ (j = 1, 2, \ldots, n).$$

On a $X_j \neq \emptyset$, $\bigcup_j X_i = \xi$, donc H^* est bien un hypergraphe.

Exemple 3 *Reconsidérons l'hypergraphe H de la Figure 1.1. La Figure 1.2 illustre l'hypergraphe dual H^* de H tel que $H^* = (\mathcal{X}^*, \xi^*)$ $\mathcal{X}^* = \{1, 2, 3, 4, 5, 6\}$ et $\xi^* = \{\{1\}, \{1, 2\}, \{2, 3\}, \{3, 4\}, \{3\}\ \{4, 5\}, \{2, 5, 6\}, \{5\}, \{6\}\}$.*

H^* est appelé l'*hypergraphe dual* de H. La matrice d'incidence IM_H de l'hypergraphe H, et la matrice d'incidence IM_{H^*} de l'hypergraphe H^*, se déduisent l'une de l'autre par transposition ; on a donc en particulier $(H^*)^* = H$. Si deux sommets x_i et x_j de H sont adjacents, il leur correspond dans H^* des hyperarêtes e_i^* et e_j^* adjacentes ; si deux hyperarêtes e_i et e_j de H sont adjacentes, il leur correspond des sommets x_i^* et x_j^* adjacents de H^*.

Définition 6 TRAVERSE MINIMALE *[Ber89]*

Soit un hypergraphe $H = (\mathcal{X}, \xi)$. L'ensemble des traverses de H, noté γ_H, est égal à : $\gamma_H = \{T \subset \mathcal{X} \mid T \bigcap e_i \neq \emptyset, \forall i = 1, \ldots, \mid \xi \mid\}$.

Une traverse T de γ_H est dite minimale *s'il n'existe pas une autre traverse S de γ_H incluse dans T : $\nexists\ S \in \gamma_H$ s.t. $S \subset T$.*

Nous noterons $\mathcal{M}_H$, l'ensemble des traverses minimales définies sur H.
Dans l'exemple illustratif de la Figure 1.1, l'ensemble $\mathcal{M}_H$ des traverses minimales de l'hypergraphe est : { {1, 4, 7}, {2, 4, 7}, {1, 3, 6, 7}, {1, 3, 6, 9}, {1, 5, 6, 7}, {2, 3, 6, 7}, {2, 3, 6, 9}, {2, 5, 6, 7}, {2, 4, 6, 9}, {2, 4, 8, 9}, {2, 5, 6, 9}, {1, 3, 4, 8, 9}}.

A partir d'un hypergraphe $H = (\mathcal{X}, \xi)$, l'ensemble des traverses minimales $\mathcal{M}_H$ permet la construction de l'hypergraphe transversal, que nous avons noté $H^t = (\mathcal{X}^t, \xi^t)$, tel que $\xi^t = \mathcal{M}_H$ et $\mathcal{X}^t = \bigcup_{i=1}^{|\xi^t|} e_i^t \ \forall\ e_i^t \in \xi^t$ [EG02].

Lemma 1 *[Ber89] Pour tout hypergraphe simple H, nous avons $H^{(t)^{(t)}} = H$.*

Définition 7 NOMBRE DE TRANSVERSALITÉ *[Ber89]*

Soit un hypergraphe $H = (\mathcal{X}, \xi)$, le nombre minimum de sommets d'un ensemble transversal est appelé le nombre de transversalité de l'hypergraphe H et est désigné par :

$$\tau(H) = min\ \{|T|,\ s.t.\ T \in \mathcal{M}_H\}.$$

Ainsi, dans l'exemple illustratif de la Figure 1.1, le nombre de transversalité de l'hypergraphe H est égal à 3 car la plus petite traverse minimale de $\mathcal{M}_H$ renferme 3 sommets.

La détermination d'un nombre de transversalité apparaît dans de nombreux problèmes combinatoires comme la détermination d'un ensemble stable maximum d'un graphe ou encore la détermination d'un ensemble absorbant minimum d'un 1-graphe [Ber89].

1.3 Problème de l'extraction des traverses minimales

L'extraction des traverses minimales d'un hypergraphe est un des problèmes les plus importants en théorie des hypergraphes. C'est un problème algorithmique central et particulièrement difficile et la question de sa complexité exacte reste toujours ouverte. Plusieurs travaux se sont attachés à proposer diverses méthodes pour le traiter [Ber89] [KS05] [BEGK03]. Fredman et Khachiyan ont proposé un algorithme avec une complexité quasi-polynomiale de $N(o^{logN})$ où N représente la taille de l'entrée et de la sortie [FK96]. Ce résultat de Fredman et Khachiyan nous donne un algorithme d'extraction des traverses minimales dont la complexité est bornée par $\mid \mathcal{M}_H \mid \times (\mid \mathcal{M}_H \mid + \mid H \mid)^{o(\mid \mathcal{M}_H \mid + \mid H \mid)}$ [Mar13]. Ce résultat relance le débat sur le fait que ce problème soit coNP-complet puisqu'à moins que tout problème coNP-complet admette un algorithme quasi-polynomial, le problème de l'extraction des traverses minimales n'est pas coNP-complet.

Trouver une traverse minimale d'un hypergraphe est une tâche aisée mais calculer l'ensemble de toutes les traverses minimales pose plusieurs problèmes dans la mesure où le nombre de sous-ensembles de sommets à tester est très important. Les travaux antérieurs, pour faire sauter les différents verrous scientifiques que posait l'extraction des traverses minimales d'un hypergraphe, se sont attachés à réduire l'espace de recherche. Néanmoins, le coût du calcul reste substantiellement élevé et les algorithmes existants se sont heurtés à des temps d'exécution conséquents et à l'incapacité de traitement lorsque le nombre de transversalité de l'hypergraphe d'entrée est grand.

Comme nous l'avons mentionné dans la section précédente, le problème de l'extraction des traverses minimales à partir d'un hypergraphe H est équivalent à celui de la construction de l'hypergraphe transversal à H. Formellement, nous définissions ce problème comme suit :

Entrée : Hypergraphe simple $H = (\mathcal{X}, \xi)$.
Sortie : Hypergraphe transversal $H^t = (\mathcal{X}^t, \xi^t)$.

Même pour des hypergraphes simples, le nombre de traverses minimales d'un hypergraphe peut être exponentiel [Hag08], comme le montre l'exemple 11.

Exemple 4 *Soit* $H = (\mathcal{X}, \xi)$ *un hypergraphe tel que* $\mathcal{X} = (x_1, x_2, \ldots, x_{2n})$ *et* $\xi = (\{x_1, x_2\}, \{x_3, x_4\}, \ldots, \{x_{2n-1}, x_{2n}\})$. H *est de taille* $2n$ *mais renferme* 2^n *traverses minimales.*

La complexité des approches proposées dans la littérature, et décrites dans le chapitre suivant, est analysée en termes de la taille d'entrée et de sortie. Plus concrètement, si $n = |\ \mathcal{X}\ |$, $m = |\ \xi\ |$ et $m' = |\ \mathcal{M}_H\ |$, nous disons qu'un algorithme d'extraction des traverses minimales est polynomial en la taille de l'entrée et de la sortie N si sa complexité peut être bornée, de manière polynomiale, par N, qui désigne une fonction de n, m et m'. En outre, un algorithme est incrémental s'il énumère une par une toutes traverses minimales de l'hypergraphe d'entrée de telle sorte que le temps nécessaire pour délivrer en sortie une nouvelle transverse minimale est polynomiale en n, m et k, k étant la taille de l'hypergraphe transversal.

La notion d'algorithme incrémental a ouvert la voie à une autre approche consistant à ne générer qu'un sous-ensemble de traverses minimales, i.e., une liste partielle de traverses minimales, à partir de l'hypergraphe d'entrée. Formellement, ce problème est défini comme suit :

Entrée : Hypergraphe simple $H = (\mathcal{X}, \xi)$ et un sous-ensemble de traverses minimales $S \subseteq \mathcal{M}_H$.
Sortie : Vrai si $S \subseteq \mathcal{M}_H$, sinon retourner une traverse minimale de $\mathcal{M}_H \setminus S$.

Un troisième problème a été défini par [BI95] et qui se résume à vérifier si deux

hypergraphes sont transversaux l'un par rapport à l'autre.

Entrée : Deux hypergraphes simples $H = (\mathcal{X}, \xi)$ et $H' = (\mathcal{X}', \xi')$.
Sortie : Vrai si $H' = H^t$, Faux sinon.

Les trois problèmes sont liés et divers algorithmes ont été proposés pour les résoudre.

1.4 Domaines d'application

L'intérêt pour l'extraction des traverses minimales s'est accru, ces dernières années, en raison de la diversité des domaines d'application où le recours aux traverses minimales peut constituer une solution. Le large éventail des domaines d'application, comme le résume la Figure 1.3 [Hag08], donne ainsi une importance plus grande aux traverses minimales et motive l'intérêt qu'elles suscitent. Dans ce qui suit, nous en donnerons un aperçu et nous citerons les problèmes les plus connus, où les traverses minimales sont applicables.

1.4.1 Bases de données

Plusieurs travaux se sont attachés à appliquer les traverses minimales pour résoudre des verrous scientifiques dans le domaine des bases de données. Jouant un rôle important dans l'identification, de façon minimale, des n-uplets que renferment les relations, l'identification des clés est fortement liée au problème du calcul des traverses minimales comme l'ont démontré les travaux de [DT99] et de [TS05]. Étant donné une relation et un ensemble de clés, décider de l'existence d'une autre clé est un problème, équivalent à celui de la recherche des traverses minimales. Les dépendances d'inclusion [MP03], qui sont une généralisation des clés étrangères dans un modèle relationnel, peuvent ainsi être déduites en adaptant les techniques de calcul

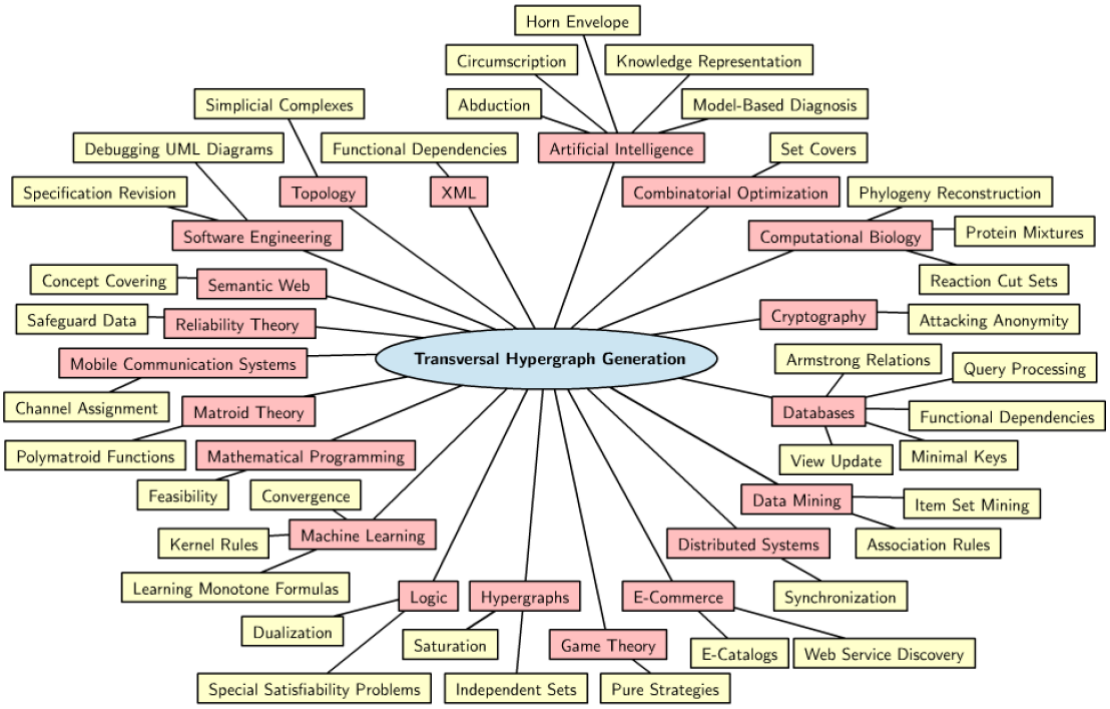


FIGURE 1.3 – Domaines d'application des traverses minimales [Hag08]

des traverses minimales d'un hypergraphe. Celles-ci peuvent, par ailleurs, présenter des solutions aux problèmes de réécriture des requêtes, d'exécution des requêtes et d'actualisation des vues. Ces dernières, dont le rôle est très important dans la présentation des données à partir des bases, peuvent en effet être gérées en se basant sur les traverses minimales. L'inférence des dépendances fonctionnelles représente aussi un domaine d'application fort intéressant des traverses minimales comme le montre les travaux de [MR94] et [LPL00] et comme nous le verrons dans le chapitre 4.

1.4.2 Logique

En logique, une *clause* est une disjonction de littéraux, qui sont des variables booléennes ou leurs négations, alors qu'un *terme* est une conjonction. Une formule

est sous sa forme normale *disjonctive* (FND) (resp. *conjonctive* (FNC)) si c'est une une disjonction de termes (resp. une conjonction de clauses). La détermination de la dualité FNC/FND est équivalente au problème de calcul des traverses minimales d'un hypergraphe. En effet, le problème, connu en logique, de la dualisation où il s'agit de calculer la forme duale normale disjonctive (FND) monotone et irrédondante à partir d'une forme normale disjonctive du même type est équivalent au calcul d'un sous-ensemble des traverses minimales d'un hypergraphe.

1.4.3 Intelligence artificielle

Plusieurs problématiques en intelligence artificielle ont un lien très fort avec les traverses minimales, à commencer par l'abduction. Définie par Peirce, l'abduction est un mode de raisonnement par lequel des faits utiles sont inventés (contrairement à l'induction qui consiste à inventer des théories) et est utilisée dans deux acceptations différentes. Plus formellement, à partir de l'ensemble caractéristique d'une théorie de Horn Σ, d'un littéral q et d'un sous-ensemble A de tous les littéraux, il s'agirait de trouver toutes les interprétations possibles pour q par rapport à A. Dans ce cas, une théorie logique est un ensemble de formules. C'est une Horn s'il s'agit d'un ensemble des clauses ayant, chacune, au plus un littéral positif. Les travaux de Eiter et *al.* [EG02] ont montré que ce problème de l'abduction est équivalent à une adaptation du calcul des traverses minimales d'un hypergraphe.

1.4.4 E-commerce

Que ce soit pour des problématiques telles que la recherche du meilleur recouvrement, dont les applications sont nombreuses, la réécriture des requêtes dans les e-catalogues ou la découverte des web services, les techniques d'extraction des traverses minimales peuvent jouer un rôle important en e-commerce, comme le montre les travaux de [BHL+05]. Ainsi, à titre d'exemple, l'une des tâches les plus impor-

tantes en web services est de trouver automatiquement les services qui répondent à des contraintes d'utilisation spécifiques aux utilisateurs. Ce problème est singulièrement similaire à celui de la recherche d'un ensemble de couverture dans un contexte de contraintes et pour lequel l'application des techniques de calcul des traverses minimales s'avère judicieuse.

1.4.5 Fouille de données

Les liens entre certaines problématiques de la fouille de données et les traverses minimales sont nombreuses. C'est d'ailleurs le domaine où l'application des traverses minimales est la plus intéressante, comme dans la génération des règles associatives, des itemsets fermés, des itemsets fréquents ou encore l'analyse formelle de concepts. Les règles associatives sont de la forme "si x est présent dans une transaction alors il y a 95% de chance que y soit aussi présent". Elles sont cruciales au cours du processus d'extraction des connaissances. Une des étapes les plus importantes dans la génération des règles associatives est le calcul des ensembles fréquents et infréquents. Or, calculer les ensembles maximaux k-fréquents ou les ensembles minimaux k-infréquents est équivalent à la construction d'un hypergraphe transversal comme l'ont démontré les travaux de Boros et *al.* [BGKM03] et, Mannila et Toivonen [MT97]. Par ailleurs, les règles associatives ayant comme prémisses les générateurs minimaux sont les règles les plus intéressantes en fouille de données. Les générateurs minimaux d'un itemset fermé F étant les plus petits itemsets ayant cette même fermeture F, leur calcul peut être optimisé en utilisant les traverses minimales d'un hypergraphe selon les travaux de [Gar06] et de [PT02]. Les treillis de concepts qui jouent un rôle important dans l'extraction de connaissances [PT02] et la génération des règles à partir de bases de données relationnelles [PT02] peuvent aussi bénéficier des techniques d'extraction des traverses minimales en raison de leur relation avec les itemsets fermés.

1.5 Conclusion

Nous avons introduit, dans ce chapitre, les notions-clés de la théorie des hypergraphes tout en mettant en exergue les verrous scientifiques que posent le problème de l'extraction des traverses minimales d'un hypergraphe. Le survol des domaines d'application des traverses minimales démontre l'intérêt, de plus en plus, croissant pour les traverses minimales et, dans la littérature, plusieurs algorithmes dédiés à leur calcul ont été proposés. Dans le chapitre suivant, nous présentons un état de l'art détaillé de ces algorithmes, qui reposent sur des approches différentes.

Chapitre 2

État de l'art

2.1 Introduction

Plusieurs auteurs se sont intéressés au problème de l'extraction des traverses minimales d'un hypergraphe. Dans ce chapitre, nous présentons un état de l'art détaillé de ces différentes approches, en mettant en exergue leurs points forts et leurs limites. Le nombre de traverses minimales d'un hypergraphe pouvant être exponentiel en la taille de l'hypergraphe, la question de la mise en place d'un algorithme résolvant le problème de l'extraction des traverses minimales d'un hypergraphe H avec une complexité polynomiale en $|H|$ reste néanmoins ouverte. Une étude comparative des différents algorithmes existants nous permet de situer nos contributions par rapport à ces travaux et mettre en lumière l'intérêt des différentes approches que nous proposons dans les chapitres suivants.

2.2 Algorithme de Berge [Ber89]

Berge est le premier à s'être intéressé au problème du calcul des traverses minimales et à avoir proposé un algorithme pour le résoudre. Cet algorithme, dont

le principe est simple, commence par calculer l'ensemble des traverses minimales de la première hyperarête, qui est équivalent à l'ensemble des sommets contenus dans cette dernière. Ensuite, il met à jour cet ensemble des traverses minimales en ajoutant les autres hyperarêtes, une à une, de manière incrémentale. Ainsi, l'algorithme de Berge construit des hypergraphes partiels au fur et à mesure qu'il ajoute des hyperarêtes. Néanmoins, l'algorithme a toujours besoin de stocker les traverses minimales intermédiaires avant de passer à l'étape suivante consistant à ajouter une nouvelle hyperarête.

Formellement, l'algorithme de Berge repose sur la formule suivante [Ber89] : à partir d'un hyperpgraphe partiel $H_i = (\mathcal{X}, \xi)$ tel que $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ et $\xi = \{e_1, e_2, \ldots, e_i\}$, l'ensemble des traverses minimales de $\{H_{i-1} \cup e_i\} = \{min\{T \times \{x\}\}$ tel que $T \in \mathcal{M}_{H_{i-1}}$ et $x \in e_i\}$. L'opérateur "$\times$" désigne le produit cartésien tel que $A \times B$ comporte toutes les pairs (a,b) tel que $a \in A$ et $b \in B$.

Algorithme 1: L'algorithme de BERGE [Hag08]

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe

Sorties : $\mathcal{M}_H$: ensemble des traverses minimales de H

1 **début**

2 $\quad \mathcal{M}_{H_1} = \{\{x\} \mid x \in \xi_1\}$;

3 $\quad$ **pour** $i = 2 \rightarrow \mid \xi \mid$ **faire**

4 $\quad\quad \mathcal{M}_{H_i} = \text{Min } \{T \times \{x\}$ tel que $T \in \mathcal{M}_{H_{i-1}}, x \in e_i\}$

5 $\quad \mathcal{M}_H = \mathcal{M}_{H_{|\xi|}}$;

6 $\quad$ **retourner** $\mathcal{M}_H$

La complexité de cet algorithme, dont le pseudo-code est donné par l'Algorithme 1 est exponentielle en la taille de l'entrée et de la sortie [Hag08]. Ceci s'explique par la nécessité de stocker toutes les traverses intermédiaires vu que l'ensemble des traverses minimales n'est généré qu'après l'insertion de la dernière hyperarête. Ceci

rend l'algorithme de BERGE impraticable sur des hypergraphes de grande taille.

Récemment, Boros *et al.* ont prouvé que le temps de traitement de l'algorithme de Berge a une borne supérieure subexponentielle de $N^{\sqrt{N}}$ [BEM08].

2.3 Améliorations de l'algorithme de BERGE

Plusieurs chercheurs ont cherché à améliorer l'algorithme de BERGE. Parmi les améliorations les plus connues proposées récemment figurent celles introduites par Dong *et al.* [DL05], Kavvadias et Stavropoulos [KS05] et Bailey *et al.* [BMR03].

Algorithme 2: L'algorithme de Dong et Li [Hag08]

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe

Sorties : $\mathcal{M}_H$: ensemble des traverses minimales de H

1 **début**
2 $\quad \mathcal{M}_{H_1} = \{\{x\} \mid x \in \xi_1\}$;
3 $\quad$ **pour** $i = 2 \rightarrow \mid \xi \mid$ **faire**
4 $\quad\quad T_g = \{t \in \mathcal{M}_{H_{i-1}} \mid t \cap e_i \neq \emptyset\}$;
5 $\quad\quad e_i^{cov} = \{x \in e_i \mid \{x\} \in T_g\}$;
6 $\quad\quad \mathcal{M}'_{H_{i-1}} = \mathcal{M}_{H_{i-1}} \backslash T_g$;
7 $\quad\quad e'_i = e_i \backslash e_i^{cov}$;
8 $\quad\quad$ **pour chaque** $t' \in \mathcal{M}'_{H_{i-1}}$ *trié par ordre croissant de cardinalité* **faire**
9 $\quad\quad\quad$ **pour chaque** $x \in e'_i$ **faire**
10 $\quad\quad\quad\quad$ **si** $t' \cup \{x\}$ *n'est le sur-ensemble d'aucun élément de* T_g **alors**
11 $\quad\quad\quad\quad\quad T_g = T_g \cup \{t' \cup \{x\}\}$
12 $\quad\quad \mathcal{M}_{H_i} = T_g$;
13 $\quad \mathcal{M}_H = \mathcal{M}_{H_{|\xi|}}$;
14 $\quad$ **retourner** $\mathcal{M}_H$

2.3.1 Algorithme de Dong et Li [DL05]

C'est en s'inspirant de l'extraction des itemsets émergeants en fouille de données que Dong et Li ont proposé une solution, dont le pseudo-code est donné par l'Algorithme 2.

L'algorithme de Dong et Li a été évalué expérimentalement sur de nombreux jeux de données. Néanmoins, les auteurs n'ont pas effectué une analyse théorique de la complexité en temps de traitement de leur algorithme. Cependant, cette adaptation de l'algorithme initial s'est avérée très fructueuse. La principale amélioration de Dong et Li par rapport à l'algorithme de Berge réside dans l'optimisation réalisée lors du calcul de $\mathcal{M}_{H_{i-1}} \times \{\{x\} \mid x \in e_i\}$, et qui consiste à considérer uniquement les traverses qui intersectent la nouvelle hyperarête traitée et, aussi, à ne prendre que les sommets de ξ_i qui n'appartiennent pas déjà aux traverses minimales déjà identifiées.

2.3.2 Algorithme de Kavvadias et Stavropoulos [KS05]

L'un des inconvénients majeurs de l'algorithme de BERGE, observé par Kavvadias et Stavropoulos, est la consommation excessive en mémoire. Dans la mesure où la minimalité des nouvelles traverses calculées doit être testée, les traverses minimales intermédiaires doivent aussi être stockées en mémoire. L'algorithme de Kavvadias et Stavropoulos tente de surmonter ce problème de consommation mémoire, en utilisant deux techniques. La première introduit la notion de "*sommets généralisés*" selon la Définition 8.

Définition 8 *Soit $H = (\mathcal{X}, \xi)$ un hypergraphe. L'ensemble $X \subseteq \mathcal{X}$ est un ensemble de sommets généralisés de H si tous les sommets de X appartiennent aux mêmes hyperarêtes de ξ.*

Le pseudo-code de l'algorithme de Kavvadias et Stavropoulos est décrit par l'Algorithme 3. Le principe est le suivant. En ajoutant une hyperarête e_i, l'algorithme met à jour l'ensemble des sommets généralisés avant de considérer les éléments de $M_{H^g_{i-1}}$ et les sommets constituant ξ_i comme les ensembles de sommets généralisés du niveau i. H^g_{i-1} étant l'hypergraphe partiel composé uniquement des sommets généralisés calculés au niveau $i-1$, les traverses minimales de l'hypergraphe H^g_i, $M_{H^g_i}$, sont ensuite calculées selon la formule de Berge, i.e., en effectuant le produit cartésien entre $M_{H^g_{i-1}}$ et les sommets généralisés de ξ^g_i, et en testant la minimalité de ces traverses candidates.

La seconde technique introduite par Kavvadias et Stavropoulos pour diminuer la consommation mémoire élevée revient à adopter une stratégie de recherche en profondeur d'abord. Berge utilisait une forme de parcours en largeur d'abord à travers la construction d'un "arbre" de traverses minimales. Au i-ème niveau de l'arbre, les noeuds sont des traverses minimales de l'hypergraphe partiel H_i. Les descendants d'une traverse minimale T, du niveau i, sont les traverses minimales de l'hypergraphe H_{i+1} incluant T. Le parcours de cet "arbre" est très coûteux puisque les traverses minimales d'un hypergraphe H sont retrouvées dans le dernier niveau de l'arbre. De plus, certains noeuds sont "visités" plusieurs fois parce qu'ils peuvent avoir plusieurs parents.

Pour remédier à ce problème, Kavvadias et Stavropoulos utilisent une stratégie en profondeur d'abord et introduisent la notion de "*sommets appropriés*" pour vérifier la minimalité des traverses générées. Ceci permet à l'algorithme de réduire considérablement le stockage en mémoire durant le calcul des traverses minimales d'un hypergraphe.

Définition 9 *Soit un hypergraphe $H = (\mathcal{X}, \xi)$ et soit T une traverse minimale de l'hypergraphe partiel H_{i-1} de H. Un ensemble de sommets généralisés $X \subseteq \mathcal{X} \setminus T$, au niveau i est un ensemble de sommets appropriés pour T si aucun sous-ensemble*

de $T \cup X$, excepté X, ne peut être supprimé sans que les sommets restants ne représentent plus une traverse.

Algorithme 3: L'algorithme de Kavvadias et Stavropoulos [KS05]

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe

Sorties : $\mathcal{M}_H$: ensemble des traverses minimales de H

début
  pour $k = 0 \rightarrow \mid \xi \mid$ **faire**
    Ajouter_hyperarête(ξ_{k+1}) ;
    Mettre à jour les sommets généralisés ;
    Considérer $M_{H_k^g}$ et ξ_{k+1} comme étant des sommets généralisés du niveau $k+1$;
    Calculer $M_{H_{k+1}^g} = Min\{M_{H_k^g} \times \{\{x_i\} : x_i \in \xi_{k+1}^g\}\}$;
  Déduire M_H à partir de $M_{H_{|\xi|}}$;
  retourner M_H

L'algorithme de Kavvadias et Stavropoulos n'est pas polynomial en la taille de la sortie. Son temps de traitement est de l'ordre de $N^{\Omega(loglogN)}$, N désignant la taille de l'entrée et de sortie [Hag08]. Cet algorithme est l'un des plus performants, en termes de temps de traitement. Adoptant une stratégie en profondeur, l'algorithme consomme, par ailleurs, très peu de mémoire vive. Ce qui représente un avantage non négligeable.

2.3.3 Algorithme de Bailey *et al.* [BMR03]

Pour traiter les hypergraphes de grande taille, Bailey *et al.* ont exploité les bonnes performances de l'algorithme de Dong et Li sur les hypergraphes renfermant des hyperarêtes de petite taille. L'algorithme de Bailey *et al.*, dont le pseudo-code est

décrit par l'Algorithme 4, prend en entrée un hypergraphe et comporte un pré-traitement récursif.

Algorithme 4: L'algorithme de Bailey *et al.* [Hag08]

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe

Sorties : $\mathcal{M}_H$: ensemble des traverses minimales de H

début
  $X = \mathcal{X}$;
  Ordonner($\mathcal{X}$) ;
  pour $i = 1 \rightarrow \mid \mathcal{X} \mid$ **faire**
    $\xi_{part} = \emptyset$;
    $X = X \setminus x_i$
    pour chaque $e \in \xi$ **faire**
      si $x_i \notin e$ **alors**
        $\xi_{part} = \xi_{part} \cup \{\mathrm{e} \setminus X\}$;
    $\mathcal{X}_{part} = \mathcal{X}_{part} \cup x_i$;
    si $\mid \xi_{part} \mid \geq 2$ *and* $Volume(\xi_{part}) \geq 50$ **alors**
      Algorithme 4 ($\mathcal{X}_{part}$, ξ_{part}) ;
    sinon
      $\mathcal{M}_{\xi_{part}}$ = Algorithme 2(ξ_{part}) ;
      $\mathcal{M}_H = \min(\mathcal{M}_H \cup (\mathcal{M}_{\xi_{part}} \times \mathcal{X}_{part}))$;
    $\mathcal{X}_{part} = \mathcal{X}_{part} \setminus \{x_i\}$
  retourner Tr

A partir d'un sommet ou d'un ensemble de sommets X_{part} apparaissant dans le plus petit nombre d'hyperarêtes dans l'hypergraphe d'entrée, ce pré-traitement vise à construire un sous-ensemble d'hyperarêtes ξ_{part} ne contenant pas ces sommets X_{part}.

Si | ξ_{part} | est elevée (≥ 2) et si son volume, fonction de la cardinalité moyenne de ces hyperarêtes est aussi elevée (≥ 50) alors l'algorithme de Bailey *et al.* est appelé de manière récursive. Sinon, les traverses minimales de ξ_{part} sont déterminées par l'algorithme de Dong et Li. Les traverses minimales de l'hypergraphe d'entrée sont ensuite déduites par la méthode de Berge via un produit cartésien entre les traverses minimales de l'hypergraphe constitué par les hyperarêtes ξ_{part} et X_{part}, conjugué à un test de la minimalité. Les expérimentations menées par les auteurs ont montré l'efficacité de leur algorithme, sur un type particulier d'hypergraphes, par rapport aux deux algorithmes considérés, i.e., celui de Fredman et Kachiyan, et celui de Kavvadias et Stavropoulos (une version antérieure à celle présentée dans la section 2.3.2) .

2.4 Algorithme de Fredman et Kachiyan [FK96]

En 1996, Fredman et Kachiyan ont proposé deux algorithmes pour le calcul des traverses minimales d'un hypergraphe, l'algorithme FK-A et sa version optimisée, appelée l'algorithme FK-B. Ce dernier possède la meilleure complexité théorique connue à ce jour et qui est de $N(o^{logN})$ où N représente la taille de l'entrée et de sortie [FK96]. Les auteurs motivent le calcul de traverses minimales comme la solution au problème de la dualisation des formules booléennes monotones et c'est cette approche intuitive que nous reprenons. Étant donnée une formule $f(x) = f(x_1, x_2, \ldots, x_n)$ sous forme normale conjonctive, il s'agit de calculer la formule duale correspondante $f^d(x) = \bar{f}(\bar{x}) = \bar{f}(\bar{x_1}, \bar{x_2}, \ldots, \bar{x_n})$ sous forme normale conjonctive également. Pour cela, on obtient aisément f^d sous forme normale disjonctive en remplaçant chaque conjonction de f par une disjonction et vice-versa. Pour calculer la forme normale conjonctive de la formule duale, il s'agit finalement de développer la forme normale disjonctive pour constituer les classes de f^d. Pour cela, on prendra un littéral dans chaque terme de $\bar{f}$ pour constituer une classe. Des simplifications apparaissent si

l'on prend plusieurs fois le même littéral. Pour calculer les traverses minimales d'un hypergraphe, chaque traverse est construite en prenant un item dans chaque terme de $\bar{f}$: le résultat obtenu est identique à celui fourni par la dualisation des formules booléennes monotones. Ce problème et celui du calcul des traverses minimales d'un hypergraphe sont alors parfaitement équivalents.

Par exemple, soit $f(x) = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3 \vee x_4)$. La formule duale correspondante, obtenue en échangeant chaque conjonction par une disjonction et vice-versa, est $f^d(x) = (x_1 \wedge x_2) \vee (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2 \wedge x_3 \wedge x_4)$. Si nous développons scrupuleusement cette dernière expression pour la transformer en forme normale conjonctive, on obtient la série de clauses suivantes : $f^d(x) = (x_1 \vee x_1 \vee x_1 \vee x_2 \vee x_1) \wedge (x_1 \vee x_1 \vee x_1 \vee x_2 \vee x_2) \wedge \ldots \wedge (x_2 \vee x_3 \vee x_4 \vee x_4 \vee x_4)$. Ceci donne alors 216 clauses, dont il n'en restera que trois, après les simplifications : $f^d(x) = x_2 \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_4)$.

La solution proposée par les auteurs revient à determiner, de façon incrémentale, si deux formules f et g ne sont pas duales, i.e., $f(x) = \bar{g}(\bar{x})$.

La vérification de la dualité, dans l'algorithme de Fredman et Khachiyan dont le pseudo-code est donné par l'Algorithme 5, et la mise en évidence d'un disqualifieur sont effectuées grâce à la propriété suivante : en factorisant f et g selon une variable x_i, les auteurs font apparaître des formules plus courtes, f_0, f_1, g_0 et g_1 qui ne contiennent pas x_i. On obtient ainsi $f(x) = (x_i \wedge f_0(y)) \vee f_1(y)$ puis $g(x) = (x_i \wedge g_0(y)) \vee g_1(y)$ (y ne contient pas le littéral x_i). f et g sont duales si et seulement si f_1 et $g_0 \vee g_1$ le sont, ainsi que $f_0 \vee f_1$ et g_1. La taille du problème est ainsi réduite et permet d'appliquer récursivement ce procédé. Néanmoins, cette méthode est peu adaptée au calcul des traverses minimales de longueur bornée.

Algorithme 5: L'algorithme FK-A [FK96]

Entrées : Deux formules monotones sous la Forme Normale Disjonctive f et g

Sorties : Le Dual de f et g

début

  Factoriser f et g

  Vérifier que f et g sont des formes mutuellement duales et que le problème peut se résoudre en un temps polynomial.

  si $\mid F \mid \mid G \mid \leq 1$ **alors**

    le dual de f et g est calculé en $O(1)$.

  si $\mid F \mid \mid G \mid \geq 2$ **alors**

    Trouver une variable x_i commune à f et g tel que Fréquence$(x_i) \geq 1/log(\mid F \mid + \mid G \mid)$

    $f = x_i f_0 \vee f_1$;

    $g = x_i g_0 \vee g_1$;

    FK$(f_1,\ g_0 \vee g_1)$;

    FK$(g_1,\ f_0 \vee f_1)$;

2.5 Algorithme MtMiner [HBC07]

L'algorithme proposé par Hébert *et al.* consiste à exploiter les travaux réalisés, dans la littérature, sur l'extraction de motifs [HBC07]. Les auteurs ont réutilisé le principe des algorithmes par niveaux pour calculer les traverses minimales d'un hypergraphe. Cette approche repose donc sur le fait que les bases de données et les hypergraphes peuvent se représenter de la même manière, i.e, sous la forme d'une matrice booléenne où les sommets correspondent aux motifs et les hyperarêtes aux objets.

Par le biais de la correspondance de Galois qui relie les ensembles de motifs et

les ensembles d'objets, un parallèle est établi entre l'extraction des motifs et l'extraction des traverses minimales. L'extension de cette nouvelle connexion permet de définir des classes d'équivalence, pour un hypergrpahe, de façon analogue aux classes d'équivalence utilisées en fouille de motifs, selon la définition 10 [Héb07].

Ces classes regroupent les ensembles de sommets appartenant aux mêmes hyperarêtes de l'hypergraphe d'entrée H. Le nombre d'hyperarêtes non recouvertes par un ensemble de sommets est appelé *fréquence* et correspond alors au nombre d'occurrences (support disjonctif) d'un motif en fouille de données. Les traverses de H sont, dans ce cas, les ensembles de sommets ayant une fréquence nulle. En utilisant les propriétés de la fouille de données, les auteurs ont prouvé que les traverses minimales de H sont les générateurs minimaux de fréquence nulle.

Définition 10 *La classe d'équivalence d'un ensemble de sommets $X \subseteq \mathcal{X}$ est notée $\mathcal{R}_{gH}(X)$ et est définie comme suit :*
$\mathcal{R}_{gH}(X) = \{X' \in \mathcal{X} \mid gH(X') = gH(X)\}$ où $gH(X)$ est l'ensemble des hyperarêtes qui ne contient aucun sommet de X.

L'algorithme MtMiner adopte deux stratégies d'élagage lors du parcours par niveau des candidats dans le treillis généré. La première repose sur la propriété d'anti-monotonie de la minimalité dans les classes d'équivalence, selon laquelle si un ensemble de sommets ne constitue pas un générateur minimal alors l'espace de recherche généré à partir de celui-ci est élagué.

En effet, si un ensemble de sommets n'est pas un générateur minimal alors aucun de ses sur-ensembles ne peut être aussi un générateur minimal. Comme les auteurs ont déjà montré qu'une traverse minimale est nécessairement un générateur minimal, il est inutile de considérer ces sur-ensembles. La deuxième stratégie d'élagage consiste à éliminer les sur-ensembles d'un ensemble de sommets qui est une traverse minimale, puisqu'ils ne vérifient pas la condition de minimalité au sens de l'inclusion.

L'algorithme MtMiner effectue un parcours en largeur en démarrant le balayage

Algorithme 6: L'algorithme MtMiner [Héb07]

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe

Sorties : $\mathcal{M}_H$: ensemble des traverses minimales de H

```
1  début
2      M_H = {{x} | x ∈ X et |gH({x})| = 0} ;
3      Gen_1 = {{x} | x ∈ X et 0 < |gH({x})| < |ξ| } ;
4      k = 1 ;
5      tant que Gen_k ≠ ∅ faire
6          pour chaque (X∪ {x_1}, X∪ {x_2}) ∈ Gen_k × Gen_k faire
7              Z = X ∪ x_1 ∪ x_2 ;
8              gH(Z) = gH(X∪ {x_1}) ∩ gH(X∪ {x_2}) ;
9              i = 1 ;
10             tant que i ≤ k+1 et Z\{x_i} ∈ Gen_k et | gH(Z) | < | gH(Z\ {x_i})
               | faire
11                 i = i+1 ;
12             si i = k + 2 alors
13                 si | gH(Z) | = 0 alors
14                     M_H = M_H ∪ Z ;
15                 sinon
16                     Gen_{k+1} = Gen_{k+1}∪ {Z} ;
17         k = k + 1 ;
18     retourner M_H
```

de l'espace de recherche par les sommets, avant de générer les ensembles plus grands, suivant une approche inspirée de Apriori [AR94]. La stratégie en largeur permet de garantir la minimalité des candidats, dans la mesure où chaque candidat n'est gardé

dans un niveau i qu'après avoir calculé et testé l'extension de ses sous-ensembles directs, qui se trouvent dans le niveau *i-1*.
L'ensemble $\mathcal{M}_H$ est initialisé avec les sommets d'extension vide, i.e, qui appartiennent à toutes les hyperarêtes de l'hypergraphe d'entrée ($|gH(\{x\})| = 0$). Ces sommets représentent donc des traverses minimales du niveau 1. A chaque niveau i, MTMINER génére des candidats Z, à partir des éléments calculés au niveau *i-1*. Si le candidat Z vérifie la propriété d'anti-monotonie et si $|gH(\{Z\})| = 0$, il est alors ajouté à $\mathcal{M}_H$, sinon il sera reversé dans Gen_{i+1} et servira comme générateur pour le niveau *i+1*. Si Z ne vérifie pas la propriété d'anti-monotonie, il n'est donc pas un générateur minimal et il est tout simplement élagué de l'espace de recherche.

D'après Hébert *et al.*, la complexité de l'algorithme MTMINER dépend de $\tau(H)$ et $|\mathcal{M}_H|$ [Héb07]. Pour chaque traverse minimale T, l'algorithme considère au plus $2^{|T|}$ ensembles de sommets et effectue, par conséquent, un nombre d'opérations inférieur à : $\sum_{T \in \mathcal{M}_H}(2^{|T|})$.

De ce fait, pour un hypergraphe H, MTMINER calcule l'ensemble des traverses minimales $\mathcal{M}_H$ en $O(2^{\tau(H)} \times |\mathcal{M}_H|)$. Cependant, pour Hagen, la complexité réelle de l'algorithme MTMINER est de $O(N^{\Omega(log(logN))})$, telle que N est la taille de l'entrée et de la sortie de l'algorithme. Hagen présente le calcul détaillé de cette complexité dans [Hag08]. D'un point de vue performances, MTMINER présente des temps de traitements intéressants, notamment pour des hypergraphes denses et ayant un nombre de transversalité assez bas. Cependant, et comme souligné par les auteurs, MTMINER est assez gourmand en consommation mémoire [Elb08].

2.6 Algorithmes de type SHD [MU13]

Murakami et Uno proposent les algorithmes de type SHD, MMCS et RS, qui visent à réduire l'espace de recherche [MU13]. En ce sens, ces algorithmes sont destinés à traiter des hypergraphes de grande taille constitués par un très grand nombre

d'hyperarêtes.

Les algorithmes de type SHD adoptent une stratégie de parcours en profondeur de l'espace de recherche qui, dans le cas de RS, est équivalente à celle de l'algorithme de Kavvadias et Stavropoulos. La principale différence entre ce dernier et RS repose sur l'élimination des itérations redondantes où aucun sommet n'est ajouté à un ensemble de sommets générés auparavant. De plus, Murakami et Uno introduisent deux nouveaux concepts, i.e, le test de la transversalité (*uncov*) et les hyperarêtes critiques (*crit*), et ce afin d'optimiser les tests sur la minimalité effectués sur l'ensemble des traverses générées.

Algorithme 7: L'algorithme MMCS [MU13]

Var. Globale : *uncov* (initialisé à ξ), *Cand* (initialisé à $\mathcal{X}$), *crit*[x] initialisé à $\emptyset$ pour chaque x

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe, X : ensemble de sommets

Sorties : T tel que $T \in \mathcal{M}_H$

```
1  début
2      si uncov = ∅ alors
3          retourner X
4      Choisir une hyperarête e à partir de uncov ;
5      C = Cand ∩ e ;
6      Cand = Cand \ C ;
7      pour chaque x ∈ C faire
8          UPDATE_CRIT_UNCOV(x, crit, uncov) ;
9          si crit(f, X ∪ x) ≠ ∅ pour chaque f ∈ X alors
10             MMCS(X ∪ x) ;
11             Cand = Cand ∪ x ;
12         Restaurer les valeurs de crit et uncov d'avant la ligne 8 ;
```

Étant donné X un ensemble de sommets, éventuellement réduit à un singleton, *uncov(X)* désigne l'ensemble des hyperarêtes que n'intersectent pas X, i.e, *uncov(X)* = $\{e \in \xi, e \cap X = \emptyset\}$. X est une traverse si et seulement si *uncov(X)* = $\emptyset$.

Pour un sommet $x \in X$, une hyperarête $e \in \xi$ est dite *critique* pour x si $X \cap e = \{x\}$. L'ensemble des hyperarêtes critiques pour x est noté *crit(x, X)*, i.e., *crit(x, X)* = $\{e \mid e \in \xi, e \cap X = \{x\}\}$. Ainsi, un ensemble de sommets $X \subseteq \mathcal{X}$ est une traverse si *uncov(X)* = $\emptyset$ et c'est une traverse minimale si, en plus, *crit(x, X)* = $\not\emptyset$ $\forall x \in X$.

Par ailleurs, si X est une traverse, alors si le sommet x n'a aucune hyperarête critique, chaque $e \in \xi$ renferme un sommet de X, autre que x, et $X \setminus x$ est alors une traverse. Ceci est résumé par la propriété 2 proposée par les auteurs.

Propriété 2 *X est une traverse minimale si et seulement si* $\text{uncov}(X) = \emptyset$ *et* $\text{crit}(x, X) \neq \emptyset$, $\forall\, x \in X$.

Les auteurs proposent aussi divers lemmes, dans [MU13], pour optimiser le calcul de la fonction *crit*, qui est la clé de leur approche.

Les algorithmes de type SHD se basent donc sur la même approche et l'Algorithme 7 décrit le pseudo-code de l'algorithme MMCS. Cet algorithme est récursif et fournit en sortie des traverses minimales en série. Pour tester un ensemble de sommets X, les algorithmes cherchent, de façon itérative, les sous-ensembles de X et effectuent un appel récursif pour chacun tout en mettant à jour les ensembles *crit* et *uncov*. En opérant de cette manière, Murakami et Uno permettent à leur algorithme de balayer l'espace de recherche en profondeur en cherchant seulement les sous-ensembles du candidat courant. La méthode et les étapes pour la recherche des sous-ensembles d'un candidat sont détaillées dans [MU13]. L'étude expérimentale effectuée par les auteurs a montré que les algorithmes de type SHD (et notamment MMCS) présentaient des performances très intéressantes et s'imposaient comme les algorithmes les plus performants dans la littérature.

2.7 Algorithme de Toda [Tod13]

L'algorithme de Toda est le plus récent dans la littérature [Tod13]. Cet algorithme fait appel à des structures de données compressées qui permettent d'exploiter les capacités des diagrammes de décision binaire (BDD) et une des améliorations de ces dernières, i.e., les zéro diagrammes supprimés de décision (ZDD). Les diagrammes de décision binaires permettent de représenter des fonctions booléennes sous la forme de graphes orientés. Leurs principes et mécanismes de fonctionnement sont décrits dans les travaux de [Ake78] et [BRB90].

Toda se base sur les travaux de Donald Knuth sur les ZDD et tente de les additionner aux BDD pour optimiser son algorithme. L'intérêt des BDD dans l'algorithme de Toda se trouve dans la représentation des résultats intermédiaires. Comme le montre le pseudo-code de l'Algorithme 8, Toda génère d'abord les traverses avant de tester leur minimalité. Les traverses candidats sont compressées et stockées dans un BDD avant que le ZDD généré ne fournisse les traverses minimales souhaitées. Ainsi, dans l'Algorithme 8, $\mathcal{S}(p)$ dénote la famille des ensembles d'un BDD (ou un ZDD).

Algorithme 8: L'algorithme de Toda [Tod13]

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe

Sorties : $\mathcal{M}_H$

1 **début**

2 $p =$ Compresser ξ en un ZDD ;

3 Calculer le BDD q pour toutes les traverses de $\mathcal{S}(p)$;

4 Calculer le ZDD r pour tous les ensembles minimaux dans $\mathcal{S}(q)$;

5 Décompresser r en un ensemble $\xi^\star$;

6 **retourner** $\xi^\star$

Les expérimentations effectuées par Toda visent à comparer son algorithme à celui de Murakami et Uno, présenté dans la section 2.6. L'etude expérimentale a montré

que l'algorithme de Toda est compétitif, y compris sur les bases éparses. Ceci peut être expliqué par les capacités qu'offrent les ZDD sur ce type de bases.

2.8 Les traverses minimales approchées

Nous avons présenté, dans le chapitre précédent, le problème de l'extraction des traverses minimales d'un hypergraphe comme étant un problème NP-difficile. En ce sens, à côté des algorithmes présentés plus haut, d'autres travaux se sont intéressés à la recherche des traverses minimales approchées dans le but de contourner la difficulté du problème [AvG09] [DQ13]. Ces travaux, assez rares toutefois, s'intéressent à une sous-classe des traverses minimales dont les éléments n'intersectent pas toutes les hyperarêtes de l'hypergraphe d'entrée. Ainsi, dans [AvG09], les auteurs se sont basés sur une approche évolutionnaire où la transversalité et la minimalité sont transcrites dans une fonction objective. D'autres approches introduisent un certain nombre d'exceptions liées à la transversalité pour générer les traverses minimales approchées. Récemment, Durand *et al.* ont présenté dans [DQ13] l'algorithme δ-MTMINER qui permet de calculer les transverses minimales approchées. L'algorithme prend en compte un nouveau paramètre, δ, qui correspond au nombre des hyperarêtes qu'une traverse minimale approchée pourrait ne pas intersecter.

2.9 Discussion

A la lumière de notre description des principaux algorithmes d'extraction des traverses minimales d'un hypergraphe et avec pour objectif de situer nos contributions, présentées dans les chapitres suivants, par rapport à ces travaux, nous avons synthétisé les caractéristiques de ces algorithmes dans le tableau 2.1. Les critères que nous avons choisis pour distinguer les différentes approches sont le principe, sur lequel se base chaque algorithme, la stratégie d'exploration et les techniques d'élagages.

Algorithme	**Algos sous-jacents**	**Stratégie d'exploration**	**Techniques d'élagages**
Berge	- Processus incrémental	En largeur	Aucune
Dong et Li	- Algorithme 1 de Berge - Itemsets emergeants	En largeur	Traverses garanties T_g Couvertures d'hyperarêtes e^{cov}
Kavvadias et Stavropoulos	- Algorithme 1 de Berge	En profondeur	*Sommets généralisés* *Sommets appropriés*
Bailey *et al.*	- Algorithme 1 de Berge et Algorithme 2 de Dong et Li - Partitionnement des hyperarêtes	En largeur	-
FK	- Dualisation des formules booléennes monotones	-	Dualité Mutuelle
MtMiner	- Extraction de motifs	En largeur	Anti-monotonie de la minimalité
Mmcs	- Itemsets fermés	En profondeur	*uncov* et *crit*
Toda	- Structures de données compressées	En profondeur	Caractéristiques des bdd et zdd

Table 2.1 – Caractéristiques des algorithmes de l'état de l'art

Les caractéristiques des principaux algorithmes d'extraction des traverses minimales ont été établies à partir des différentes sections présentées dans ce chapitre. La première constatation qui se dégage de ce tableau est qu'aucune approche n'a mis à profit la notion de nombre de transversalité, introduite par la Définition 7. Cette notion qui donne une indication claire sur le nombre minimum de sommets formant une traverse minimale peut être intéressante, notamment en adoptant une stratégie en largeur pour cibler directement le niveau qui contient ces plus petites traverses minimales. Notre première contribution, qui consiste à détecter les multi-membres d'un réseau social et qui correspondent à des plus petites traverses minimales d'un hypergraphe représentant les différentes communautés d'un réseau social, se base sur cette notion de nombre de transversalité pour optimiser l'extraction des plus petites traverses minimales à travers un algorithme appelé OM2D que nous introduisons dans le chapitre suivant.

Une deuxième constatation concerne les éléments générés en sortie par les différents algorithmes. Tous ces derniers calculent toutes les traverses minimales et leurs cardinalités sont généralement très importantes. Ce nombre de traverses minimales pouvant s'avérer exponentiel en la taille de l'hypergraphe, nous nous sommes intéressés à chercher une forme d'irrédondance dans l'ensemble des traverses minimales. Le fait de représenter cet ensemble de manière concise et exacte améliore le temps de traitement nécessaire à l'extraction de toutes les traverses minimales. En outre, et partant du fait que les traverses minimales apportent des solutions dans de nombreuses applications, comme présenté dans la section 1.4 du chapitre 1 (page 14), cette représentation concise a des répercussions directes sur l'optimisation de bien d'autres problématiques. Notre deuxième contribution, présentée dans le chapitre 4, met en avant cette notion d'irredondance qui se cache dans les traverses minimales et l'illustre en présentant son impact sur le problème du calcul de la couverture minimale des dépendances fonctionnelles en bases de données.

Enfin, notre troisième contribution consiste en une optimisation de l'extraction des traverses minimales en adoptant la stratégie *diviser pour régner*. Cette stratégie a été utilisée par l'algorithme 4 de Bailey *et al.* sauf que les auteurs se sont focalisés sur le partitionnement des hyperarêtes quand celles-ci sont composées d'un nombre important de sommets. Notre idée se base, plutôt, sur le partitionnement de l'hypergraphe d'entrée en k hypergraphes partiels tel que k est égal au nombre de transversalité. De cette manière, nous éliminons le test coûteux de la minimalité sur les traverses formées par k sommets. Cette approche s'est avérée fructueuse mais, seulement, sur un certain type d'hypergraphes. Une étude détaillée est présentée dans le chapitre 5.

2.10 Conclusion

Dans ce chapitre, nous avons présenté les principaux algorithmes de calcul des traverses minimales, proposés dans la littérature. Que ce soit en adoptant une stratégie en largeur d'abord ou en profondeur d'abord, les différentes approches ont tenté d'innover avec pour but commun d'optimiser l'extraction des traverses minimales. En profitant de la notion de nombre de transversalité, inutilisée jusque-là, nous nous intéressons, dans le chapitre suivant, à une classe particulière des traverses minimales et à son application dans les systèmes communautaires.

Chapitre 3

Identification des multi-membres dans un réseau social

3.1 Introduction

Avec l'expansion des systèmes communautaires du Web 2.0, beaucoup de travaux se sont intéressés à identifier les membres clés dans les réseaux sociaux, qualifiés, selon les auteurs, d'influenceurs, de médiateurs, d'ambassadeurs ou encore d'experts. Ce problème a été notamment considéré comme un problème de maximisation. Dans ce chapitre, nous présentons un type particulier d'acteurs que nous appelons multi-membres, en raison de leur appartenance à plusieurs communautés. Nous introduisons alors un cadre méthodologique pour identifier ce type d'acteurs dans un hypergraphe, dans lequel les sommets sont les acteurs et les hyperarêtes représentent les communautés. Nous démontrons que détecter les multi-membres pourrait être ramené au problème d'extraction des traverses minimales à partir d'un hypergraphe et nous présenterons deux algorithmes d'extraction des multi-membres qui conjuguent des concepts de la fouille de données et de la théorie des hypergraphes. Au cours de l'étude expérimentale, nous étudierons notamment la nature

des acteurs qui constituent une traverse minimale multi-membres et leurs rôles au sein du réseau.

3.2 Problématique

C'est en s'appuyant sur des représentations et des concepts issus de la théorie des graphes que les réseaux sociaux ont été étudiés en sciences sociales dès les années soixante [Mor34] [CH77]. Parmi les questions essentielles que l'analyse de réseau s'efforce de traiter figure l'identification d'individus occupant un rôle déterminant dans le réseau. Ainsi, plusieurs indicateurs tels que la *centralité* ou le *prestige* ont été définis pour caractériser la position occupée par un acteur [Fre79] [Sco00] [WF94] [BE92]. Néanmoins, la communauté scientifique informatique a été confrontée au problème du passage à l'échelle des algorithmes classiques de détection de tels individus. Avec l'émergence du Web 2.0 et l'explosion des réseaux sociaux sur Internet, des travaux plus récents se sont attachés à repérer des acteurs qui occupent une place particulière dans le réseau et qui selon appelés, selon les auteurs, *influenceurs*, *médiateurs*, *ambassadeurs* ou encore les *experts* [Dom05], [STE07a], [STE07b], [ALTY08], [OH10].

L'identification de tels acteurs a eu, en effet, de nombreuses applications, e.g., dans les domaines de l'épidémiologie, du marketing, ou encore de la diffusion d'innovation.

En particulier, plusieurs algorithmes ont été présentés récemment [LKG+07], [CWY09], [WCSX10], [CWW10], [CYZ10], [KS06], [GLL11] pour résoudre le problème de recherche d'influenceurs, redéfini comme un problème de maximisation [DR01], [RD02], [KKT03].

Parmi les modèles de diffusion dans un réseau, répertoriés dans la littérature, on peut distinguer d'une part les modèles linéaires à seuil inspirés des travaux de Granovetter et Schelling [Gra78], [Sch78] et d'autre part les modèles à cascade in-

dépendantes [JG01]. Dans tous ces modèles, on considère qu'à un instant donné, chaque membre du réseau est soit actif soit inactif. On cherche, par un processus itératif, à déterminer les acteurs devenus actifs à partir d'un sous-ensemble d'acteurs initialement actifs. On suppose bien sûr qu'un acteur peut être influencé par ses voisins suivant un certain seuil ou une certaine probabilité. La mise en oeuvre de ces modèles requiert donc l'estimation de ces probabilités d'influence ou de ces seuils. Cependant, l'estimation des paramètres n'est pas le seul inconvénient de ces modèles.

En effet, la recherche des influenceurs dans un réseau peut être énoncée plus formellement comme un problème d'optimisation discrète, connu dans la littérature sous le nom de "*influence maximization*" ou "*spread maximisation*". Étant donné A, un ensemble d'acteurs du réseau et une mesure d'influence associée à cet ensemble, définie comme le nombre des acteurs devenus actifs à partir de A, le problème revient à déterminer, pour un paramètre k donné, les k acteurs du réseau qui maximise la fonction d'influence. Or, Kempe *et al.* ont démontré que ce problème était NP-complet pour les deux familles de modèles citées précédemment [KKT03].

En s'appuyant sur la théorie des fonctions submodulaires, Kempe a aussi défini un cadre d'analyse généralisant les modèles à cascade et les modèles à seuil, et a montré qu'il est possible de déterminer une solution qui approche la solution optimale à un facteur près, à l'aide d'algorithmes gloutons, tels que *Greedy Algorithm* [KKT03]. Ceci a conduit au développement d'heuristiques permettant de déterminer approximativement les influenceurs dans un réseau. Ainsi, en suivant ce cadre d'analyse, Leskovec *et al.* ont développé l'algorithme "Cost-Effective Lazy Forward" (CELF), qui a donné lieu ensuite à plusieurs extensions, telles que NewGreedy et MixedGreedy introduit par Chen *et al.* ou, plus récemment, CELF++ par Goyal *et al.* [LKG+07], [CWY09], [GLL11].

L'algorithme *Greedy* a fait aussi l'objet d'autres améliorations, exploitant des pro-

priétés spécifiques du modèle à cascade [KS06] [CWW10] ou du modèle à seuil [CYZ10]. D'autres solutions ont aussi été proposées pour résoudre le problème de maximisation de l'influence, comme par exemple le modèle de vote de Even-Dar *et al.*, qui exploite d'ailleurs les mêmes hypothèses que les modèles à seuil [EDS07]. Cependant, tous les travaux cités précédemment considèrent la recherche d'influenceurs comme un problème d'optimisation sans tenir compte explicitement des communautés présentes dans le réseau. Or, comme le souligne Scripps *et. al.*, il peut être utile, pour identifier des influenceurs, de mieux connaître les positions occupées par les acteurs au sein des communautés présentes dans le réseau [STE07b], [STE07a]. C'est d'ailleurs le principe de l'algorithme *Community-based Greedy*, qui consiste justement à détecter des communautés en tenant compte du processus de diffusion au sein du réseau puis à identifier les influenceurs au sein des communautés [WCSX10]. Cependant, comme les algorithmes précédemment cités, *Community-based Greedy* suppose que le réseau est décrit par un graphe simple de sorte que les relations entre les acteurs pris deux à deux sont connues. Ainsi, ces algorithmes exploitent la matrice d'adjacence associée au graphe décrivant le réseau. Dans de nombreuses applications, on ne dispose pas forcément de cette information. Par contre, on sait à quelle(s) communauté(s) appartient un acteur. Ainsi par exemple, on sait quels sont les chercheurs qui ont participé à la conférence *KDD* et ceux qui ont assisté à *VLDB* sans forcément connaître les liens directs existants entre ces chercheurs. De même, dans le domaine du marketing, on peut savoir quels sont les clients qui ont acheté des articles d'une gamme de produits sans savoir s'ils sont en relation. Dans ces deux cas, il peut être intéressant d'identifier des acteurs, en nombre le plus petit possible, susceptibles de diffuser des idées ou recommandations d'un groupe à un autre. C'est ce problème que nous nous proposons de résoudre. Nous émettons l'hypothèse que la propagation repose sur des acteurs qui sont susceptibles d'assurer la transmission entre les groupes d'individus. En ce sens, les multi-membres que

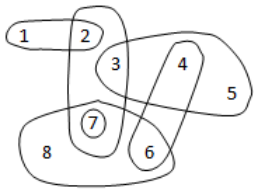
nous recherchons sont, pour partie, des ambassadeurs tels que Scripss *et al* les définissent [STE07b]. Il s'agirait donc de déterminer le plus petit ensemble de membres du réseau susceptibles de couvrir toutes les communautés.

L'objectif est donc de déterminer le plus petit ensemble d'acteurs, appelés *multi-membres*, qui sont susceptibles de représenter au mieux possible les différentes communautés du réseau en analysant le réseau dans ce contexte d'information incomplète où nous ne disposons pas de la matrice d'adjacence associée au graphe représentant le réseau mais où, en revanche, les communautés sont données.

Pour ce faire, nous proposons de représenter le système communautaire sous la forme d'un hypergraphe dans lequel les sommets représentent les acteurs et les hyperarêtes représentent les communautés. Dans cet hypergraphe, les multi-membres pourront être déterminés à partir des traverses minimales de l'hypergraphe, elles-mêmes définies comme un ensemble de sommets, minimal au sens de l'inclusion, qui intersecte toutes les hyperarêtes [Ber89].

3.3 Definition d'une traverse minimale multi-membres

Un réseau social peut être défini comme un ensemble d'entités interconnectés les unes aux autres [WF94]. Ces entités sont généralement des individus ou des organisations. Les relations matérialisent les interactions entre les entités.
Dans le contexte d'un hypergraphe où nous disposons seulement des communautés d'un réseau modélisées par les hyperarêtes, nous considérons que les multi-membres correspondent au plus petit ensemble de sommets tel qu'au moins un élément appartient à chaque communauté et, si possible, avec plusieurs éléments appartenant à des communautés de large taille. En ce sens, la définition des multi-membres peut se baser sur celle d'une traverse minimale définie comme étant un ensemble de sommets ayant une intersection, non vide, avec chaque hyperarête.



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

FIGURE 3.1 – Un exemple d'hypergraphe $H = (\mathcal{X}, \xi)$ et la matrice d'incidence IM_H correspondante

Exemple 5 *Dans la suite de ce chapitre, nous utiliserons à titre illustratif l'hypergraphe* $H = (\mathcal{X}, \xi)$ *de la Figure 3.1 (gauche) tel que* $\mathcal{X} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ *et* $\xi = \{\{1, 2\}, \{2, 3, 7\}, \{3, 4, 5\}, \{4, 6\}, \{6, 7, 8\}, \{7\}\}$. *L'ensemble de ses traverses minimales* $\mathcal{M}_H$ *est* $\{\{1, 4, 7\}, \{2, 4, 7\}, \{1, 3, 6, 7\}, \{1, 5, 6, 7\}, \{2, 3, 6, 7\}, \{2, 5, 6, 7\}\}$. *La Table de la figure 3.1 (droite) représente la matrice d'incidence associée à l'hypergraphe H. Cet exemple d'hypergraphe H sera repris tout au long de ce chapitre pour illustrer notre approche et extraire les multi-membres de H.*

Nous pouvons redéfinir une traverse minimale à partir de la notion d'ensemble de sommets essentiels.

Définition 11 SUPPORT D'UN ENSEMBLE DE SOMMETS
Soit l'hypergraphe $H = (\mathcal{X}, \xi)$ *et* X *un sous-ensemble de sommets de* $\mathcal{X}$. *Nous définissons* $Supp(X)$ *comme le nombre d'hyperarêtes de H, renfermant au moins un élément de* X : $Supp(X) = |\{e \in \xi | \exists x \in X \wedge \mathcal{R}(e, x) = 1\}|$, $\mathcal{R} \subseteq \xi \times \mathcal{X}$ *étant la relation binaire entre les hyperarêtes et les sommets de la matrice d'incidence correspondante à H.*

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 12 Ensemble essentiel de sommets *([CCL05]) 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 d'ordre des itemsets essentiels, *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,* $X \bigcap e_i \neq \emptyset \quad \forall e_i \in \xi, i = 1, \ldots, m$. *Donc, d'après la définition 7, 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.*

Exemple 6 *L'ensemble des traverses minimales* $\mathcal{M}_H$, *calculées à partir de l'hypergraphe de l'Exemple 5, est* $\{\{1,4,7\}, \{2,4,7\}, \{1,3,6,7\}, \{1,5,6,7\}, \{2,3,6,7\}, \{2,5,6,7\}\}$.

En se basant sur les définitions présentées ci-dessus, nous pouvons donner une définition plus formelle des traverses minimales multi-membres (TMM).

Définition 13 TRAVERSE MINIMALE MULTI-MEMBRES

Soit $H = (\mathcal{X}, \xi)$, un hypergraphe et $X \subset \mathcal{X}$. X est appelé Traverse minimale multi-membre, noté TMM*, si X vérifie les trois conditions suivantes :*

1. *(**Condition nécessaire**) : X est une traverse minimale : $X \in \mathcal{M}_H$.*
2. *(**Condition de composition**) : X est minimale dans $\mathcal{M}_H$ dans le sens de la cardinalité : $|X| = \tau(H)$ where $\tau(H) = Min\ \{|T|, \forall T \in \mathcal{M}_H\}$. $\tau(H)$ est le nombre de transversalité de H.*
3. *(**Condition de recouvrement maximum**) :*

$$\sum_{e_i \in \xi / e_i \bigcap X \neq \emptyset} |e_i| = Max\ \{ \sum_{e_i \in xi / e_i \bigcap T \neq \emptyset} |e_i|,\ \forall\ T \in \mathcal{M}_H\ tel\ que\ |T| = \tau(H)\ \}.$$

Ainsi, un ensemble de sommets est une TMM s'il constitue une traverse minimale, si sa taille est la plus petite possible et s'il maximise la condition de recouvrement. Spécifiquement, la première condition assure qu'il existe au moins un multi-membre dans chaque communauté. La seconde suppose que l'ensemble des multi-membres est le plus petit possible. Ainsi, l'objectif est de représenter toutes les communautés avec un nombre minimal de sommets. La troisième condition, calculant le recouvrement maximum, prend en compte le fait que certains multi-membres peuvent appartenir à une ou plusieurs mêmes communautés. Dans ce cas, le but est de favoriser les éléments qui appartiennent aux communautés les plus grandes.

Exemple 7 *Pour l'hypergraphe de l'Exemple 5, nous avons une seule* TMM*. C'est la traverse minimale $\{2, 4, 7\}$.*

3.4 Méthodologie et algorithmes d'extraction des multi-membres

A présent, nous introduisons un premier algorithme d'extraction des multi-membres, baptisé M2D, qui balaye l'espace de recherche en largeur. M2D repose sur la pro-

priété d'ordre idéal garantie par les ensembles de sommets essentiels pour l'élagage des candidats. En ce sens, cet algorithme agit d'une manière brute-force en générant les candidats nécessaires. Il s'arrête après avoir atteint le niveau k, i.e. le *nombre de transversalité*, où a été détecté la première traverse minimale.

3.4.1 Algorithme M2D

L'algorithme M2D, dont le pseudo-code est décrit par l'algorithme 9, prend en entrée un hypergraphe H et fournit en sortie l'ensemble des TMMs. L'algorithme effectue un parcours en largeur d'abord, i.e., il opère par niveau pour déterminer les ensembles de sommets essentiels. A chaque niveau k, un appel à la procédure APRIORI-GEN [AR94] est effectué pour calculer les candidats de taille k, à partir des ensembles de sommets essentiels de taille $k-1$. En effet, APRIORI-GEN génère un nouveau candidat $X'' = \{x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}\}$ à partir de deux candidats X' et X, tels que $X' = \{x_1, x_2, \ldots, x_{i-1}, x_i\}$ et $X = \{x_1, x_2, \ldots, x_{i-1}, x_{i+1}\}$. M2D calcule, ensuite, le support des k-candidats générés, à la ligne 9, et vérifie si leurs supports respectifs sont strictement supérieurs à ceux de leurs sous-ensembles directs (ligne 10).

Si parmi les ensembles de sommets essentiels, générés à un niveau k, il existe au moins un sommet, dont le support est égal au nombre d'hyperarêtes de l'hypergraphe (ligne 12), la boucle de la ligne 8 s'arrête et $\mathcal{TMM}$ renferme alors l'ensemble des traverses minimales qui sont minimales au sens de la cardinalité.

Ces ensembles de sommets vérifient aussi bien la condition nécessaire que la condition de composition de la définition 13. Au final, parmi ces candidats, les TMMs sont déterminés en se basant sur la fonction de calcul du recouvrement (ligne 16). Cette fonction calcule le nombre de sommets couverts par chaque candidat (*i.e.*, la somme des cardinalités des communautés auxquelles appartient ce candidat) et retourne ceux qui ont la valeur maximale. Ainsi, la troisième condition de la définition 13 est

Algorithme 9: M2D

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe

Sorties : $\mathcal{TMM}$

```
1  début
2      TMM = ∅ ;
3      i := 1;
4      pour chaque x ∈ X faire
5          si Supp(x) = |ξ| alors
6              TMM = TMM ∪ {x};
7      si TMM ≠ ∅ alors
8          Aller ligne 19
9      sinon
10         find = false ;
11         tant que L_i ≠ ∅ ou find = false faire
12             C_{i+1} := Apriori-Gen(L_i);
13             L_{i+1} := {X ∈ C_{i+1} | ∄ x ∈ X : Supp(X) = Supp(X\x)};
14             pour chaque X ∈ L_{i+1} faire
15                 si Supp(X) = |ξ| alors
16                     TMM = TMM ∪ {X};
17                     find = true ;
18             i := i + 1;
19     TMM = Recouvrement(TMM) ;
20     retourner TMM
```

aussi vérifiée.

Exemple 8 *Illustrons le déroulement de l'algorithme* M2D *sur l'hypergraphe de la*

Min. Tran.	Recouvrement
1 4 7	8
2 4 7★	10
1 3 6 7	
1 5 6 7	
2 3 6 7	
2 5 6 7	

Table 3.1 – Les Tmms extraits à partir de l'hypergraphe de la Figure 3.1

Figure 3.1. M2D *balaye l'espace de recherche en opérant en largeur jusqu'à arriver au niveau 3. Toutes les traverses minimales, que renferme l'hypergraphe de la figure 3.1 (page 44), sont données par la première colonne du tableau 3.1. Seules les plus petites traverses minimales, au sens de la cardinalité, nous intéressent et c'est la raison pour laquelle l'algorithme s'arrête au troisième niveau. Ces traverses minimales sont marquées comme étant des* Tmms *candidates :* {1 4 7} *et* {2 4 7} *dans le tableau 3.1. La fonction* Recouvrement *calcule, pour chaque candidat, la somme des tailles des communautés auxquelles il appartient et ne gardera que celui qui la maximise. Ainsi, le premier* Tmm *candidat est* {1, 4, 7}. *Le sommet* 1 *couvre le sommet* 2, *le sommet* 4 *couvre les sommets* 3, 5 *et* 6. *Enfin, le sommet* 7 *couvre les sommets* 2, 3, 6 *et* 8. *Le candidat* {1, 4, 7} *couvre donc, deux fois, chacun des sommets* 2, 3 *et* 6, *et une seule fois les sommets* 5 *et* 8. *Ainsi, le recouvrement du candidat* Tmm {1, 4, 7} *est égal à 8 sommets alors que, dans le même temps, le candidat* {2, 4, 7} *couvre au total 10 sommets. Ce dernier candidat est alors la seule* Tmm *de l'hypergraphe d'entrée et est retourné par l'algorithme* M2D.

L'algorithme M2D balaye l'espace de recherche du niveau 1 jusqu'au niveau k, i.e., le niveau renfermant les plus petites traverses minimales. Sachant que les Tmm appartiennent à un et un seul niveau, l'ensemble des candidats générés du niveau 1

jusqu'au niveau $k-1$ est inutile puisque ces derniers ne vérifient pas la condition nécessaire de la définition 13. Cette génération inutile des candidats handicape sérieusement l'efficacité du processus de recherche des TMMs, spécialement dans les bases éparses où la taille des TMMs est large (i.e., localisées dans un niveau élevé de l'espace de recherche). Idéalement, il serait plus bénéfique d'accéder directement à ce niveau k. Dans l'exemple ci-dessus, les candidats de taille 1 et 2 ne renferment pas des TMMs puisque la taille de la plus petite traverse minimale est égale à 3. Ainsi, "sauter" les niveaux 1 et 2 présenterait une optimisation conséquente dans la mesure où le nouveau algorithme n'aura pas à générer et tester les candidats inutiles. Cet algorithme, appelé O-M2D, est une optimisation de l'algorithme M2D et détermine intelligemment le niveau adéquat k pour identifier les TMM à partir des k-candidats uniquement.

3.4.2 Algorithme O-M2D

Comme M2D, l'algorithme O-M2D prend en entrée un hypergraphe H et donne en sortie l'ensemble des TMMs. Le balayage s'effectue sur un seul niveau, i.e., le niveau qui renferme les TMMs. O-M2D commence par invoquer la fonction GET-MINTRANSVERSALITY, dont le pseudo-code est décrit par l'Algorithme 11, pour localiser le niveau où la taille des candidats générés est égale à celle des TMMs (ligne 2). Ce niveau correspond au nombre de transversalité de l'hypergraphe d'entrée.

Ensuite, O-M2D génére, un à un, l'ensemble des k-candidats (ligne 3). Pour chaque k-candidat, i.e., ensemble de sommets de taille k, l'algorithme calcule son support (ligne 4). Si ce support est strictement supérieur au maximum des supports de ses sous-ensembles directs et qu'il est égal au nombre des hyperarêtes (ligne 5), alors ce candidat est marqué comme étant une traverse minimale et donc une TMM potentielle.

Quand toutes les traverses minimales ont été extraites, la fonction RECOUVRE-

MENT (ligne 7) se charge d'identifier l'ensemble des TMMs, comme expliqué plus haut.

Algorithme 10: O-M2D

Entrées : $H = (\mathcal{X}, \xi)$: Hypergraphe et IM_H sa matrice d'incidence correspondante

Sorties : $\mathcal{TMM}$

1 **début**
2 $Level :=$ GETMINTRANSVERSALITY(IM_H) ;
3 **pour chaque** $X \subseteq \mathcal{X}$ *tel que* $\mid X \mid = level$ **faire**
4 **si** $\nexists\, x \in X : Supp(X) = Supp(X \backslash x)$ **alors**
5 **si** $Supp(X) = |\xi|$ **alors**
6 $\mathcal{TMM} = \mathcal{TMM} \cup \{X\}$;
7 $\mathcal{TMM} =$ RECOUVREMENT$(\mathcal{TMM})$;
8 **retourner** $\mathcal{TMM}$

La fonction GETMINTRANSVERSALITY recherche le nombre minimal de sommets pouvant constituer une traverse minimale, i.e. le nombre de transversalité de l'hypergraphe. Pour ce faire, la fonction parcourt les sommets, un par un (ligne 3). Pour chaque élément x de $\mathcal{X}$, GETMINTRANSVERSALITY supprime de la matrice d'incidence IM_H les hyperarêtes de ξ qui contiennent x (ligne 5). Les hyperarêtes restantes sont stockés dans ξ'. La fonction invoque ensuite HYP_EMPTY, dont le pseudo-code est donné par l'Algorithme 12. HYP_EMPTY est une fonction récursive qui stocke dans T les sommets ayant le plus grand support dans ξ' (ligne 5) et les traitera, un par un, en supprimant à chaque fois les hyperarêtes auxquelles appartient le sommet traité (ligne 8). La condition d'arrêt de notre fonction récursive est l'absence d'hyperarêtes dans ξ' (ligne 2). La valeur stockée dans m correspond au nombre d'appels à la fonction HYP_EMPTY nécessaires pour que ξ' soit égal à

Algorithme 11: GetMinTransversality

Entrées : Matrice d'incidence IM_H associée à $H = (\mathcal{X}, \xi)$

Sorties : T : Une plus petite traverse minimale de H ; k : Nombre de transversalité de H

```
1  début
2      k = | ξ | ;
3      T = ∅ ;
4      pour chaque x ∈ 𝒳 faire
5          i = 1 ;
6          T_tmp = ∅ ;
7          T_tmp[i] = x ;
8          ξ' = ξ \ {e ∈ ξ | x ∈ e} ;
9          (n, T_tmp) = HYP_EMPTY(ξ', | ξ |, i, T_tmp) ;
10         si n < k alors
11             k = n ;
12             T = T_tmp ;
13     retourner (k, T)
```

l'ensemble vide. Pour chaque élément de T, la fonction vérifie si m est la valeur trouvée jusque-là, parmi les éléments traités de T. Si tel est le cas, elle est stockée dans *min* dont la valeur est retournée à la fin. Pour chaque sommet x traité, l'ensemble ξ' est réactualisé à toutes les hyperarêtes de l'hypergraphe auxquelles nous supprimons celles qui contiennent x. Au final, GetMinTransversality retourne le nombre minimum d'itérations permettant de "vider" la matrice d'incidence. La valeur de k, retournée par la fonction, correspond ainsi au nombre de transversalité de l'hypergraphe d'entrée H.

Conjecture 1 *La fonction* GetMinTransversality *permet d'obtenir une borne*

Algorithme 12: HYP_EMPTY

Entrées : ξ' : Ensemble d'hyperarêtes ; min, i : entier ; T_{tmp} : tableau de sommets

Sorties : min : Nombre minimum d'itérations pour obtenir un hypergraphe vide ; T' : Ensemble de sommets de cardinalité égale à min

```
1  début
2      si ξ' = ∅ alors
3          retourner (i, T_tmp)
4      sinon
5          T = {x ∈ X tel que | {e ∈ ξ' | x ∈ e} | = max
           {| {e ∈ ξ' | x_l ∈ e} |, x_l ∈ X}} ;
6          T' = ∅ ;
7          i = i + 1 ;
8          pour chaque x ∈ T faire
9              ξ'' = ξ' \ {e ∈ ξ' | x ∈ e} ;
10             T_tmp[i] = x ;
11             (m, T_tmp) = HYP_EMPTY(ξ'', min, i, T_tmp) ;
12             si m < min alors
13                 min = m ;
14                 T' = T_tmp ;
15         retourner (min, T')
```

maximale du nombre de transversalité, noté $\tau(H)$ dans la Definition 13, d'un hypeergraphe. Dans le meilleur des cas, cette borne est exactement le nombre de transversalité.

Exemple 9 *Reconsidérons l'exemple illustratif de la Figure 3.1. En optimisant,*

comme nous l'avons expliqué précédemment, l'algorithme M2D*, les sommets et 2-candidats ne sont pas générés et, donc, leurs supports ne sont pas calculés.* O-M2D *accède directement au niveau 3, générant tous les 3-candidats. Parmi ces 3-candidats,* O-M2D *détecte 24 ensembles de sommets essentiels mais seulement deux d'entre eux sont des traverses minimales :* $\{1, 2, 7\}$ *et* $\{2, 4, 7\}$*. En d'autres termes, seuls ces deux ensembles de sommets ont un support égal au nombre d'hyperarêtes. La fonction* RECOUVREMENT *permet de déterminer, au final, la ou les* TMMs*. Le recouvrement du Tmm candidat {1, 4, 7} est égal à* 8 *sommets alors que celui du candidat* {2, 4, 7} *est égal à* 10 *sommet. Donc,* O-M2D *sélectionne* {2, 4, 7} *comme unique* TMM*, en sortie.*

3.5 Etude de la complexité

A partir d'un hypergraphe de n sommets et m hyperarêtes, nous avons :

1. La fonction GETMINTRANSVERSALITY a une complexité exponentielle, au pire des cas, de $O(m * n^{m+1})$, avec $n = \mathcal{X}$ et $m = \mid \xi \mid$, pour déterminer le nombre de transversalité.
2. La cardinalité de l'ensemble des candidats de taille k générés est égale à C_n^k.
   (a) O-M2D calcule, ensuite, le support de chaque sous-ensemble direct. Le support est obtenu en m opérations.
   (b) Le support de X est calculé en m opérations.
3. Les tests pour vérifier que X est une traverse minimale s'effectuent en $O(1)$.

Pour un hypergraphe donné H, l'algorithme O-M2D calcule donc l'ensemble des multi-membres en : $O(m * n^{m+1} \times m \times C_n^k) \equiv O(m^2 * n^{m+1} * C_n^k)$.

3.6 Etude expérimentale

Au cours de notre étude expérimentale, nous mettons l'accent sur une évaluation approfondie des performances des algorithmes présentés dans la section précédente. Nous comparons à travers les nombreuses expérimentations menées, les performances de O-M2D *vs* respectivement M2D, MtMiner [HBC07] et ks [KS05]. De tous les algorithmes d'extraction des traverses minimales existants dans la littérature, notre choix s'est porté sur ces deux derniers en raison des disponibilités de leurs codes sources [1]. Nous avons ainsi eu la possibilité de les modifier pour qu'ils ne calculent que les plus petites traverses minimales. Par ailleurs, tous les algorithmes considérés sont implémentés en *C++* (compilés avec *GCC* 4.1.2) et les expérimentations réalisées sur une machine munie d'un processeur Intel Core *i*7 ayant une fréquence d'horloge de 2GHz et 6 Go de mémoire centrale, et avec le système d'exploitation de Linux, Ubuntu 10.04.

Durant ces expérimentations, nous avons considéré un jeu de données lié à une application de gestion de projet, des jeux de données "pire des cas" ainsi qu'un autre ensemble de jeux que nous avons construit à partir de deux bases de données du monde réel. Tout au long de notre étude expérimentale, nous avons vérifié que la borne maximale retournée par la fonction GetMinTransversality est bien égale au nombre de transversalité pour chaque hypergraphe traité.

Jeu de données de gestion de projets

En gestion de projet, nous pouvons connaître les compétences requises pour mener à bien un projet donné, ainsi que celles des acteurs. L'objectif est alors d'identifier le plus petit ensemble d'individus capables de réaliser le projet. De plus, on peut souhaiter que chaque acteur puisse avoir le plus de compétences possibles. Ceci revient alors à chercher les tmms.

1. Nous remercions les auteurs d'avoir mis à notre disposition leurs codes sources.

Dans ce cas, une communauté serait composé d'un ensemble d'acteurs offrant une même compétence. Les communautés ne sont pas disjointes puisqu'un acteur peut avoir plusieurs compétences. Le jeu de données correspondant à ce problème est représenté par un hypergraphe constitué de 168 sommets, dont chacun correspond à un acteur, et 50 hyperarêtes, dont chacune correspond à une communauté, c'est à dire à une compétence nécessaire pour le projet. Les caractéristiques de cet hypergraphe,

	$\|Comm.\|$	$\|\mathcal{X}\|$	$\|\xi\|$	$\|\mathcal{M}_H\|$	#TMM	$\tau(H)$
PM	5	168	50	320	16	9

TABLE 3.2 – Caractéristiques du jeu de données de gestion de projets

appelé PM, sont résumés par le tableau 3.2 où $|Comm.|$ correspond à la taille de la plus petite hyperarête, $|\mathcal{X}|$ au nombre de sommets, $|\xi|$ au nombre d'hyperarêtes, $|\mathcal{M}_H|$ au nombre de traverses minimales, #TMM au nombre de multi-membres calculés et $\tau(H)$ au nombre de transversalité. L'objectif est donc de rechercher les plus petits ensembles d'acteurs ayant les compétences requises pour mener à bien le projet.

Performances et interpretations sur le jeu de données de gestion de projet : Comme le montre le tableau 3.3, MTMINER est incapable de traiter ce jeu de données alors que les algorithmes KS, M2D et O-M2D nécessitent, respectivement, 307, 15, 1688,22 et 158,93 secondes pour extraire les traverses minimales multi-membres.

Nos algorithmes ont extrait 320 traverses minimales d'une taille égale à 9 qui correspond à la valeur du nombre de transversalité, noté $\tau(H)$ dans la définition 13. Ceci signifie que nous devons réunir au moins neuf acteurs pour la réalisation du projet et ces 320 traverses minimales correspondent aux sous-ensembles d'acteurs ayant les compétences nécessaires pour le projet. Parmi ces 320 traverses minimales, seulement 16 sont considérées comme des traverses minimales multi-membres. Ces

dernières maximisent la condition de recouvrement. La particularité de ces traverses minimales multi-membres est qu'elles contiennent un ou plusieurs acteurs présentant diverses compétences. Ainsi, nos algorithmes parviennent à trouver des équipes, de taille minimale ayant le maximum de compétence, qui sont le plus aptes à conduire le projet.

	KS	MTMINER	M2D	O-M2D
PM	307,15	-	1688,22	**158,93**

TABLE 3.3 – Jeu de données de gestion de projets : temps d'exécution (en secondes)

Bases de communautés sociales

Dans cette seconde expérimentation, nous considérons des folksonomies, à partir desquelles nous avons extrait des communautés. Une *folksonomie* est un néologisme, né de la jonction des mots *folk* (*i.e.,* les gens) et *taxonomie*, désignant un système de classification collaborative par les internautes [Mik07]. L'idée est de permettre à des utilisateurs de partager et de décrire des objets via des tags librement choisis. Formellement, une *folksonomie* est composée de trois ensembles $\mathcal{U}$, $\mathcal{T}$, $\mathcal{R}$ et d'une relation ternaire Y entre eux, où $\mathcal{T}$ est un ensemble de tags (ou étiquettes) et $\mathcal{R}$ est un ensemble de ressources partagées par les utilisateurs, qui peuvent être des sites web à marquer [2], des vidéos personnelles à partager [3] ou des films à décrire [4] selon le type de la *folksonomie* considérée. Quant à l'ensemble $\mathcal{U}$, il consiste en l'ensemble d'utilisateurs d'une *folksonomie* qui sont décrits par leurs identifiants (pseudonymes).

Nous avons appliqué l'algorithme TRICONS [CJB12] pour l'extraction des tri-

2. *http ://del.icio.us*
3. *http ://youtube.com*
4. *http ://movielens.org*

concepts associés à de telles folksonomies. Ces derniers sont des ensembles maximaux de la forme (Utilisateurs, Ressources, Tags) : l'ensemble maximum d'utilisateurs, qui ont partagé un ensemble maximal de ressources qu'ils ont annoté avec un ensemble maximal de tags.

Pour extraire les communautés, nous projetons les tri-concepts sur la dimension "Utilisateurs". Ceci est réalisé en faisant varier le seuil du support minimal des utilisateurs, *i.e.,* $minsupp_u$, qui est le nombre minimal d'utilisateurs qu'un tri-concept peut contenir. Dans ce qui suit, nous décrivons les folksonomies considérées au cours de nos expérimentations.

1. DEL.ICIO.US [1] : le système DEL.ICIO.US est un service de marque-page social qui offre à ses utilisateurs la possibilité de partager leurs pages web préférées. La base de données considérée dans ce rapport contient tous les marque-pages ajoutés sur le site *http ://delicious.com* en Janvier 2007. Le processus de récupération regroupe quelque $494,636$ marque-pages qui ont été publiés par $54,915$ utilisateurs par le biais de $64,968$ tags sur $129,220$ ressources. Dans cette étude expérimentale, nous considérons qu'une communauté, dans une base DEL.ICIO.US, est constituée des utilisateurs ayant partagé, au moins, deux mêmes pages web. Avant l'application de nos algorithmes, un pré-traitement sur ces données a permis de dégager les communautés qui serviront d'hyper-arêtes dans l'hypergraphe d'entrée.
2. MOVIELENS [2] : il s'agit d'un système de recommandation filmographique MOVIELENS, dont le site web a été conçu par un groupe de recherche, *GroupLens*, à l'université de Minnesota, aux États-Unis. Disponible au public, ce jeu de données contient des évaluations explicites au sujet de films. Le site met à disposition deux jeux de données d'évaluations de films, de tailles différentes. Le premier jeu comprend $1,000,000$ évaluations, de 1 à 5 étoiles, faites par

1. www.delicious.com
2. www.movielens.umn.edu

environ 6,000 utilisateurs, et le second comprend 100,000 évaluations fournies par 943 utilisateurs sur 1,682 films, entre Septembre 1997 et Avril 1998. C'est ce second jeu de données que nous avons utilisé pour nos tests, en considérant qu'une communauté est formée des utilisateurs qui ont fourni leur avis sur au moins deux mêmes films.

En variant $minsupp_u$, nous obtenons quatre jeux de données à partir de la base de données DEL.ICIO.US des folksonomies (notés *Del1*, *Del2*, *Del3* et *Del4*) et trois jeux de données à partir de la base de données MOVIELENS des folksonomies (notés *Mov1*, *Mov2* et *Mov3*). Dans l'hypergraphe associé à chaque jeu de données, les sommets représentent les utilisateurs et les hyperarêtes correspondent aux communautés où une communauté représente un ensemble d'utilisateurs ayant partagé le même ensemble de ressources avec le même ensemble de tags. Pour chaque jeu de données, l'objectif est de trouver le plus petit ensemble d'utilisateurs permettant de représenter toutes les communautés.

Les caractéristiques des différents jeux de données sont résumés dans le Tableau 3.4. Ainsi, la première colonne $|Comm.|$ indique le nombre minimum de sommets dans une communauté (*i.e.* $minsupp_u$). La seconde colonne contient le nombre de sommets ($|\mathcal{X}|$) de l'hypergraphe et la troisième le nombre d'hyperarêtes ($|\xi|$). La quatrième colonne indique le nombre de traverses minimales ($|\mathcal{M}_H|$) que renferme l'hypergraphe. L'avant-dernière colonne montre le nombre de TMMs. La dernière colonne correspond à la taille des TMMs, en termes de nombre de sommets (τ(H)). Nous pouvons noter que plus la valeur de $|Comm.|$ est basse, plus le nombre de TMMs (#TMM) et leurs tailles ($\tau(H)$) sont élevées, atteignant 21 pour *Del4* et 20 pour *Mov3*.

Performances : comme le montre le tableau 3.5, les différents tests confirment que l'algorithme O-M2D surpasse largement les algorithmes M2D, MTMINER et KS, pour l'ensemble des jeux de données. Par ailleurs, si M2D présente des temps

	$\|Comm.\|$	$\|\mathcal{X}\|$	$\|\xi\|$	$\|\mathcal{M}_H\|$	#TMM	$\tau(H)$
Del1	6	51	38	13	4	5
Del2	5	119	91	52	10	6
Del3	3	165	157	1800	78	13
Del4	2	248	179	8976	201	21
Mov1	5	88	80	108	1	6
Mov2	3	143	246	172	3	12
Mov3	2	196	501	306	26	20

TABLE 3.4 – Caractéristiques des bases sociales [(**Haut**) DEL.ICIO.US (**Bas**) MOVIELENS]

d'exécution élevés, il est assez robuste pour venir à bout de tous les jeux de données, alors que la consommation mémoire élevée de MTMINER l'empêche de s'exécuter sur les jeux *Del3*, *Del4* et *Mov3*. En déterminant le nombre d'éléments dans une TMM, M2D est capable d'élaguer de nombreux candidats qui sont générés et traités par MTMINER. Dans les deux types de jeux de données, les résultats confirment, par ailleurs, que l'écart, en termes de temps d'exécution, des différents algorithmes est en faveur de O-M2D. Cet écart est plus conséquent quand les nombres de sommets et d'hyperarêtes sont grands. En effet, la dernière colonne du tableau 3.4 montre que la taille des TMMs est très large, atteignant respectivement, 13 pour *Del3*, 21 pour *Del4*, et 20 pour *Mov3*. L'avantage principal de O-M2D se résume dans sa faculté à cibler directement ce niveau (appelé *level* dans l'Algorithm 2 et τ(H) dans la *Définition* 13).

Par exemple, pour *Del3* et *Del4*, MTMINER ne peut pas extraire les traverses minimales quand l'hypergraphe renferme plus de 157 hyperarêtes. Par ailleurs, sachant que le nombre de traverses minimales croît exponentiellement, l'avantage que

présente l'algorithme O-M2D est sa capacité à fournir un résultat sans stocker les candidats en mémoire.

	KS	MTMINER	M2D	O-M2D
Del1	88,26	77,45	276,65	**61,28**
Del2	263,90	200,66	964,32	**112,50**
Del3	401,38	-	1920,12	**174,78**
Del4	793,08	-	2880,84	**364,92**
Mov1	72,00	53,28	335,71	**59,52**
Mov2	262,09	185,56	1492,34	**131,84**
Mov3	881,73	-	2655,63	**351,27**

TABLE 3.5 – Bases sociales [(**Haut**) DEL.ICIO.US (**Bas**) MOVIELENS] : Temps d'exécution (en secondes)

Dans le but d'analyser, en profondeur, les TMMs calculées par nos algorithmes, considérons le jeu de données *Del2* du Tableau 3.4 et les caractéristiques des sommets qui appartiennent à l'ensemble des TMMs.

O-M2D donne en sortie *10* TMMs de taille 6, i.e, composé de 6 sommets. Cela signifie que nous devons trouver au moins 6 utilisateurs pour représenter l'ensemble des communautés. Un examen de près de ces TMMs montre que *4* sommets (10, 47, 77, 78) appartiennent à tous les TMMs extraites à partir de ce jeu de données. Nous les appelons "*actifs*" car ils ont la plus importante activité de marquage (tagging) dans le jeu de données Del2 de DEL.ICIO.US, comme le montre la Figure 3.2 et la Figure 3.3.

Les deux autres sommets (que nous appelons "*stratégiques*") n'ont pas, au contraire, une activité de marquage exceptionnelle. Leur appartenance aux TMMs s'explique par le fait qu'ils représentent des communautés qui ne renferment aucun sommet

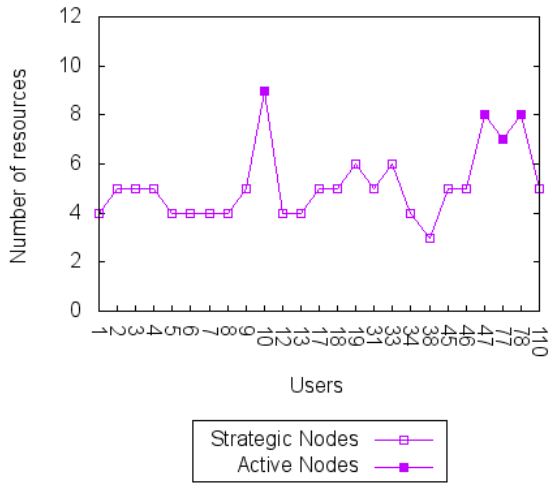


FIGURE 3.2 – Nombre de ressources partagées par les 25 utlisateurs les plus actifs

actif.

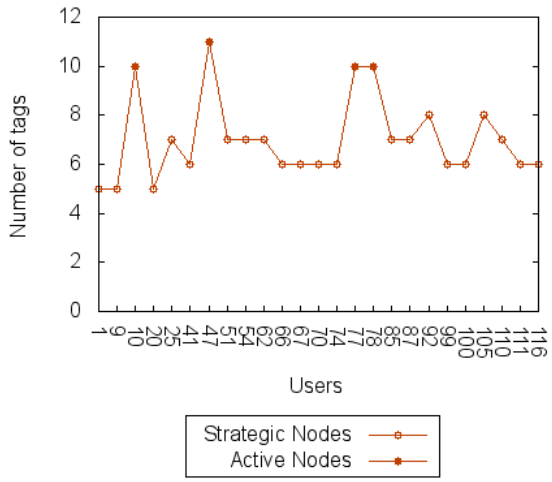


FIGURE 3.3 – Nombres de tags des 25 utilisateurs les plus actifs

Consommation mémoire : les statistiques fournies par le Tableau 3.6 mettent en évidence la consommation en RAM très faible des algorithmes O-M2D et KS. Lorsque MTMINER et M2D doivent générer et sauvegarder en mémoire tous les

candidats de tous les niveaux balayés, O-M2D cible directement le niveau adéquat et teste la condition d'essentialité des candidats générés. Un examen attentif du nombre de transversalité des hypergraphes *Del4* et *Mov3*, dans le Tableau 3.4, indique la quantité de candidats que MtMiner et M2D ont à traiter. En effet, ces derniers algorithmes doivent atteindre le 21^{me} niveau pour le jeu de données Del4 et le 20^{me} niveau pour Mov3. Ceci explique pourquoi MtMiner est dans l'incapacité d'atteindre le 20^{me} niveau.

	KS	Mtminer	M2D	O-M2D
Del1	4.335	4.334.102	1.807.980	**9.223**
Del2	6.290	5.568.931	2.606.722	**11.810**
Del3	9.603	-	3.723.119	**16.369**
Del4	13.844	-	4.458.656	**18.454**
Mov1	3.991	3.518.223	1.366.841	**8.604**
Mov2	6.387	4.976.213	2.461.857	**10.611**
Mov3	9.640	-	3.004.886	**13.602**

Table 3.6 – Bases sociales[(**Haut**) del.icio.us (**Bas**) MovieLens] : Consommation mémoire (en KO)

Bases "pires des cas"

Les bases "pire des cas" sont introduites pour étudier plus en profondeur les performances des algorithmes considérés au cours de cette étude expérimentale. Elles correspondent à une matrice d'incidence définie comme suit :

Définition 14 *Un contexte "pire des cas" $IM_H = (\xi, \mathcal{X}, \mathcal{R})$, où ξ et $\mathcal{X}$ sont, respectivement, les ensembles finis d'hyperarêtes et de sommets de l'hypergraphe H, est une matrice dans laquelle toutes les hyperarêtes sont formées du même nombre*

d'éléments, égal à n, et où chaque sommet a un support égal à 1 (Supp(x) = 1, $\forall x \in \mathcal{X}$).

Par exemple, un contexte "pire des cas" pour $|\xi| = 3$, $\mathcal{X} = 9$ et $n = 3$, est donné par le Tableau 3.7. Les bases "pire des cas" nous permettent d'évaluer le comportement des algorithmes dans des cas extrêmes. Le test consiste à varier les valeurs de n et $|\xi|$, jusqu'à ce que les algorithmes ne puissent plus s'exécuter correctement.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
e_1	×	×	×						
e_2				×	×	×			
e_3							×	×	×

TABLE 3.7 – Base pire des cas pour $|\xi| = 3$

Performances : le Tableau 3.8 montre pour chaque valeur de n, le nombre maximal d'hyperarêtes qui peuvent être traitées par les algorithmes considérés et les temps de traitement associés. Grâce à sa capacité à déterminer le nombre d'éléments d'une TMM, l'algorithme O-M2D présente un avantage indéniable par rapport à MTMINER et M2D. Selon les données du Tableau 3.8, pour une valeur donnée de n, O-M2D présente des temps d'exécution nettement meilleurs que les trois autres algorithmes et est capable de traiter des bases "pire des cas" pour des valeurs élevées de $|\xi|$. A titre d'exemple, pour $n = 4$, MTMINER s'arrête brusquement pour un nombre d'hyperarêtes égal à 12 alors que l'algorithme M2D résiste mieux et s'arrête à une valeur égale à 20. Dans le même temps, O-M2D et KS s'arrête pour une valeur de $|\xi|$ égal à 74.

Consommation mémoire : pour ces bases "pire des cas", nous avons étudié aussi la consommation mémoire. Ainsi, le Tableau 3.9 montre, respectivement, la mémoire consommée par les cinq algorithmes. O-M2D et KS se montrent alors très

performant par rapport aux algorithmes MtMiner et M2D. En effet, par rapport à la consommation mémoire des trois autres algorithmes, celle de O-M2D, tout comme celle de ks, est négligeable. O-M2D ne stocke en mémoire que l'hypergraphe d'entrée et les candidats, de taille égale au nombre de transversalité, générés sont traités sans être sauvegardés.

3.7 Conclusion

Au cours de ce chapitre, nous avons introduit une nouvelle approche pour la détection d'une classe particulière des traverses minimales, que nous avons appelées traverses minimales multi-membres, à partir d'un système communautaire représenté par un hypergraphe. L'une de nos contributions se trouve dans la définition des Tmms en se basant sur la notion d'ensemble de sommets essentiels. Ceci nous a permis de mettre en place un algorithme optimisé, qui cible directement le niveau qui renferme les Tmms. Des expérimentations effectuées sur différents jeux de données ont montré que l'algorithme O-M2D présente des performances très intéressantes par rapport à celles obtenues avec des algorithmes classiques. Cette contribution a été publiée dans une revue internationale [JLB14a] et deux conférences avec comité de lecture [JLB12b, JLB12a]. Dans le chapitre suivant, nous allons étendre notre approche pour extraire l'ensemble de toutes les traverses minimales, en proposant une représentation concise et exacte de cet ensemble grâce à la notion d'irrédondance. Ceci a été motivé par le nombre de traverses minimales, qui peut être exponentiel même pour des hypergraphes simples.

n	$\lvert\xi\rvert$	KS	MTMINER	M2D	O-M2D
4	11	174,06	180,23	257,97	87,54
4	12	215,73	-	295,43	111,21
4	19	498,68	-	1457,78	380,89
4	73	5527,90	-	-	3234,30
5	10	194,42	214,31	308,10	90,12
5	11	229,14	-	352,10	96,12
5	19	650,97	-	1866,42	398,78
5	60	5139,02	-	-	3094,17
6	9	211,30	245,12	344,64	97,38
6	10	228,74	-	400,58	101,09
6	16	616,51	-	2119,47	472,39
6	53	5349,28	-	-	3181,40
7	9	248,59	299,46	400,49	100,72
7	10	262,93	-	468,18	119,72
7	16	668,31	-	2288,46	495,08
7	44	5172,55	-	-	2802,89
8	8	257,30	326,02	466,21	103,69
8	9	284,36	-	517,36	138,28
8	14	700,94	-	2557,73	531,44
8	29	5311,80	-	-	2949,06
9	7	248,27	387,66	511,92	104,09
9	8	270,686	-	534,84	168,03
9	13	753,04	-	2780,79	567,56
9	22	4868,93	-	-	2354,98
10	7	264,91	428,88	683,36	110,34
10	8	289,53	-	711,49	133,28
10	11	841,71	-	3283,36	624,66
10	13	4390,22	-	-	1899,29

TABLE 3.8 – Bases pire des cas : Temps d'exécution (en secondes)

n	$\lvert\xi\rvert$	KS	MtMiner	M2D	O-M2D
4	11	4.285	2.682.886	1.398.441	6.108
4	12	4.300	-	1.844.364	6.221
4	19	4.617	-	4.995.732	6.908
4	73	20.093	-	-	32.805
5	10	3.866	2.811.429	1.470.266	5.294
5	11	4.077	-	1.719.498	5.565
5	19	4.902	-	5.338.173	6.403
5	60	18.620	-	-	26.164
6	9	3.996	3.066.422	1.712.089	5.482
6	10	4.098	-	1.999.695	5.607
6	16	4.781	-	5.514.683	6.962
6	53	15.092	-	-	22.197
7	9	4.094	3.185.089	2.085.366	5.165
7	10	4.168	-	2.473.281	5.537
7	16	4.830	-	5.800.962	6.308
7	44	10.982	-	-	17.389
8	8	3.841	3.541.797	2.226.625	4.821
8	9	4.117	-	2.826.625	4.996
8	14	4.497	-	5.741.793	5.570
8	29	8.896	-	-	11.411
9	7	3.602	3.922.008	2.677.026	4.757
9	8	3.997	-	3.168.442	4.886
9	13	4.406	-	5.694.223	5.101
9	22	7.911	-	-	9.734
10	7	3.802	4.433.787	3.019.860	4.955
10	8	4.106	-	3.840.117	5.093
10	11	4.212	-	5.899.004	5.202
10	13	4.537	-	-	5.817

TABLE 3.9 – Bases pire des cas : Consommation mémoire (en KO)

Chapitre 4

Nouvelle représentation concise et exacte des traverses minimales par élimination de la redondance

4.1 Introduction

Dans le but d'optimiser l'extraction des traverses minimales, la réduction de l'espace de recherche se présente comme une solution potentielle évidente tant le nombre de traverses minimales d'un hypergraphe est généralement très important. Dans ce chapitre, nous proposons une représentation concise et exacte de l'ensemble des traverses minimales. Pour ce faire, nous introduisons la notion de traverse minimale irrédondante, basée sur trois nouveaux concepts que nous définissons, à partir desquels nous pouvons retrouver l'ensemble global de toutes les traverses minimales. Cette nouvelle approche, basée sur la notion de l'irrédondance dans les hypergraphes, a donné lieu à un algorithme d'extraction des traverses minimales, appelé IMT-EXTRACTOR. De plus, et afin de mettre en avant l'intérêt de cette nouvelle représentation des traverses minimales, nous proposons de l'appliquer au problème

de l'inférence des dépendances fonctionnelles afin d'optimiser le calcul de la couverture minimale d'une relation donnée. Nous montrons que cette couverture minimale extraite à travers les traverses minimales irrédondantes est plus réduite que celle calculée par les approches existantes dans la littérature.

4.2 Motivations

A la lumière du Chapitre *2*, nous avons vu, dans la littérature, que plusieurs algorithmes ont été proposés pour l'extraction des traverses minimales d'un hypergraphe. A partir de cet état de l'art, nous mettons en exergue, dans cette section, les motivations qui nous ont conduit à la proposition d'une représentation concise des traverses minimales introduite dans le présent chapitre.

De tous les algorithmes qui se sont attachés à calculer les traverses minimales, aucun n'a fait, à ce jour, une complexité théorique polynomiale en la taille de l'entrée et de la sortie, comme l'a démontré Hagen [Hag08]. Ceci est dû essentiellement au fait que le nombre de traverses minimales peut être exponentiel par rapport à la taille de l'hypergraphe d'entrée.

Notre approche, inspirée des travaux de [HBN08], est donc de chercher un sous-ensemble représentant de manière concise et exacte l'ensemble des traverses minimales. Ce sous-ensemble, qu'on appellera ensemble irrédondant de traverses minimales, sera construit en considérant l'ensemble des hyperarêtes auxquelles appartient chaque sommet, appelé *extension* et en construisant un hypergraphe irrédondant généralisé limité à un sous-ensemble des sommets initiaux ayant des extensions différentes que Kavvadias et Stavropoulos définissent comme des noeuds généralisés. L'espace de recherche s'en trouve alors réduit puisque plusieurs candidats ne seront pas générés par l'algorithme. Cette intuition se base sur le fait que si deux sommets X et Y appartiennent aux mêmes hyperarêtes, (*i.e*, ils ont la même extension) et si X appartient à une traverse minimale T, alors en substituant X par Y dans T on

obtient une nouvelle traverse minimale. En outre, le risque de redondance est éliminé puisque Y n'est plus pris en compte dans l'exploration de l'espace de recherche.

Contrairement aux travaux de [KS05], notre traitement s'effectue en prétraitement, et non en cours du processus de recherche des traverses minimales. Ainsi, à partir d'un sous-ensemble restreint de traverses minimales, notre approche permet de générer l'ensemble de toutes les traverses minimales. En ce sens, plus le nombre de sommets élagués est élevé, plus compact sera l'hypergraphe irrédondant et plus l'approche sera avantageuse. Ceci implique que notre approche cible un certain type d'hypergraphes qui, d'une part, renferment un très grand nombre de traverses minimales et, d'autre part, ont un nombre de transversalité élevé. De plus, ces hypergraphes sont composés par des larges ensembles de sommets appartenant aux mêmes hyperarêtes.

Dans la littérature, Medina et Nourine ont proposé une approche dont l'objectif est de réduire la taille de l'hypergraphe d'entrée [GMNR05]. Les auteurs introduisent une nouvelle notion, appelé "les clones", dont le principe est le suivant : deux items (ou sommets) sont des clones si nous pouvons les permuter sans que ceci n'altère l'hypergraphe initial. Cette approche a pour principal objectif de réduire le nombre d'hyperarêtes de l'hypergraphe ($\mid \xi \mid$). En effet, si à partir d'un premier ensemble d'hyperarêtes, il est possible d'en déduire un second ensemble, l'ensemble ξ s'en trouve considérablement réduit puisque le second ensemble serait éliminé. Cependant, notre approche propose de réduire la taille des hyperarêtes et non d'éliminer un ensemble d'hyperarêtes de l'hypergraphe d'entrée et c'est ce qui nous distingue des travaux de Medina et Nourine. Les deux notions ne sont pas équivalentes, comme le montre l'Exemple 10.

Exemple 10 *Soit un hypergraphe* $H = (\{1, 2, 3, 4, 5, 6\}, \{e_1 = \{1, 3, 4\}, e_2 = \{1, 3, 5\}, e_3 = \{1, 4, 5\}\}, e_4 = \{3, 4, 5\}, e_5 = \{2, 3, 4\}, e_6 = \{2, 4, 5\}\})$. *H renferme* 2 *sommets "clones", qui sont le* 3 *et le* 5, *mais aucun ensemble de sommets*

généralisés dans la mesure où chaque sommet de H a une extension différente des autres sommets

4.3 Notion d'irrédondance dans les hypergraphes

L'approche que nous proposons pour calculer les traverses minimales d'un hypergraphe, consiste à réduire l'hypergraphe en le représentant de manière plus concise et sans perte d'information. Elle part de la constatation que deux ou plusieurs sommets qui ont la même extension, (*i.e*, appartiennent exactement aux mêmes hyperarêtes) tiennent, à tour de rôle, la même position dans une traverse minimale mais ne peuvent y appartenir en même temps.

Afin de remédier au problème du nombre extrêmement élevé des traverses minimales que peut contenir un hypergraphe, comme le montre l'exemple suivant, nous définissons la notion de l'irrdédondance dans les hypergraphes. L'ensemble des sommets est partitionné en des groupes de *sommets généralisés* possédant un et un seul *représentant*. Les sommets généralisés sont définis en se basant sur les hyperarêtes auxquelles appartient chaque sommet.

Exemple 11 *Soit $H = (\mathcal{X}, \xi)$ un hypergraphe tel que $\mathcal{X} = (x_1, x_2, \ldots, x_{2n})$ et $\xi = (\{x_1, x_2\}, \{x_3, x_4\}, \ldots, \{x_{2n-1}, x_{2n}\})$. H est de taille $2n$ mais renferme 2^n traverses minimales.*

À partir de ces sommets généralisés, nous proposons de construire un hypergraphe généralisé (irrédondant) sur lequel seront calculées les traverses minimales irrédondantes qui serviront, par la suite, à déduire l'ensemble de toutes les traverses minimales d'un hypergraphe donné. Pour ce faire, nous proposons d'introduire à travers la Définition 15, un nouveau concept, noté *Extension*, qui nous permet de calculer pour chaque sommet l'ensemble exact des hyperarêtes auxquelles il appartient.

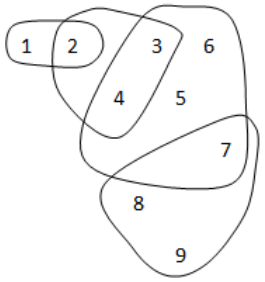


FIGURE 4.1 – Hypergraphe de 9 sommets et 4 hyperarêtes

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$e_1 = \{1, 2\}$	1	1	0	0	0	0	0	0	0
$e_2 = \{2, 3, 4\}$	0	1	1	1	0	0	0	0	0
$e_3 = \{3, 4, 5, 6, 7\}$	0	0	1	1	1	1	1	0	0
$e_4 = \{7, 8, 9\}$	0	0	0	0	0	0	1	1	1

TABLE 4.1 – Matrice d'incidence correspondante à l'hypergraphe de la Figure4.1

Définition 15 EXTENSION D'UN SOMMET

Soit IM_H une matrice d'incidence correspondante à l'hypergraphe $H = (\mathcal{X}, \xi)$ et $x \in \mathcal{X}$. $E = (e_1, e_2, \ldots, e_l) \subseteq \xi$ est une extension de x si $x \in e_i$, $\forall\ e_i \in E$. Nous noterons EXTENT*(x), l'extension de x dans IM_H. Le lien entre l'extension d'un sommet x et son support est donné par la formule : $Supp(x) = |Extent(x)|$.*

Exemple 12 *La Figure 4.1 illustre un hypergraphe $H = (\mathcal{X}, \xi)$ tel que $\mathcal{X} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ et $\xi = \{e_1, e_2, e_3, e_4\}$ avec $e_1 = \{1, 2\}$, $e_2 = \{2, 3, 4\}$, $e_3 = \{3, 4, 5, 6, 7\}$ et $e_4 = \{7, 8, 9\}$. Ainsi, la matrice d'incidence IM_H correspondante à l'hypergraphe H est décrite par le tableau 4.1. Dans cet exemple, l'extension du sommet 2 est égale aux hyperarêtes e_1 et e_2. Son support est, par conséquent, égal à 2.*

4.4 Traverses Minimales irrédondantes : approche et algorithme

Notre approche s'attache à réduire la taille de l'hypergraphe d'entrée dans le but d'obtenir une représentation concise et exacte, sans perte d'information, des traverses minimales. Cette dernière est basée sur deux notions importantes présentées par Kavvadias *et al.* dans leurs travaux [KS05] : les sommets généralisés et l'hypergraphe généralisé.

Définition 16 SOMMETS GÉNÉRALISÉS *[KS05]*
Soit $H = (\mathcal{X}, \xi)$ un hypergraphe. L'ensemble $X \subseteq \mathcal{X}$ est un ensemble de sommets généralisés de H si tous les sommets de X appartiennent aux mêmes hyperarêtes de ξ.

En d'autres termes et selon la Définition 15, X est considéré comme un ensemble de sommets généralisés où tous ses sommets ont la même extension. Ainsi, deux sommets x_i et x_j appartiennent à deux sommets généralisés différents si EXTENT(x_i) $\neq$ EXTENT(x_j).

Les ensembles de sommets généralisés, calculés à partir d'un hypergraphe H=($\mathcal{X}$, ξ) nous permettent de construire l'hypergraphe généralisé H' associé à H. H' est dit irrédondant puisque chaque sommet de H' a une extension différente. Chaque ensemble de sommets généralisés a un et un seul *représentant*, sélectionné selon la définition 17.

Définition 17 REPRÉSENTANT
Soit S un ensemble de sommets généralisés. x_i est un Représentant *de S si x_i est le premier élément de S, S étant trié par ordre lexicographique.*

Exemple 13 *Pour l'hypergraphe H de la Figure 4.1, constitué de 9 sommets, les extensions correspondantes à chacun des sommets sont indiquées dans le tableau 4.2.*

Ces extensions nous permettent de déduire les représentants des sommets généralisés et la liste de leurs autres membres, décrits par le tableau 4.3.

Sommet	Extension
1	$\{e_1\}$
2	$\{e_1, e_2\}$
3	$\{e_2, e_3\}$
4	$\{e_2, e_3\}$
5	$\{e_3\}$
6	$\{e_3\}$
7	$\{e_3, e_4\}$
8	$\{e_4\}$
9	$\{e_4\}$

TABLE 4.2 – Extensions des sommets de l'hypergraphe de la Figure 4.1

4.4.1 Cadre méthodologique

Notre représentation concise des traverses minimales repose sur deux notions importantes, expliquées dans la section 3 : les classes de substitution et surtout les *Représentants*. Les classes de substitution, calculées à partir de l'hypergraphe H=($\mathcal{X}$, ξ) permettent tout d'abord de construire l'hypergraphe irrédondant H' associé à H.

Définition 18 HYPERGRAPHE GÉNÉRALISÉ (IRRÉDONDANT) *Soit l'hypergraphe H=($\mathcal{X}$, ξ), $\mathcal{X}' \subseteq \mathcal{X}$ l'ensemble des représentants des différents sommets généralisés associés aux sommets de H et ξ' l'ensemble des hyperarêtes de H privées des éléments de $\mathcal{X}$-$\mathcal{X}'$ et défini par $\xi' = \{e_i \cap \mathcal{X}', e_i \cap \mathcal{X}' \neq \emptyset, \forall e_i \in \xi\}$. L'hypergraphe H' = ($\mathcal{X}'$, ξ') est appelé hypergraphe généralisé associé à H.*

Un hypergraphe généralisé H' est un hypergraphe irrédondant puisqu'il n'existe pas deux sommets de H' ayant la même extension.

Sommets généralisés	Représentants	Autres membres
$S_1 = \{1\}$	1	-
$S_2 = \{2\}$	2	-
$S_3 = \{3, 4\}$	3	4
$S_4 = \{5, 6\}$	5	6
$S_5 = \{7\}$	7	-
$S_6 = \{8, 9\}$	8	9

TABLE 4.3 – Sommets généralisés

Exemple 14 *A la lecture du tableau 4.3, nous aurons, pour l'hypergraphe de la figure 4.1,* 6 *ensembles de sommets généralisés, et les* 6 *représentants correspondant :* 1,2, 3, 5, 7, 8. *C'est uniquement à partir de ces derniers que sera construit l'hypergraphe irrédondant de la Figure 4.2 dont les hyperarêtes sont* $e'_1 = \{1, 2\}$, $e'_2 = \{2, 3\}$, $e'_3 = \{3, 5, 7\}$ *et* $e'_4 = \{7, 8\}$. *A partir de l'hypergraphe initial, nous remarquons que le sommet* 7, *par exemple, est le seul à avoir une extension* $\{3, 4\}$. 7 *sera donc le représentant d'une classe de substitution qui ne compte aucun autre élément. Par ailleurs, le sommet* 4 *a la même extension que le sommet* 3, i.e, $\{e_2, e_3\}$. *Cette dernière n'est pas la même pour aucun autre sommet. Par conséquent, une classe de substitution est créée, dont le représentant est* 3 *et qui renfermera les sommets* 3 *et* 4.

Définition 19 TRAVERSE MINIMALE IRRÉDONDANTE *Soit l'hypergraphe* $H=(\mathcal{X}, \xi)$ *et* $H' = (\mathcal{X}', \xi')$ *l'hypergraphe irrédondant associé à* H, *toute traverse minimale de* H' *constitue une traverse minimale irrédondante de* H.

L'ensemble des traverses minimales de H' (*i.e.*, des traverses irrédondantes de H), $\mathcal{M}_{H'}$, permet de construire l'ensemble des traverses minimales de H. En effet, toute

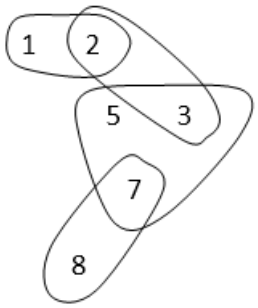


FIGURE 4.2 – Hypergraphe irrédondant correspondant à l'hypergraphe de la Figure 4.1

traverse minimale irrédondante $T = \{x_1, ...x_l\}$ de H, composée de l représentants x_i, $i = 1, .., l$ des classes de substitution $S_i, i = 1, .., l$ permet de générer $\prod_{i=1,l} |S_i|$ traverses minimales en remplaçant les représentants de chaque classe de substitution par les autres éléments de la classe.
Ainsi, dans l'exemple 13, la traverse minimale irrédondante $\{1, 3, 8\}$ formée des représentants des classes de substitution S_1, S_3 et S_6 de taille respective 1, 2 et 2 permet de construire 4 traverses minimales : $\{1, 3, 8\}$, $\{1, 4, 8\}$, $\{1, 3, 9\}$ et $\{1, 4, 9\}$.

4.4.2 Etude de la complétude

La construction de l'hypergraphe irrédondant $H' = (\mathcal{X}', \xi')$ à partir d'un hypergraphe initial $H = (\mathcal{X}, \xi)$ repose sur trois propriétés importantes, présentées ci-dessous. Ces propriétés et leurs preuves montrent que toute traverse minimale de H' est une traverse minimale de H et que toute traverse minimale de H peut être déduite à partir d'une traverse minimale de H'.

Proposition 2 *Toute hyperarête e_i, $i \in \{1, 2, \ldots, m\}$, de ξ est associée à exactement une hyperarête e'_i of ξ' définie par $e'_i = e_i \cap \mathcal{X}'$ et inversement toute hyperarête e'_i, $i \in \{1, 2, \ldots, m\}$, de ξ' correspond à une seule hyperarête e_i de ξ.*

Preuve 2 *Soit e_i une hyperarête de ξ et $e'_i \in \xi'$ tel que $e'_i = e_i \cap \mathcal{X}'$. Raisonnons par l'absurde et supposons que $e_i \cap \mathcal{X}' = \emptyset$, alors $\exists\ x \in e_i$ tel que $x \notin \mathcal{X}'$ puisque $e_i \neq \emptyset$ selon la Définition 1 d'un hypergraphe.*

Par ailleurs, la définition de H', où $x \notin \mathcal{X}'$, nous permet de déduire que $\exists\ y \in \mathcal{X}'$ qui représente un ensemble de sommets généralisés de x et, x et y ont la même extension.

Par conséquent, $y \in \mathcal{X}'$ et $y \in e_i$ contredisent le fait que $e_i \cap \mathcal{X}' = \emptyset$. Ainsi, chaque hyperarête $e_i \in \xi$, $i \in \{1, 2, \ldots, m\}$ correspond à exactement une hyperarête $e'_i \in \xi'$, $\in \{1, 2, \ldots, m\}$ définie par $e'_i = e_i \cap \mathcal{X}'$.

Inversement à présent, soit $e'_i \in \xi'$ tel que $e'_i = e_i \cap \mathcal{X}'$. Prouvons que e'_i correspond à exactement une seule hyperarête e_i de ξ. Supposons qu'il existe une autre hyperarête e_j de ξ, différente de e_i telle que $e'_i = e_j \cap \mathcal{X}'$. Donc, $\exists\ y \in e_j$ tel que $y \notin e_i$ puisque $e_i \neq e_j$ et EXTENT*$(y) \neq$* EXTENT*(x), $\forall x \in e_i$. Ainsi, y ou son représentant, y', appartiennent à $\mathcal{X}'$ et e_j, et ils n'appartiennent pas à e_i. Ceci contredit le fait que $\mathcal{X}' \cap e_i = \mathcal{X}' \cap e_j$. En conclusion, toute hyperarête e'_i de ξ' correspond à exactement une seule hyperarête e_i de ξ, $i \in \{1, 2, \ldots, m\}$.*

Nous présentons à présent diverses propriétés concernant la relation entre les traverses minimales irrédondantes et les traverses minimales.

Proposition 3 *Toute traverse minimale T de H' est une traverse minimale de H, i.e., $\forall\ T \subseteq \mathcal{X}'$, $T \in \mathcal{M}_{H'} \Rightarrow T \in \mathcal{M}_H$.*

Preuve 3 *$T \in \mathcal{M}_{H'} \Leftrightarrow T \cap e'_i \neq \emptyset$, $\forall e'_i \in \xi'$. Selon la Proposition 2, toute hyperarête e'_i de ξ' correspond à exactement une hyperarête e_i of ξ. Soit $e_i \in \xi$ tel que $e'_i = e_i \cap \mathcal{X}'$, $\forall\ i \in \{1, 2, \ldots, m\}$. Nous avons $T \cap e'_i \neq \emptyset\ \forall\ i \in \{1, 2, \ldots, m\} \Leftrightarrow T \cap e_i \cap \mathcal{X}' \neq \emptyset$. Puisque $T \subseteq \mathcal{X}'$, ceci implique que $T \cap e_i \neq \emptyset\ \forall\ i \in \{1, 2, \ldots, m\}$. Ce qui prouve que $T \in \mathcal{M}_H$.*

Proposition 4 *(*COMPLÉTUDE*) Toute traverse minimale T de H est associée à une traverse minimale T' de H'.*

Preuve 4 *Commençons par prouver que T' est une traverse de H'. Soit T une traverse minimale de H : $T = \{x_1, x_2, \ldots, x_k\} \in \mathcal{M}_H$ avec $T \cap e_i \neq \emptyset$, $\forall e_i \in \xi$. D'après les Définitions 16, 17 et 18, nous avons $\forall x \in T$, $\exists x' \in \mathcal{X}'$ tel que* EXTENT*(x) =* EXTENT*(x'). De plus, puisque $T \in \mathcal{M}_H$, nous avons $\forall x_i \in T$ et $\forall x_j \in T$, $i \neq j$,* EXTENT*$(x_i) \neq$* EXTENT*(x_j).*

Si nous notons T', le sous-ensemble de $\mathcal{X}'$ qui contient les représentants des éléments de T, nous avons $T' = \{x'_1, x'_2, \ldots x'_l\}$ et $T' \subset \mathcal{X}'$. Puisque les éléments de T et ceux de T' ont la même extension et T est une traverse de H, nous pouvons en déduire que T' est une traverse de H. Nous avons alors : $T' \cap e_i \neq \emptyset$, $\forall e_i \in \xi$. Or, comme T' est un sous-ensemble de $\mathcal{X}'$, nous pouvons affirmer que : $T' \cap \mathcal{X}' \cap e_i \neq \emptyset$, $\forall e_i \in \xi$
$\Leftrightarrow T' \cap e'_i \neq \emptyset$, $\forall e'_i \in \xi'$, suivant la Proposition 2. D'où, T' est une traverse de H'.

Prouvons, à présent, que T' est minimale dans H'. D'abord, nous pouvons noter que $T \in \mathcal{M}_H$ signifie que $\forall x \in T$, $\exists e_j \in \xi$ tel que $x \cap e_j \neq \emptyset$ et $T \setminus \{x\} \cap e_j = \emptyset$ puisque T est minimale, d'après l'hypothèse de départ. Ensuite, puisque les éléments de T et ceux de T' ont la même extension, nous avons $\forall x' \in T'$, $\exists e_j \in \xi$ tel que $x' \cap e_j \neq \emptyset$ et $T' \setminus \{x'\} \cap e_j = \emptyset$ (1). Cependant, selon la définition de H', $e'_j = e_j \cap \mathcal{X}'$ ce qui implique que $e'_j \subseteq e_j$ (2). À partir de (1) et (2), nous pouvons déduire que la traverse T' est minimale : $\forall x' \in T'$, $\exists e'_j \in \xi'$ tel que $T' \setminus \{x'\} \cap e'_j = \emptyset$.

Dans ce qui suit, nous présentons le cadre méthodologique, basé sur cette approche, qui a donné naissance à l'algorithme IMT-EXTRACTOR, pour l'extraction des traverses minimales irrédondantes.

4.4.3 Algorithme IMT-EXTRACTOR

L'algorithme IMT-EXTRACTOR, dont le pseudo-code, est décrit par l'Algorithme 13 prend en entrée une matrice d'incidence IM_H (correspondante à l'hypergraphe d'entrée) et fournit en sortie l'ensemble des traverses minimales, noté M_H.

L'algorithme commence par calculer les extensions de chaque sommet de $\mathcal{X}$ à partir desquelles seront construits les différentes sommets généralisés (ligne 3). Cette tâche est effectuée par la procédure SEARCH-SUBSTITUTION, dont le pseudo-code est décrit par l'Algorithme 14. Cette procédure fournit en sortie les différents sommets généralisés, conformément à la proposition 16, avec pour chacun la liste des sommets qui le compose et son *représentant*. A partir de ces données, une nouvelle matrice d'incidence $IM_{H'}$ associée à l'hypergraphe irrédondant H' est générée à l'aide de la procédure CHANGE-HYP selon la définition 18.

Algorithme 13: IMT-EXTRACTOR

Entrées : Une matrice d'incidence IM_H associée à $H = (\mathcal{X}, \xi)$

Sorties : $\mathcal{M}_H$, l'ensemble des traverses minimales de H

1 **début**
2 $M_{H'} = \phi$;
3 Gen_nodes := SEARCH-SUBSTITUTION(IM_H) ;
4 $IM_{H'}$ = CHANGE-HYP(IM_H, Gen_nodes) ;
5 $\mathcal{M}_{H'}$ = MMCS($IM_{H'}$) ;
6 $\mathcal{M}_H$ = GET-ALL-MT($\mathcal{M}_{H'}$, Gen_nodes) ;
7 **retourner** $\mathcal{M}_H$

IMT-EXTRACTOR invoque un algorithme efficace d'extraction des traverses minimales, par exemple MMCS [MU13] (line 5), présenté dans le chapitre 2. Ce dernier calcule l'ensemble des traverses minimales de H' selon une stratégie en profondeur d'abord et en utilisant une technique efficace pour tester la minimalité, basé sur la

notion d'"hyperarêtes critiques" [MU13]. MMCS retourne l'ensemble des traverses minimales de H' qui représentent l'ensemble des traverses minimales irrédondantes de H. A noter que nous pouvons employer un autre algorithme que MMCS pour la réalisation de cette tâche. Notre choix a été conditionné par les performances supérieures à ses concurrents de l'algorithme MMCS de [MU13].

La procédure SEARCH-SUBSTITUTION($\mathcal{K}$) calcule l'extension de chaque sommet x à partir de la matrice d'incidence (ligne 5), *i.e* l'ensemble des hyperarêtes auxquelles il appartient, à travers la fonction GET-EXTENT. Cette extension de x est ensuite comparée aux extensions des différents *représentants* des sommets généralisés existants (lignes 6-7). S'il existe déjà un représentant pour un ensemble de sommets généralisés S qui a la même extension alors x est ajouté à l'ensemble de sommets généralisés en question (ligne 8) et la variable booléenne $IsRep$ est mis à jour à *False*, sinon (*i.e*, $IsRep$ est égal à *True*), un nouveau ensemble de sommets généralisés est créé pour le sommet x qui en devient le représentant (ligne 11) et ce dernier est ajouté à la liste des *Représentants* (ligne 12). Une fois, tous les sommets traités, la procédure renvoie à l'algorithme IMT-EXTRACTOR le représentant et la liste des sommets composant chaque ensemble de sommets généralisés calculé (ligne 13).

Exemple 15 *Nous reprenons l'exemple 14 où on distingue* 6 *sommets généralisés, dans l'hypergraphe initial H de la Figure 4.1, représentés par les sommets* 1, 2, 3, 5, 7 *et* 8. *Ces sommets calculés par la procédure* SEARCH-SUBSTITUTION *permettent la construction de l'hypergraphe irrédondant H' illustré par la Figure 4.2. La matrice d'incidence correspondante à H' est ensuite utilisé par l'algorithme* MMCS *pour calculer toutes les traverses minimales $\mathcal{M}_{H'}$.* MMCS *donne ainsi en sortie l'ensemble* $\{\{2,7\}, \{1,3,7\}, \{1,3,8\}, \{2,3,8\}, \{2,5,8\}\}$. *Cet ensemble représente les traverses minimales irrédondantes de H. A partir de ces dernières, l'ensemble de toutes les traverses minimales de H est, par la suite, généré comme le décrit la section suivante.*

Algorithme 14: SEARCH-SUBSTITUTION

Entrées : Matrice d'incidence IM_H associée à H

Sorties : Gen_nodes : Ensemble de sommets généralisés

```
1  début
2      Rep = ∅ ; /* Ensemble des Représentants*/
3      pour chaque x ∈ 𝒳 faire
4          Extent(x) = GET-EXTENT(x, IM_H) ;
5          pour chaque y ∈ Rep faire
6              IsRep = true ;
7              si Extent(x) = Extent(y) alors
8                  ADD-TO-GENERALIZED-NODE(Gen_nodes, y, x) ;
9                  IsRep = false ;
10         si IsRep = true alors
11             ADD-GEN-NODE(Gen_nodes, x, Extent(x)) ;
12             Rep = Rep ∪ x ;
13     retourner (Gen_nodes) ;
```

4.4.4 Génération de toutes les traverses minimales

Le processus de substitution pour la reconstruction de toutes les traverses minimales à partir de leur ensemble irrédondant est réalisé par la fonction GET-ALL-MT, dont le pseudo-code est donné par l'Algorithme 15.

Algorithme 15: GET-ALL-MT

Entrées : $\mathcal{M}_{H'}$, Gen_nodes

Sorties : $\mathcal{M}_H$

```
1 début
2     M_H = ∅ ;
3     pour chaque T ∈ M_H' faire
4         M_H = M_H ∪ T ;
5         SUBSTITUTE(T, Gen_nodes, M_H) ;
6     retourner (M_H) ;
```

La fonction GET-ALL-MT prend en entrée un ensemble de traverses minimales irrédondantes de H, i.e., l'ensemble des traverses minimales de H'. Dans chacunes de ces traverses irrédondantes, tout représentant est substitué successivement par tous les sommets qui appartiennent au même ensemble de sommets généralisés, par le biais de la fonction SUBSTITUTE, presenté par l'Algorithme 16. Pour revenir à notre exemple 14, le tableau 4.4 indique les traverses minimales de H générées à partir de chaque traverse minimale irrédondante de H' tout en méttant en évidence les sommets qui appartiennent au même ensemble de sommets généralisés. Au final, les *15* traverses minimales de l'hypergraphe H sont calculées à partir de seulement *5* traverses minimales irrédondantes. Par exemple, les traverses minimales de M_H, $\{1,3,8\}$, $\{1,4,8\}$, $\{1,3,9\}$ et $\{1,4,9\}$, sont construites à partir de $\{1,3,8\}$ de $M_{H'}$ en remplaçant, respectivement, 3 par 4 et 8 par 9.

Proposition 5 *Toute traverse minimale $T \subseteq \mathcal{X}$ générée par la fonction* GET-ALL-

Algorithme 16: SUBSTITUTE

Entrées : T, Gen_nodes, $\mathcal{M}_H$

```
1 début
2     pour chaque x ∈ T faire
3         pour chaque G_n ⊂ Gen_nodes tel que x est un Représentant de G_n
          faire
4             si ∃ y ∈ G_n et x ≠ y alors
5                 T = (T \ x) ∪ y ;
6                 M_H = M_H ∪ T ;
7                 SUBSTITUTE(T, Gen_nodes, M_H) ;
```

TMs IRRÉDONDANTES	TMs CORRESPONDANTES
$\{2, 7\}$	$\{2, 7\}$
$\{1, \mathbf{3}, 7\}$	$\{1, \mathbf{3}, 7\}$, $\{1, \mathbf{4}, 7\}$
$\{1, \mathbf{3}, \underline{8}\}$	$\{1, \mathbf{3}, \underline{8}\}$, $\{1, \mathbf{4}, \underline{8}\}$, $\{1, \mathbf{3}, \underline{9}\}$, $\{1, \mathbf{4}, \underline{9}\}$
$\{2, \mathbf{3}, \underline{8}\}$	$\{2, \mathbf{3}, \underline{8}\}$, $\{2, \mathbf{4}, \underline{8}\}$, $\{2, \mathbf{3}, \underline{9}\}$, $\{2, \mathbf{4}, \underline{9}\}$
$\{2, \mathbf{5}, \underline{8}\}$	$\{2, \mathbf{5}, \underline{8}\}$, $\{2, \mathbf{5}, \underline{9}\}$, $\{2, \mathbf{6}, \underline{8}\}$, $\{2, \mathbf{6}, \underline{9}\}$

TABLE 4.4 – Déduction de l'ensemble des traverses minimales

MT *à partir de* T' *appartenant à* $\mathcal{M}_{H'}$*, est une traverse minimale de* H*.*

Preuve 5 *Soit* $T' = \{x'_1, x'_2, \ldots x'_k\} \in \mathcal{M}_{H'}$ *et* $T = \{x_1, x_2, \ldots x_k\}$ *générée par la fonction* GET-ALL-MT *à partir de* T'*. Commençons par prouver que* T *est une traverse de* H*.*

Par construction, $\forall x \in T$, $\exists\ x' \in T'$ *tel que* EXTENT(x) = EXTENT(x')*, ce qui implique que* x *et* x' *appartiennent exactement aux mêmes hyperarêtes de* H*.*

De plus, $T' \in \mathcal{M}_{H'} \Leftrightarrow T' \cap e'_i \neq \emptyset\ \forall e'_i \in \xi'$*. Sachant que* $e'_i = e_i \cap \mathcal{X}' \Rightarrow e'_i \subseteq$

e_i, nous avons alors $T' \cap e_i \neq \emptyset$ $\forall e_i \in \xi$ et puisque les éléments de T et T' ont la même extension, donc $T \cap e_i \neq \emptyset$, $\forall e_i \in \xi$, et T est ainsi une traverse de H.

Prouvons à présent que $T \in \mathcal{M}_H$. Nous supposons que la traverse T de H, générée à partir de T', n'est pas minimale dans H. Dans ce cas, il existerait un sous-ensemble $T_0 \subset T$ qui est une traverse minimale de H. Soit T_0 cette traverse minimale : $T_0 = \{x_1, x_2, \ldots x_{k-1}\} \subset T$ tel que $T_0 \cap e_i \neq \emptyset$, $\forall e_i \in \xi$.

$\forall x \in T_0$, $\exists x' \in T'$ tel que EXTENT*(x) =* EXTENT*(x'). Soit T'_0 le sous-ensemble de T' contenant les représentants des éléments de T_0. T'_0 vérifie les propriétés suivantes :*

(1) *$T'_0 \cap e_i \neq \emptyset$, $\forall e_i \in \xi$;*

(2) *$T'_0 \subset \mathcal{X}'$;*

(3) *$T'_0 \subset T'$ et $T_0 \neq T'$;*

A partir de (1) et (2), nous avons :
$T'_0 \cap e_i \cap \mathcal{X}' \neq \emptyset$, $\forall e_i \in \xi \Leftrightarrow T'_0 \cap e'_i \neq \emptyset$, $\forall$ $e'_i \in \xi'$.
Et par conséquent, $T'_0 \in \mathcal{M}_{H'}$.

D'après (3), T'_0, la traverse minimale de H', est incluse et non égale à T', ce qui constitue une contradiction avec le fait que nous avons supposé que T' est une traverse minimale de H'. Ainsi, T est minimale.

4.5 Etude expérimentale

Afin d'évaluer l'intérêt de notre approche, nous avons étudié, à travers une série d'expérimentations, les performances de notre algorithme IMT-EXTRACTOR. Dans cet objectif, nous avons introduit la notion de taux de compacité qui est défini comme étant la proportion de traverses minimales qui peuvent être déduites à partir de l'ensemble des traverses minimales irrédondantes, sans perte d'information. De plus, nous comparons les temps d'exécution de notre algorithme IMT-EXTRACTOR à ceux de KS [KS05] et MMCS [MU13]. Le temps global que requiert IMT-EXTRACTOR cor-

respond à la somme de celui du prétraitement, du post-traitement et de l'extraction des traverses minimales irrédondantes réalisée par MMCS à partir de l'hypergraphe irrédondant. Notre algorithme est implémenté en *C++* (compilé avec *GCC* 4.1.2) et nous avons utilisé une machine équipée d'un processeur Intel Core *i*7 CPU system, d'une capacité mémoire de 6 Go RAM et du système d'exploitation Linux, UBUNTU 10.04. L'algorithme MMCS utilisé au cours de ces expérimentations correspond à la version 3.1 disponible sur *Hypergraph Dualization Repository* [MU].

	p_l	p_u	$\|\mathcal{X}\|$	$\|\xi\|$	$\|Sommets\_Gen\|$	$\tau(H)$
Accidents1	0.88	0.88	81	990	31	1
Accidents2	0.90	0.90	336	10968	49	2
Connect-Win	0.05	0.11	79	12800	78	2

TABLE 4.5 – Caractéristiques des hypergraphes *Accidents* et *Connect*

Divers types d'hypergraphes ont été utilisés au cours de ces expérimentations. Le premier correspond à des hypergraphes générés à partir de la base de données "*Accidents*", disponible depuis le FIMI *repository*[5], et depuis la base de données "*Connect-4*" disponible à partir de l'UCI Machine Learning Repository[6]. Les caractéristiques de ces jeux de données sont résumés dans le tableau 4.5. La première et la seconde colonne, p_l et p_u, correspondent respectivement aux probabilités minimales et maximales pour qu'un sommet appartienne ou non à une hyperarête. En effet, un sommet appartient à une hyperarête avec une probabilité comprise entre p_l et p_u ($0 \leq p_l \leq p_u \leq 1$). Une probabilité d'appartenance d'un sommet x à une hyperarête e est donnée par le quotient $\mid e \mid / \mid \mathcal{X} \mid$. La cinquième colonne indique le nombre de sommets généralisés générés pour chaque hypergraphe.

La dernière colonne correspond au nombre de transversalité τ(H) de chaque hyper-

5. Frequent Itemset Mining Datasets Repository, *http ://fimi.cs.helsinki.fi/data/*
6. UCI Machine Learning Repository, *http ://archive.ics.uci.edu/ml*

graphe, le nombre de transversalité étant la taille de la plus petite traverse minimale, i.e., le nombre de sommets la composant.

	$\lvert\mathcal{M}_H\rvert$	$\lvert\mathcal{M}_{H'}\rvert$	θ
Accidents1	1961	1866	4,84%
Accidents2	17486	17199	1,64%
Connect-Win	4587967	4423837	3,57%

TABLE 4.6 – Statistiques sur les hypergraphes *Accidents* et *Connect*

La Table 4.6 montre les statistiques relatives à l'hypergraphe initial et à l'hypergraphe irrédondant pour chaque jeu. La seconde colonne indique le nombre de traverses minimales que renferme l'hypergraphe d'entrée, $|\mathcal{M}_H|$. La troisième correspond au nombre de traverses minimales irrédondantes, $|\mathcal{M}_{H'}|$ tandis que la dernière colonne représente le taux de compacité θ, calculé comme suit : $(|\mathcal{M}_H| - |\mathcal{M}_{H'}|)/|\mathcal{M}_H|$. Ce taux représente le pourcentage de traverses minimales qui peuvent être déduites à partir de l'ensemble des traverses minimales irrédondantes sans perte de l'information.

	Accidents1	Accidents2	Connect-Win
MMCS(H)	0,301	2,787	88,491
KS(H)	8,620	-	-
MMCS(H')	0,001	2,366	84,015
GET-ALL-MT($\mathcal{M}_{H'}$)	0,035	0,010	7,291
IMT-EXTRACTOR (H)	0,036	2,376	91,306

TABLE 4.7 – Temps de traitement sur les hypergraphes *Accidents* et *Connect* (en secondes)

En observant les statistiques récoltées sur les hypergraphes *Accidents* et *Connect-*

Win, nous remarquons que le taux de compacité θ est très bas (variant entre 1.64% et 4.84%). Ceci peut être expliqué par le fait que les valeurs de p_l et p_u, dans les différents hypergraphes, sont élevées mais principalement par les valeurs basses du nombre de transversalité, τ, sur ces jeux de données. En effet, τ est égal à 1 dans *Accidents1* et à 2 dans *Accidents2* et *ConnectWin*. A titre d'exemple, *ConnectWin*, malgré des valeurs de p_l et p_u relativement basses, présente un taux de compacité égal à 4.48%, essentiellement dû à la valeur de τ.

Dans le tableau 4.7, les temps d'exécution sont calculés en secondes et "-" signifie que le temps de traitement a dépassé les 3000 secondes. Comme le met en évidence le tableau, l'algorithme KS est considérablement lent par rapport à IMT-EXTRACTOR et MMCS. Toutefois, le dernier algorithme nécessite approximativement le même temps pour extraire les traverses minimales à partir de H et H' pour *Accidents2* et *Connect-Win*. Cette constatation n'est pas valide pour *Accidents1*. Par conséquent, pour ce dernier jeu de données, le temps d'exécution de IMT-EXTRACTOR, incluant pré-traitement et post-traitement, est moins important que celui requis par l'algorithme MMCS.

D'autre part, pour les jeux de données *Accidents2* et *Connect-Win*, l'hypergraphe irrédondant H' et ses traverses minimales irrédondantes ne nous permettent pas d'optimiser sensiblement le temps de calcul des traverses minimales de H. Pour *Connect-Win*, MMCS est même légèrement plus rapide que notre algorithme. Cette constatation est d'autant plus évidente lorsque les valeurs du taux de compacité de ces deux hypergraphes sont assez basses. En effet, le nombre de traverses minimales qui peuvent être déduites à partir de $\mathcal{M}_{H'}$ n'est pas assez important pour avoir un impact favorable sur notre approche. De la même manière, les trois hypergraphes testés ne renferment pas un très large ensemble de traverses minimales. Les expérimentations suivantes montreront que la réduction du temps d'exécution dépend fortement des valeurs de $|\mathcal{M}_H|$ et du nombre de transversalité τ.

	p_l	p_u	$\|\mathcal{X}\|$	$\|\xi\|$	$\|Sommets\_Gen\|$	$\tau(H)$
$H1$	0.07	0.13	95	51	65	8
$H2$	0.06	0.11	99	101	64	9
$H3$	0.03	0.06	117	20005	68	4
$H4$	0.05	0.05	60	20	20	20

TABLE 4.8 – Caractéristiques des hypergraphes aléatoires

Dans le second volet de nos expérimentations, nous considérons un ensemble de trois hypergraphes (H_1, H_2 et H_3) générés de manière aléatoire. A partir d'un certain nombre de sommets $|\mathcal{X}|$, et d'un nombre d'hyperarêtes $|\xi|$, nous générons, de manière indépendante, $|\xi|$ ensembles de sommets tels que chacun d'eux correspond à une hyperarête, de l'hypergraphe, choisie de façon aléatoire. Le quatrième jeu de données (H_4) testé est un hypergraphe qui est composé de $|\mathcal{X}|$ sommets, $|\mathcal{X}|/3$ hyperarêtes et la cardinalité de chaque hyperarête est égale à 3. Ce dernier jeu de données est composé d'un ensemble réduit d'hyperarêtes mais d'un nombre de traverses minimales qui est exponentiel, i.e., égal à $3^{|\mathcal{X}|/3}$.

Les caractéristiques de ces quatre hypergraphes aléatoires ($H1$, $H2$, $H3$ et $H4$) sont récapitulées dans le tableau 4.8.

	$\|\mathcal{M}_H\|$	$\|\mathcal{M}_{H'}\|$	θ
$H1$	832564740	358392	99,95%
$H2$	265765380	189444	99,92%
$H3$	1693	1250	26,17%
$H4$	3^{20}	1	99,99%

TABLE 4.9 – Statistiques sur les hypergraphes aléatoires

Dans le but d'analyser plus en profondeur les traverses minimales irrédondantes

utilisées par IMT-EXTRACTOR pour générer l'ensemble global des traverses minimales, considérons les statistiques données par le tableau 4.9 sur les valeurs de $|\mathcal{M}_H|$, $|\mathcal{M}_{H'}|$ et θ. Nous remarquons que, pour $H1$ par exemple, IMT-EXTRACTOR se base sur seulement 358392 traverses minimales irrédondantes pour retrouver les 832564740 traverses minimales que renferme $H1$. Ce qui représente un taux de compacité de 99.95%. L'algorithme MMCS n'extrait donc que 0.05% des traverses minimales qui sont utilisées, par la suite, par IMT-EXTRACTOR, pour retrouver l'ensemble $|\mathcal{M}_{H1}|$. Les taux de compacité des trois autres hypergraphes traités varient entre 26.17% et 99.99% et dépendent beaucoup de la structure de ces hypergraphes, à savoir les valeurs de p_l et p_u, mais aussi et surtout de leur nombre de transversalité. Généralement, plus ces valeurs sont basses, plus l'hypergraphe est éparse. Ceci implique, conformément à nos remarques préliminaires, un taux de compacité élevé et donc, un nombre très réduit de traverses minimales irrédondantes.

	$H1$	$H2$	$H3$	$H4$
MMCS(H)	2083,79	766,19	124,69	245,61
KS(H)	-	-	1041,29	18,55
MMCS(H')	2,740	0,37	87,54	0,001
GET-ALL-MT($\mathcal{M}_{H'}$)	11,69	1,53	0,03	19.80
IMT-EXTRACTOR (H)	**14,43**	**1,90**	**87,57**	**19.81**

TABLE 4.10 – Temps de traitement sur les hypergraphes aléatoires (en secondes)

Prenons à présent l'hypergraphe $H4$, notre approche représente les 3^{20} traverses minimales qu'il renferme par une et une seule traverse minimale irrédondante. Cette dernière est composée de tous les représentants des sommets généralisés de $H4$ dans la mesure où le nombre d'hyperarêtes est égal au nombre de transversalité $\tau(H)$. Par ailleurs, nous remarquons que les valeurs de p_l et p_u sont les plus basses des hypergraphes du tableau 4.9 et le nombre de transversalité est le plus élevé, i.e.,

égal à 20

D'autre part, les valeurs de $\mathcal{M}_H$ et de $\mathcal{M}_{H'}$ nous permettent d'établir un lien entre le taux de compacité et le nombre de transversalité. Plus le nombre du transversalité est élevé, plus le taux de compacité l'est aussi. Nous remarquons de surcroît que la cardinalité de $\mathcal{M}_H$ pour ces quatre hypergraphes ($H1$, $H2$, $H3$ et $H4$) est largement plus élevée que les hypergraphes traités du tableau 4.6. Ainsi, nous pouvons affirmer que le taux de compacité est bien meilleur quand le nombre de traverses minimales est élevé. En effet, un très large ensemble de traverses minimales accroît la capacité d'IMT-EXTRACTOR à réduire la taille de l'hypergraphe d'entrée.

Le tableau 4.10 récapitule les temps de traitement des algorithmes MMCS, KS et IMT-EXTRACTOR. Notre algorithme présente des performances intéressantes par rapport aux deux autres, en particulier sur l'hypergraphe H_1 où MMCS extrait les traverses minimales en 2084 secondes alors que notre algorithme le fait en 14 secondes. La différence est encore plus évidente sur H_2 où MMCS présente un temps de $766,19$ secondes alors que IMT-EXTRACTOR génère toutes les traverses en seulement $1,90$ seconde. KS, de son côté, ne s'exécute que sur les hypergraphes H_3 et H_4. Il est, d'ailleurs, le plus rapide des 3 algorithmes considérés, sur ce dernier hypergraphe. Sur H_1 et H_2, il présente des temps de traitement supérieurs à 3000 secondes. Ils ne sont donc pas pris en compte par le tableau 4.10. Notre représentation concise des traverses minimales permet donc d'optimiser singulièrement les temps de traitement sur ces hypergraphes aléatoires. De plus, les caractéristiques de ces hypergraphes nous permettent de dresser un profil du type d'hypergraphes sur lesquels IMT-EXTRACTOR est le plus efficace. En effet, notre algorithme se montre plus performant sur des hypergraphes ayant un nombre de transversalité élevé et tel que leurs hyperarêtes sont de petite taille et relativement disjoints. Au cours de nos expérimentations, nous avons pu remarquer que plus les hyperarêtes, d'un hypergraphe, sont disjoints, plus le nombre de sommets ayant une même extension

est grand.

4.6 Application de la représentation concise des traverses minimales au problème de l'inférence des dépendances fonctionnelles

Nous avons introduit, dans les sections précédentes, une représentation concise et exacte des traverses minimales basée sur la notion de l'irrédondance de l'information. Pour mettre en exergue l'intérêt de notre approche, nous nous sommes intéressés au problème de l'inférence des dépendances fonctionnelles où les traverses minimales ont été utilisées comme une solution intéressante pour optimiser le calcul de la couverture minimale de toutes les dépendances fonctionnelles d'une relation r. Ainsi, nous montrons dans ce qui suit qu'il est possible de réduire la couverture minimale, qui permet de retrouver toutes les dépendances fonctionnelles satisfaites par une relation donnée. Nous introduisons ainsi le processus à suivre pour calculer cette couverture minimale et montrons comment retrouver l'ensemble de toutes les dépendances fonctionnelles.

4.6.1 Notions de la théorie des BD relationnelles

Avant de présenter l'application de notre représentation concise des traverses minimales, nous commençons par rappeler quelques définitions de la théorie des bases de données relationnelles ainsi que la problématique de l'inférence des dépendances fonctionnelles.

Dans le premier chapitre, nous avons exposé diverses problématiques où les traverses minimales peuvent présenter des solutions. Ceci est vrai pour le problème de l'inférence des dépendances fonctionnelles comme en témoignent les travaux recen-

sés, dans la littérature, proposant d'optimiser le calcul du plus petit ensemble de dépendances fonctionnelles (appelé *couverture minimale*) qui permet de retrouver l'ensemble de toutes les dépendances fonctionnelles satisfaites par une relation r, à travers l'application des traverses minimales.

Dans la suite, on note $R = \{a_1, a_2, \ldots, a_n\}$ un ensemble fini d'attributs, appelé aussi schéma de relation. Une relation r est un ensemble fini de tuples $\{t_1, t_2, \ldots, t_n\}$ de R. Soit un sous-ensemble d'attributs X, $u[X]$ dénote la restriction du tuple u de r à X.

Définition 20 Dépendance fonctionnelle *[Mai83]*
*Soit un ensemble d'attributs R, une dépendance fonctionnelle (*df*) sur R est une expression de la forme $X \rightarrow Y$, tel que $X, Y \subseteq R$. Une* df *$X \rightarrow Y$ est satisfaite par r, noté $r \vDash X \rightarrow Y$, telle que r est une relation sur R, si pour tout tuple u et v de r, nous avons $u[X] = v[X] \Rightarrow u[Y] = v[Y]$.*

Nous notons par D_r l'ensemble de toutes les dépendances fonctionnelles satisfaites par la relation r.

Définition 21 Dépendance fonctionnelle minimale *[Mai83] Une dépendance fonctionnelle $X \rightarrow Y$ satisfaite par r ($r \vDash X \rightarrow Y$) est dite minimale si et seulement si $\nexists Z \subset X$ tel que $r \vDash Z \rightarrow Y$.*

Définition 22 Axiomes d'Armstrong *A partir d'un ensemble de dépendances fonctionnelles, nous pouvons en déduire d'autres grâce aux axiomes d'Armstrong qui représentent un système d'inférence complet et valide formé par les trois règles suivantes :*

- *Réflexivité : Si $Y \subseteq X$, alors $X \rightarrow Y$.*
- *Augmentation : Si $X \rightarrow Y$, alors $XZ \rightarrow Y$; nous avons aussi $XZ \rightarrow YZ$.*
- *Pseudo-transitivité : Si $X \rightarrow Y$ et $YZ \rightarrow W$, alors $XZ \rightarrow W$.*

Définition 23 FERMETURE D'UN ENSEMBLE DE DFS *[Mai83]*
Soient un ensemble F de dépendances fonctionnelles. Nous appelons fermeture de F, notée F^+, les dépendances fonctionnelles contenant F et tel qu'aucune autre dépendance fonctionnelle ne puisse être déduite de F.

La fermeture d'un ensemble de dépendances fonctionnelles peut ainsi être considérée comme étant l'ensemble de toutes les DFs pouvant être déduites par application des axiomes d'Armstrong. Cependant, cette fermeture dépend naturellement du schéma de relation R.

Définition 24 COUVERTURE*[Mai83]*
Soit D et G deux ensembles de dépendances fonctionnelles. D est une couverture de G si et seulement si $D^+ = G^+$.

Autrement dit, si r est une relation dans R et D un ensemble de dépendances fonctionnelles, si nous avons $r \vDash D$ alors toutes les DFs de D sont satisfaites par r. De plus, une dépendance $X \rightarrow Y$ est une *conséquence* de D si $r \vDash D$ implique $r \vDash X \rightarrow Y$ à travers les axiomes d'Armstrong, présentés dans la Définition 22. De ce fait, si D et G sont deux ensembles de dépendances fonctionnelles, D est considéré comme une couverture de G si toutes les dépendances de G sont des *conséquences* de D.

Définition 25 COUVERTURE CANONIQUE*[Mai83]*
Soit r une relation sur R et D_r l'ensemble de toutes les dépendances fonctionnelles satisfaites par r. La couverture canonique de D_r, noté COVER*(D_r) est définie comme suit :* COVER*$(D_r) = \{X \rightarrow Y \mid X \subset R, Y \in R, r \vDash X \rightarrow Y$ et $X \rightarrow Y$ est minimale$\}$.*

Notons, par ailleurs, que la couverture canonique est unique pour une relation r.

Définition 26 COUVERTURE IRRÉDONDANTE*[Mai83]*

Une couverture D est dite irrédondante si et seulement si $\nexists\ G \subset D$ tel que $G^+ = D^+$.

Définition 27 COUVERTURE MINIMALE*[Mai83]*

Une couverture est dite minimale si elle est canonique et irrédondante.

L'ensemble D_r peut avoir plusieurs couvertures équivalentes de différente tailles. Le problème d'inférence des dépendances fonctionnelles consiste donc à calculer la couverture de D_r qui soit minimale selon la Définition 27, et constituée du plus petit nombre de DFs dont la fermeture permet de retrouver toutes les DFs satisfaites par r.

Exemple 16 *Considérons la relation r, synthétisée par le tableau 4.11, où A, B, C, D et E représentent les attributs de R [Gas13]. Le tableau 4.12 résume l'ensemble de toutes les dépendances fonctionnelles satisfaites par r. Ainsi, nous pouvons remarquer que $AB \rightarrow D$ est une dépendance fonctionnelle satisfaite par r mais qui n'est pas minimale puisque, pour $A \subseteq AB$, nous avons $r \vDash A \rightarrow D$. $AB \rightarrow D$ n'appartient donc pas à* COVER*(D_r). D_r est l'ensemble de toutes les dépendances fonctionnelles satisfaites par r.*

Le problème de calcul d'une couverture minimale d'une relation r, plus connu sous le nom d'*inférence de dépendances fonctionnelles*, a intéressé bon nombre de chercheurs. Cet intérêt est motivé, principalement, par les multiples applications de ce problème dans divers domaines, comme la conception et l'analyse des bases de données, l'optimisation des requêtes, etc. [EG02]. Certains travaux se sont penchés sur la complexité du problème dans la mesure où l'espace de recherche des dépendances fonctionnelles est exponentiel en fonction du nombre d'attributs du schéma de relation considérée.

	A	B	C	D	E
t_1	1	100	1	2	50
t_2	4	101	1	2	50
t_3	1	102	2	2	70
t_4	1	200	1	2	50
t_5	2	101	3	3	100
t_6	2	200	1	3	70
t_7	1	100	3	2	50

TABLE 4.11 – Une relation r

$BE \rightarrow A$	$A \rightarrow D$	$AB \rightarrow E$
$BD \rightarrow A$	$CE \rightarrow D$	$BD \rightarrow E$
	$BE \rightarrow D$	$AC \rightarrow E$
		$CD \rightarrow E$

TABLE 4.12 – L'ensemble COVER(D_r) associé à la relation r

4.6.2 Problème de l'inférence des dépendances fonctionnelles

Heikki and Räihä ont montré dans leurs travaux que le problème de l'inférence de dépendances fonctionnelles peut être réduit à un problème d'extraction des traverses minimales [MR94]. En effet, si D_r représente toutes les dépendances fonctionnelles satisfaites par une relation r, la couverture minimale de D_r est formée d'un ensemble de dépendances fonctionnelles dont les prémisses sont les traverses minimales d'un hypergraphe. Les hyperarêtes de ce dernier représentent les complémentaires des ensembles en accord de r, pour chaque attribut de R.

C'est en s'inspirant de cette idée que Lopes *et al.* ont introduit l'algorithme DEP-MINER [LPL00], dont les expérimentations ont montré l'efficacité par rapport aux meilleurs algorithmes proposés dans la littérature et notamment celui de [HKPt98].

Dans le but de mettre en évidence l'intérêt de notre représentation concise et exacte des traverses minimales pour le problème d'inférence des dépendances fonctionnelles, nous nous appuyons sur ce travail de Lopes *et al.* qui introduisent les notions d'*ensemble en accord* et d'*ensemble maximal*.

Définition 28 ENSEMBLE EN ACCORD*[LPL00]*

Soient t_i et t_j deux tuples et X un ensemble d'attributs. Les tuples t_i et t_j sont en accord *sur X si $t_i[X] = t_j[X]$. Ainsi, l'ensemble en accord de t_i et t_j est défini comme suit : $Ag(t_i, t_j) = \{A \in R \mid t_i[A] = t_j[A]\}$. D'une manière générale, les ensembles en accord d'une relation r, que nous notons $Ag(r)$ sont calculés selon la formule suivante : $\{Ag(t_i, t_j) \mid t_i, t_j \in r, t_i \neq t_j\}$.*

Définition 29 ENSEMBLE MAXIMAL

Un ensemble maximal d'un attribut A est un ensemble d'attributs $X \subseteq R$, maximal au sens de l'inclusion, et qui, à partir d'un ensemble de dépendances fonctionnelles D, ne détermine pas A. L'ensemble de tous les ensembles maximaux de A dans une relation r, noté $\text{MAX}(D_r, A)$, est défini comme suit : $\text{MAX}(D_r, A) = \{X \subseteq R \mid r \not\models$

$X \rightarrow A$ et $\forall Y \subseteq R$, $X \subset Y$, $r \vDash Y \rightarrow A$}. De plus, nous notons cMAX*(D_r, A) les compléments des ensembles maximaux* MAX*(D_r, A) privés de A.*

Lemma 2 *Soit r une relation sur R, A un attribut de R et $Ag(r)$ tous les ensembles en accord de r. Nous avons* MAX*(D_r, A) = Max{ $X \in Ag(r) \mid A \notin X$ }.*

L'ensemble de toutes les dépendances fonctionnelles D_r peut être calculé à partir des ensembles maximaux. En effet, l'ensemble des prémisses (ou antécédents) de D_r pour un attribut A, noté LHS(D_r, A), est donné par la formule suivante : LHS(D_r, A) = $\{X \subseteq R \mid r \vDash X \rightarrow A$ et $\forall X' \subset X$, $r \nvDash X' \rightarrow A\}$. Par conséquent, l'ensemble des dépendances fonctionnelles de la forme $X \rightarrow A$ tel que $A \in R$ et $X \in$ LHS(D_r, A) représente une couverture minimale de D_r.

Dans leur approche, Lopes *et al.* ont remarqué que pour un attribut A, l'ensemble cMAX(D_r, A) représente un hypergraphe simple dont les traverses minimales constituent les prémisses des dépendances fonctionnelles de telle sorte que : $\mathcal{M}_{cMAX(D_r,A)}$ = LHS(D_r, A).

Pour une relation r, notre contribution intervient après la phase de calcul des différents cMAX, pour chaque attribut de R. A partir de chaque hypergraphe simple généré, nous extrayons les traverses minimales irrédondantes. En ce sens, les ensembles de sommets généralisés sont calculés et nous n'utiliserons que les attributs qui seront considérés comme des représentants. A partir de ces représentants, nous définissons une couverture minimale de D_r qui est succincte, i.e., formée d'un ensemble de dépendances fonctionnelles plus réduit que celles calculées par les approches existantes. La couverture calculée par ces dernières peut être retrouvée, à partir de notre ensemble succinct, de deux manières différentes. La première consiste à substituer chaque représentant par un attribut appartenant au même ensemble de sommets généralisés. Ainsi, si $X \rightarrow A$ est une dépendance fonctionnelle satisfaite par r et si X est le représentant d'un ensemble S de sommets généralisés tel que S = $\{X, Z\}$ alors nous avons $r \vDash Z \rightarrow A$. Pour résumer donc, à partir des traverses mini-

males irrédondantes, nous générons les traverses minimales, comme démontré dans le chapitre 4, et à partir de ces dernières nous retrouvons la couverture minimale telle que calculée par les approches existantes.

La seconde manière de retrouver cette couverture, est décrite par la proposition 6 et repose sur l'axiome de pseudo-transitivité d'Armstrong, présenté dans la Définition 22.

Proposition 6 *Soit H un hypergraphe généré par $c\text{MAX}(D_r, A)$, S un ensemble de sommets généralisés de H et X son représentant. $\forall\ Y \in S$ et $Y \neq X$, nous avons : $r' \vDash Y \rightarrow X$ tel que $r' \subset r$ est une relation composée des tuples dont les ensembles en accord sont les éléments de $\text{MAX}(D_r, A)$.*

Preuve 6 *Soit A un attribut de R. L'hypergraphe associé à A est $H_A = (\mathcal{X}_A, \xi_A)$ tel que $\mathcal{X}_A = R \cap \xi_A$ et $\xi_A = c\text{MAX}(D_r, A)$. L'ensemble $c\text{MAX}(D_r, A)$ représente les compléments des éléments de $\text{MAX}(D_r, A)$, lequel est constitué des éléments maximaux, au sens de l'inclusion, de $Ag(r)$ ne contenant pas l'attribut A.*

De ce fait, $\forall T \in \text{MAX}(D_r, A)$ tel que $T \subset R$, nous avons $\forall X \in T$, $\exists$ deux tuples t et t' tel que $t[X] = t'[X]$. Puisque $c\text{MAX}(D_r, A)$ représente les compléments des éléments de $\text{MAX}(D_r, A)$ alors nous pouvons en déduire que $\forall T' \in c\text{MAX}(D_r, A)$ tel que T' est le complément de T, alors $\forall X \in T'$, $t[X] \neq t'[X]$.

Considérons à présent un sommet généralisé S de H_A tel que $S = \{X, Y\}$ et X est le représentant de S. $\text{EXTENT}(X) = \text{EXTENT}(Y) \Leftrightarrow X$ et Y appartiennent aux mêmes hyperarêtes de H_A et donc aux mêmes éléments de $c\text{MAX}(D_r, A)$. Ainsi, $\forall T' \in c\text{MAX}(D_r, A)$ tel que $X \in T'$ et $Y \in T'$, et T' est le complément de T tel que $T = Ag(t, t')$, alors : $t[X] \neq t[Y]$ et $t'[X] \neq t'[Y]$.

Donc, pour une relation composée des tuples t et t', nous avons les deux dépendances fonctionnelles conditionnelles de r, $Y \rightarrow X$ et $X \rightarrow Y$. Pour notre approche, nous n'aurons besoin que d'une seule. Celle où le représentant est l'antécédent (prémisse).

En effet, pour un attribut donné, les éléments de l'ensemble MAX sont des ensembles en accord, maximaux au sens de l'inclusion. Ces éléments sont calculés à partir de couples de tuples. De ce fait, l'ensemble des tuples concernés représentent une relation r' au sein de la relation initiale r. De plus, suivant la Définition 28, les attributs appartenant à un même ensemble de sommets généralisés ont une même valeur au sein des tuples qui l'ont généré. Ainsi, la relation r' satisfait naturellement les dépendances fonctionnelles de type $Y \rightarrow X$ tel que X est le représentant de l'ensemble des sommets généralisés auquel appartient Y.

4.6.3 Etude de cas

Dans l'objectif d'illustrer l'intérêt de notre représentation concise et exacte des traverses minimales pour le problème de l'inférence des dépendances fonctionnelles, nous présentons dans cette section une étude de cas qui détaille la manière dont nous calculons une couverture plus concise, que les approches existantes, de l'ensemble des dépendances fonctionnelles pour une relation donnée.

Reprenons ainsi la relation r décrite par l'exemple 16. Le tableau 4.13 montre les différentes étapes qui mènent à la couverture minimale COVER(D_r) consistant en la génération des ensembles LHS pour chaque attribut de r. Le processus commence par calculer tous les ensembles en accord (ligne 1), Ag(r), selon la Définition 28. Ensuite, et à partir de AG(r), les ensembles maximaux, pour chaque attributs, sont générés (ligne 3). D'ailleurs, nous remarquons que les ensembles maximaux B et C sont, respectivement, $ACDE$ et $ABDE$. Ils n'ont pas été pris en considération dans la ligne 3 car leurs compléments CMAX, privés de B et C, sont égaux à l'ensemble vide. En ce sens, dans la ligne 4, nous ne trouvons que les ensembles CMAX des attributs A, D et E. Pour chaque attribut, chaque élément de CMAX représente une hyperarête de l'hypergraphe correspondant à l'attribut traité. Ainsi, si nous considérons par exemple l'attribut A, l'hypergraphe qui lui est associé (H_1) est constitué de deux

hyperarêtes $e_1^1 = \{B\}$ et $e_1^2 = \{D, E\}$.

Au final, nous obtenons trois hypergraphes simples (ligne 5). H_1 pour l'attribut A, H_2 pour D et H_3 pour E. Un calcul des traverses minimales (ligne 6) de chacun de ces hypergraphes permet de retrouver l'ensemble de dépendances fonctionnelles de COVER(D_r). Pour H_1, l'ensemble des traverses minimales $\mathcal{M}_{H_1}$ renferme BD et BE. Les deux dépendances fonctionnelles de COVER(D_r) qui en découlent sont : $BD \rightarrow A$ et $BE \rightarrow A$. Selon le même principe, nous retrouvons les neuf autres DF de COVER(D_r) résumées par le Tableau 4.12.

Notre approche intervient juste après la construction des trois hypergraphes H_1, H_2 et H_3. Au lieu d'extraire les traverses minimales à partir de ces hypergraphes, nous en déduisons les hypergraphes irrédondants correspondants, H_1', H_2' et H_3', à partir du calcul des ensembles de sommets généralisés (ligne 7). Les représentants sont marqués en gras et les autres attributs sont supprimés des hypergraphes initiaux. Une fois les hypergraphes irrédondants construits (ligne 8), nous calculons leurs traverses minimales correspondantes (ligne 9) et qui nous permettront de représenter l'ensemble COVER(D_r) de manière succincte.

Comme le montre le tableau 4.14, notre approche ne représente l'ensemble COVER(D_r) qu'avec 4 dépendances fonctionnelles absolues (satisfaites par r) alors que les approches existantes en calculent 9. Cet ensemble est noté COVER'(D_r). A partir de nos 4 dépendances fonctionnelles, nous allons examiner la manière avec laquelle nous retrouvons l'ensemble COVER(D_r) tel que représenté par le tableau 4.12 en suivant les deux méthodes présentées plus haut :

- En appliquant le processus de substitution sur les dépendances fonctionnelles de l'ensemble COVER'(D_r), nous retrouvons naturellement l'ensemble COVER(D_r). Pour cela, nous avons besoin des ensembles de sommets généralisés déjà calculés. En effet, pour l'attribut A, S_1^1 ne contient que l'attribut B alors que les attributs D et E appartiennent au même ensemble S_2^1 et puisque seulement le

Lopes *et al.*	$\text{Ag}(r) = \{CDE,\ AD,\ D,\ ACDE,\ B,\ C,\ E,\ BC, ABDE,\ DE,\ ADE\}$		
	A	D	E
	$\text{MAX}(A) = \{CDE,\ BC\}$	$\text{MAX}(D) = \{E,\ BC\}$	$\text{MAX}(E) = \{AD,\ BC\}$
	$\text{cMAX}(A) = \{B,\ DE\}$	$\text{cMAX}(D) = \{ABC,\ AE\}$	$\text{cMAX}(E) = \{BC,\ AD\}$
	$H_1 = (\mathcal{X}_1, \xi_1)$ $\mathcal{X}_1 = \{B,\ D,\ E\}$ $\xi_1 = \{e_1^1 = \{B\},$ $e_1^2 = \{D, E\}\}$	$H_2 = (\mathcal{X}_2, \xi_2)$ $\mathcal{X}_2 = \{A,\ B,\ C,\ E\}$ $\xi_2 = \{e_2^1 = \{A, B, C\},$ $e_2^2 = \{A, E\}\}$	$H_3 = (\mathcal{X}_3, \xi_3)$ $\mathcal{X}_1 = \{A,\ B,\ C,\ D\}$ $\xi_3 = \{e_3^1 = \{B, C\},$ $e_3^2 = \{A, D\}\}$
	$\mathcal{M}_{H_1} = \{BD,\ BE\}$	$\mathcal{M}_{H_2} = \{A,$ $BE,\ CE\}$	$\mathcal{M}_{H_3} = \{AB,\ AC,$ $BD,\ CD\}$
Notre contribution	$S_1^1 = \{\mathbf{B}\}$ $S_2^1 = \{\mathbf{D}, E\}$	$S_1^2 = \{\mathbf{A}\}$ $S_2^2 = \{\mathbf{B}, C\}$ $S_3^2 = \{\mathbf{E}\}$	$S_1^3 = \{\mathbf{B}, C\}$ $S_2^3 = \{\mathbf{A}, D\}$
	$H'_1 = (\mathcal{X}'_1, \xi'_1)$ $\mathcal{X}'_1 = \{B,\ D\}$ $\xi'_1 = \{e'^1_1 = \{B\},$ $e'^2_1 = \{D\}\}$	$H'_2 = (\mathcal{X}'_2, \xi'_2)$ $\mathcal{X}'_2 = \{A,\ B,\ E\}$ $\xi'_2 = \{e'^1_2 = \{A, B\},$ $e'^2_2 = \{A, E\}\}$	$H'_3 = (\mathcal{X}'_3, \xi'_3)$ $\mathcal{X}'_1 = \{A,\ B\}$ $\xi'_3 = \{e'^1_3 = \{B\},$ $e'^2_3 = \{A\}\}$
	$\mathcal{M}_{H'_1} = \{BD\}$	$\mathcal{M}_{H'_2} = \{A,\ BE\}$	$\mathcal{M}_{H'_3} = \{AB\}$
	$BD \to A$	$A \to D\,;\ BE \to D$	$AB \to E$

TABLE 4.13 – Les étapes de génération de notre représentation concise de COVER(D_r)

$BD \to A$	$A \to D$	$BE \to D$	$AB \to E$

TABLE 4.14 – L'ensemble COVER'(D_r)

représentant D a été considéré dans la construction de H'_1, nous n'avons généré qu'une dépendance fonctionnelle, pour A, qui est $BD \rightarrow A$. A partir donc de S_2^1, nous pouvons substituer D par E pour obtenir la dépendance fonctionnelle $BE \rightarrow A$. Cette dernière appartient effectivement dans COVER(D_r). De même, pour l'attribut D, $CE \rightarrow D$ est obtenue à partir de $BE \rightarrow D$ puisque B et C appartiennent à S_2^2. Enfin, pour l'attribut E, trois dépendances fonctionnelles sont obtenues à partir de $AB \rightarrow E$: $AC \rightarrow E$, $BD \rightarrow E$ et $CD \rightarrow E$.

– La seconde méthode qui permet de retrouver COVER(D_r) à partir de COVER'(D_r) serait d'appliquer l'axiome de pseudo-transitivité d'Armstrong. Selon la proposition 6, deux attributs X et Y qui appartiennent au même ensemble de sommets généralisés, tel que Y en est le représentant, sont liés par la dépendance fonctionnelle conditionnelle suivante : $X \rightarrow Y$. Cette relation serait "relative", et non absolue donc, car elle n'est vrai que par rapport à l'attribut à partir duquel les ensembles de sommets généralisés ont été calculés. Prenons l'exemple de l'attribut A où nous avons généré deux ensembles S_1^1 et S_2^1. Ce dernier ensemble est composé des attributs D et E. Nous pouvons donc déduire que la dépendance fonctionnelle $E \rightarrow D$ est satisfaite par les tuples qui ont permis de calculer les éléments de MAX(A).

  Analysons à présent l'ensemble MAX(A), il est constitué de deux éléments CDE et BC. Comme déjà expliqué, CDE et BC sont les ensembles en accord, de taille maximale, qui ne déterminent pas l'attribut A. L'ensemble en accord CDE a été généré par les tuples t_1 et t_2 alors que BC l'a été par les tuples t_4 et t_6. Si l'on considère donc uniquement la relation r' formée par les tuples t_1, t_2, t_4 et t_6, nous pouvons aisément vérifier que la dépendance fonctionnelle $E \rightarrow D$ est satisfaite par r'. Puisque $r' \subset r$, $E \rightarrow D$ est dite alors dépendance fonctionnelle conditionnelle. Ainsi, en appliquant l'axiome de pseudo-transitivité d'Armstrong, nous avons : $E \rightarrow D$ et $BD \rightarrow A$ engendrent $BE \rightarrow A$. Nous

retrouvons ainsi la troisième dépendance fonctionnelle de COVER(D_r) ayant A comme conclusion. Selon le même principe, nous générons pour l'attribut D, la dépendance fonctionnelle conditionnelle $C \rightarrow B$, et pour l'attribut E, $C \rightarrow B$ et $D \rightarrow A$. Le tableau 4.15 récapitule l'ensemble des DFs conditionnelles qui, conjuguée à notre couverture succincte COVER'(D_r) et à l'axiome d'Armstrong, permettent de retrouver COVER(D_r).

COVER'(D_r)	$BD \rightarrow A$	$A \rightarrow D$ $BE \rightarrow D$	$AB \rightarrow E$
Dep. Fonct. Cond.	$E \rightarrow D$	$C \rightarrow B$ $C \rightarrow B$	$D \rightarrow A$

+ Axiome d'Armstrong

COVER(D_r)	$BE \rightarrow A$ $BD \rightarrow A$	$A \rightarrow D$ $CE \rightarrow D$ $BE \rightarrow D$	$AB \rightarrow E$ $BD \rightarrow E$ $AC \rightarrow E$ $CD \rightarrow E$

TABLE 4.15 – Relation entre COVER'(D_r) et COVER(D_r)

Ainsi, nous avons montré qu'en se basant sur notre représentation des traverses minimales, nous pouvons calculer une couverture minimale de l'ensemble de toutes les dépendances fonctionnelles d'une relation donnée encore plus réduite que ce que présentent les approches existantes. Ceci a pour objectif de réduire les temps de traitement nécessaires à la génération d'une couverture minimale et minimiser l'ensemble de dépendances fonctionnelles permettant de retrouver l'ensemble dans son intégralité d'une relation donnée. L'étude de cas illustre ainsi le processus proposé et donne un aperçu des dépendances fonctionnels "éliminées" dans la nouvelle couverture minimale que nous proposons.

4.7 Conclusion

Dans ce chapitre, nous avons introduit une nouvelle approche pour le calcul des traverses minimales d'un hypergraphe. Cette approche présente une représentation concise et exacte de l'ensemble des traverses minimales en se basant sur la notion d'irrédondance dans les hypergraphes. Ceci nous a permis d'introduire un cadre méthodologique qui a donné naissance à un nouvel algorithme IMT-EXTRACTOR pour l'extraction des traverses minimales. L'étude expérimentale a confirmé l'intérêt de notre approche sur un certain type d'hypergraphes présentant des taux de compacité intéressants. Néanmoins, au-delà de l'aspect expérimental, nous souhaitons prouver l'intérêt de cette nouvelle représentation des traverses minimales en montrant sa répercussion sur un des nombreux domaines d'application des traverses minimales. Pour cela, nous avons choisi d'appliquer notre représentation concise et exacte des traverses minimales au problème d'inférence des dépendances fonctionnelles, pour calculer la couverture minimale d'une relation donnée. Cette contribution a été publiée dans une conférence avec comité de lecture [JLB13].Dans le chapitre suivant, nous introduisons une nouvelle méthode pour l'extraction des traverses minimales d'un hypergraphe en adoptant la stratégie "diviser pour régner". Nous démontrons que le partitionnement de l'hypergraphe initial en des hypergraphes partiels peut aussi s'avérer avantageux sur des hypergraphes particuliers.

Chapitre 5

"Diviser pour régner" pour l'extraction des traverses minimales d'un hypergraphe

5.1 Introduction

Optimiser le calcul des traverses minimales en optant pour la décomposition de l'hypergraphe d'entrée peut présenter une solution intéressante à condition de bien choisir le nombre optimal d'hypergraphes partiels de manière à éliminer des tests de minimalité des traverses calculées. En se basant sur le nombre de transversalité, nous proposons une approche basée sur la stratégie "diviser pour régner" pour l'extraction des traverses minimales. Un hypergraphe peut, en effet, être décomposé en un certain nombre d'hypergraphes partiels, égal à la taille de la plus petite traverse minimale que renferme l'hypergraphe d'entrée. Les traverses minimales extraites à partir des hypergraphes partiels, que nous appellerons "locales", permettent de retrouver l'ensemble de toutes les traverses minimales de l'hypergraphe initial. Les traverses obtenues et dont la cardinalité est égale au nombre de transversalité se-

ront considérées directement comme des traverses minimales et ce, sans vérifier par des tests, leurs minimalités. C'est ce que nous détaillons à travers ce chapitre via l'algorithme LOCAL-GENERATOR et à travers l'étude expérimentale dont il a fait l'objet.

5.2 Objectifs de la décomposition

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, comme le montre l'exemple 11 du chapitre 4 (page 71).

5.2.1 Diviser pour régner

Les algorithmes d'extraction des traverses minimales les plus performants [BMR03, KS05, MU13] sont des améliorations de l'algorithme de Berge [Ber89]. 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*", présenté dans la Définition 30.

Définition 30 DIVISER POUR RÉGNER *Le paradigme* diviser pour régner *se compose de trois étapes. La première est de* diviser *le problème en un certain nombre de sous-problèmes. La deuxième est de* régner *sur ces sous-problèmes en les résolvant de manière récursive ou directement. Enfin,* combiner *les solutions des sous-problèmes en une solution finale du problème initial.*

Le principe consiste à réduire ce nombre d'itérations en décomposant 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 décomposer 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.

Il est clair que cette approche ne saurait être efficace sur des hypergraphes, dont le nombre de tranversalité est très bas dans la mesure où le nombre d'hypergraphes partiels n'est pas conséquent et ne permet pas une optimisation intéressante du calcul des traverses minimales. Un profil du type d'hypergraphes sur lequel notre approche peut se montrer efficace est dressé au terme de l'étude expérimentale que nous détaillons à la fin de ce chapitre.

5.2.2 Originalité de l'approche

Nous avons détaillé dans le chapitre 2 les différents algorithmes dédiés à l'extraction des traverses minimales. Nous avons souligné que les algorithmes les plus performants étaient des améliorations de l'algorithme de Berge mais ils ne reposent pas sur le paradigme "diviser pour régner". En ce sens, notre approche est une extension complètement différente 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, nous nous proposons de traiter les hyperarêtes ensemble par ensemble. L'hypergraphe d'entrée H se trouve alors décomposé 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 test de la minimalité est nécessaire dans la mesure où les traverses minimales locales sont effectivement minimales mais au sein de l'hypergraphe partiel à partir duquel elles sont extraites mais rien n'assure qu'elles seront minimales dans l'hypergraphe initial. Comme nous l'avons déjà souligné, seules les traverses de taille égale au nombre de transversalité sont effectivement minimales puisqu'il ne peut exister une traverse incluse dedans.

Un seul algorithme, parmi tout ceux présentés dans le chapitre 2 met à profit cette notion de décomposition mais d'une façon différente. Il s'agit de celui de Bailey *et al.*, qui consiste à décomposer les hyperarêtes formées par un nombre important de sommets de manière à n'avoir que des hyperarêtes de relativement petite taille. Dans notre algorithme, l'hypergraphe est décomposé indépendamment de la taille de ses hyperarêtes.

Le nombre de transversalité d'un hypergraphe est la notion-clé, autour de laquelle est bâtie notre approche. 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. C'est sur cette élimination de certains de ces tests inutiles de la minimalité que nous comptons pour optimiser les temps d'extraction des traverses minimales.

5.3 Définitions et notations

Dans cette section, nous proposons de présenter des définitions clés et notations que nous utiliserons tout au long des sections suivantes. Pour aboutir à notre approche d'extraction des traverses minimales, basée sur la notion de traverse minimale locale, nous avons encore combiné des concepts de la théorie des hypergraphes (union et produit cartésien d'hypergraphes) avec d'autres de la fouille de données (ensemble essentiel, support), présentés dans les chapitres précédents.

Définition 31 UNION ET PRODUIT CARTÉSIEN *[Ber89]*

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 $H \cup G$:

$H \cup G = (\mathcal{X} \cup \mathcal{X}', \xi \cup \xi')$

$H \times G$ 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 7 *[Ber89]*

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 32 HYPERGRAPHE PARTIEL *[Ber89]*

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 notre approche, nous proposons d'étendre la proposition 7 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 triés par ordre

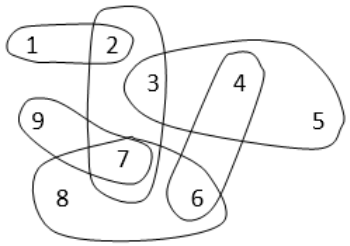
de support décroissant de sorte que x_1 est le sommet qui appartient au plus grand nombre d'hyperarêtes, nous proposons de construire k hypergraphes partiels H_i= $(\mathcal{X}_i, \xi_i)$, $i = 1, \ldots, k$ tels que :

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

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 $\mathcal{M}_{H_i}$.



	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$e_1 = \{1, 2\}$	1	1	0	0	0	0	0	0	0
$e_2 = \{2, 3, 7\}$	0	1	1	0	0	0	1	0	0
$e_3 = \{3, 4, 5\}$	0	0	1	1	1	0	0	0	0
$e_4 = \{4, 6\}$	0	0	0	1	0	1	0	0	0
$e_5 = \{6, 7, 8\}$	0	0	0	0	0	1	1	1	0
$e_6 = \{7, 9\}$	0	0	0	0	0	0	1	0	1

FIGURE 5.1 – Un exemple d'hypergraphe $H = (\mathcal{X}, \xi)$ et la matrice d'incidence IM_H correspondante

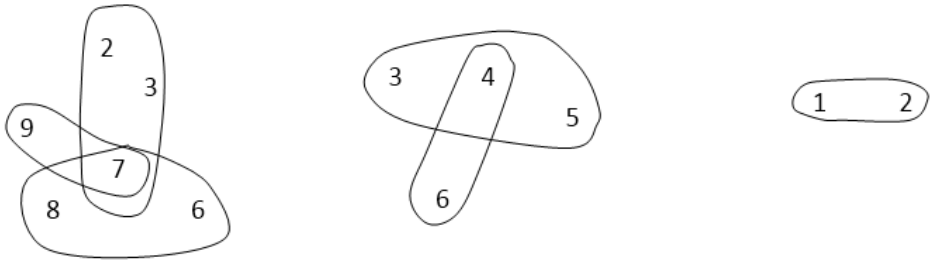


FIGURE 5.2 – Les 3 hypergraphes partiels dérivés de H : H_1, H_2 et $H3$

Exemple 17 *La figure 5.1 illustre un hypergraphe simple* $H = (\mathcal{X}, \xi)$ *tel que* $\mathcal{X} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ *et* $\xi = \{ e_1, e_2, e_3, e_4, e_5, e_6 \}$ *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* $e_6 = \{7, 9\}$. *H a un nombre de transversalité égal à* 3. *H possède* 2 *traverses minimales de cardinalité minimale égale à* 3 *:* $\{1, 4, 7\}$ *et* $\{2, 4, 7\}$. *Prenons, par exemple, la traverse minimale* $\{1, 4, 7\}$. *Après avoir ordonné les trois sommets le composant, selon un ordre décroissant de support, nous obtenons les trois hypergraphes partiels, présentés par la Figure 5.2, 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.*

5.4 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 que nous proposons consiste à construire, à partir de l'hypergraphe d'entrée H, k hypergraphes partiels (H_1, H_2, ..., H_k). 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.

L'algorithme LOCAL-GENERATOR, dont le pseudo-code, est décrit par l'Algorithme 17 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 11 du chapitre 3 (page 52). Comme déjà mentionné, cette fonction calcule et retourne une traverse minimale dont la taille est minimale et le nombre de transversalité k de l'hypergraphe. Ce dernier correspond désormais à la cardinalité de la traverse minimale retournée par la fonction. C'est à partir de cette traverse minimale retournée par GETMINTRANSVERSALITY que notre algorithme décompose l'hypergraphe d'entrée en des hypergraphes partiels.

Une fois la construction des k hypergraphes partiels (lignes 7-8) effectuée, l'algorithme LOCAL-GENERATOR fait appel à un algorithme d'extraction des traverses minimales pour calculer leurs traverses minimales locales [7], stockées dans $\mathcal{M}_{H_i}$ (ligne 9). Etant donné que les hypergraphes H_i sont relativement de petite taille, n'importe quel algorithme existant peut calculer les traverses minimales $\mathcal{M}_{H_i}$ en des temps très courts. Cet algorithme prend donc en entrée un hypergraphe partiel H_i de H, dont

7. Dans les expérimentations, nous avons utilisé l'algorithme MTMINER pour accomplir cette tâche et nous remercions les auteurs de nous en avoir fourni une version.

Algorithme 17: 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 $T = \text{GETMINTRANSVERSALITY2}(IM_H)$;
3 Ordonner les éléments de T par ordre décroissant du support ;
4 $k = \mid T \mid$;
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 $\mathcal{M}_{H_i} = \text{MTMINER}(H_i)$;
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** $(\mathcal{M}_H)$

l'ensemble des sommets $\mathcal{X}_i$ et l'ensemble des hyperarêtes ξ_i ont été déjà calculés (lignes 7 - 8) et extrait, 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 préparé 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 $\mathcal{M}_H$.

5.5 Etude de la complétude

Le résultat du produit cartésien de deux ensembles de traverses minimales, $\mathcal{M}_H$ et $\mathcal{M}_G$, représente un ensemble de traverses. La minimalité est ensuite vérifiée par le biais de la condition d'essentialité de la Définition 12. Dans le but de vérifier la complétude de notre approche, nous nous proposons de prouver par récurrence que $\mathcal{M}_H = \text{Min}\ \{\mathcal{M}_{H_1} \times \mathcal{M}_{H_2} \times \ldots \times \mathcal{M}_{H_k}\}$, en s'inspirant de la Proposition 7.

Proposition 8 *Toute traverse minimale de l'hypergraphe H peut être déduite à partir des ensembles de traverses minimaux des H_i, $i = 1 \ldots k$.*

Preuve 7 *Soient $H_1 = (\mathcal{X}_1, \xi_1)$ et $H_2 = (\mathcal{X}_2, \xi_2)$ deux hypergraphes et $H = H_1 \cup H_2$, avec $H = (\mathcal{X}, \xi)$ tel que $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ et $\xi = \xi_1 \cup \xi_2$. D'après Berge [Ber89], l'ensemble des traverses minimales de H est égal aux éléments, minimaux au sens de l'inclusion, appartenant au produit cartésien des ensembles minimaux de H_1 et H_2 :*

$\mathcal{M}_H = \mathcal{M}_{H_1 \cup H_2} = Min\ \{\ T \mid T \in \mathcal{M}_{H_1} \times \mathcal{M}_{H_2}\ \}$ ***(P1)***

Supposons que cette propriété est vraie pour l'union de k hypergraphes : $H' = H_1 \cup H_2 \cup \ldots \cup H_k$ avec $H' = (\mathcal{X}', \xi')$ et $H_i = (\mathcal{X}_i, \xi_i)$, $i = 1 \ldots k$.

Par hypothèse, nous avons alors :

$\mathcal{M}_{H'} = Min\ \{\ T \mid T \in \mathcal{M}_{H_1} \times \mathcal{M}_{H_2} \times \ldots \times \mathcal{M}_{H_k}\ \}$. ***(P2)***

Montrons, à présent, que la propriété est vraie pour l'union de $k+1$ *hypergraphes tel que* $H = H_1 \cup H_2 \cup \ldots \cup H_{k+1}$ *où* $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_{k+1}$ *et* $\xi = \xi_1 \cup \xi_2 \cup \ldots \cup \xi_{k+1}$. *On a :*

$H = H_1 \cup H_2 \cup \ldots \cup H_{k+1}$

$\Leftrightarrow H = H' \cup H_{k+1}$ *où* $H' = \cup_{i=1}^{k} H_i$ *et* $H' = (\mathcal{X}', \xi')$ *avec* $\mathcal{X}' = \cup_{i=1}^{k} \mathcal{X}_i$ *et* $\xi' = \cup_{i=1}^{k} \xi_i$.

D'après ***(P1)****, on a* $\mathcal{M}_H = Min\ \{\ T \mid T \in \mathcal{M}_{H'} \times \mathcal{M}_{H_{k+1}}\ \}$ *et d'après* ***(P2)****, nous avons* $\mathcal{M}_{H'} = Min\ \{\ T \mid T \in \mathcal{M}_{H_1} \times \mathcal{M}_{H_2} \times \ldots \times \mathcal{M}_{H_k}\ \}$. *Nous obtenons, au final donc :* $\mathcal{M}_H = Min\ \{\ T \mid T \in \mathcal{M}_{H_1} \times \mathcal{M}_{H_2} \times \ldots \times \mathcal{M}_{H_{k+1}}\ \}$.

Ceci prouve donc que toute traverse minimale de H, *i.e., de l'ensemble* $\mathcal{M}_H$, *peut être calculée à partir des traverses minimales locales des* k *hypergraphes partiels* H_i, $1 \leq i \leq k$.

Proposition 9 *Toute traverse minimale* T *déduite à partir des* $\mathcal{M}_{H_i}$, *tel que* $T = min\ \{T_1 \cup T_2 \cup T_3 \ldots \cup T_k\}$ *et* $T_i \in \mathcal{M}_{H_i}$, *est une traverse minimale de* H.

Preuve 8 $T_i \in \mathcal{M}_{H_i} \Rightarrow T_i \cap e_{ji} \neq \emptyset\ \forall e_{ji} \in \xi_i,\ 1 \leq i \leq k$.

Étant donné que $\bigcup_{i=1}^{k} \xi_i = \xi$ *et* $\forall (\xi_i, \xi_j) \in \xi \times \xi \backslash \{\xi_i\},\ \xi_i \cap \xi_j = \emptyset$, *donc* $\forall e \in \xi,\ \exists\ T_i \subset T$ *tel que* $T_i \cap e \neq \emptyset$. *Ainsi,* T *est une traverse de* H. *(1).*

Par vérification de la condition de minimalité de la Définition 12 (ligne 16 *de l'algorithme 17),* $T = min\ \{T_1 \cup \ldots \cup T_k\}$ *donc* T *est minimal dans* H *(2).*

(1) et (2) permettent de considérer T *comme une traverse minimale de* H.

5.6 Etude Expérimentale

Différentes expérimentations ont été réalisées sur des jeux de données variés afin d'évaluer l'algorithme LOCAL-GENERATOR. Le premier lot de jeux de données considérés correspond aux hypergraphes générés à partir des bases de données "*Accidents*" [8] et "*Connect-4*" [9], également utilisés dans le chapitre 4. Le deuxième lot contient des hypergraphes aléatoires générés (à l'aide du générateur "*random hypergraph generator*" implementé par Boros et *al.* [BEGK03]), en fonction du nombre de sommets, du nombre d'hyperarêtes et de la taille minimale des hyperarêtes. De plus, au cours de notre étude expérimentale, nous avons pris soin de vérifir que la borne maximale retournée par la fonction GETMINTRANSVERSALITY est bien égale au nombre de transversalité pour chaque hypergraphe traité. Ceci implique que les traverses générées par un produit cartésien des différents ensembles de traverses minimales locales et composées d'un nombre de sommets égal à la valeur retournée par GETMINTRANSVERSALITY sont bien des traverses minimales.

	$\|\mathcal{X}\|$	$\|\xi\|$	$\tau(H)$	$\|\mathcal{M}_H\|$	MMCS	KS	LOCAL-GENERATOR
Accidents1	81	990	1	1 961	0,30	8,620	1,52
Accidents2	336	10968	2	17 486	0,81	-	2,47
Connect-Win	79	12800	3	4 869 431	100,59	-	294,50

TABLE 5.1 – Caractéristiques et temps de traitement des hypergraphes Accidents et Connect (en secondes)

Les caractéristiques de chacun des hypergraphes du premier lot considéré sont rappellées dans le tableau 5.1. La première et la seconde colonne correspond, respectivement, au nombre de sommets et au nombre d'hyperarêtes des différents hy-

8. *http ://archive.ics.uci.edu/ml*
9. *http ://fimi.cs.helsinki.fi/data/*

pergraphes. 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 MMCS de [MU13] l'algorithme KS de [KS05] et notre algorithme LOCAL-GENERATOR. Le tableau 5.1 récapitule aussi les temps d'exécution de chaque algorithme sur chaque hypergraphe. L'algorithme MMCS 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 MMCS s'explique par le faible nombre de transversalité des 3 hypergraphes qui varie entre 1 et 3 comme indiqué dans le tableau 5.1. Dans ce cas, la décomposition 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.

	$\lvert\mathcal{X}\rvert$	$\lvert\xi\rvert$	$\tau(H)$	$\lvert\mathcal{M}_H\rvert$	MMCS	KS	LOCAL-GENERATOR
$H1$	96	52	8	832 564 740	2804,64	3911,431	1004,269
$H2$	95	51	9	5 040 431 550	3608,182	-	2899,088
$H3$	119	91	4	4 186 560 000	3115,226	-	1918,101
$H4$	159	142	20	7 158 203 125	5509,455	-	4775,364

TABLE 5.2 – Caractéristiques et temps de traitement des hypergraphes aléatoires (en secondes)

Le Tableau 5.2 récapitule les caractéristiques des différents hypergraphes que nous

avons générés. Ces données synthétiques ont été générées 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 832 564 740 et 7 158 203 125. Le nombre de transversalité, τ(H) variant de 4 à 20, est aussi élevé en comparaison avec les hypergraphes du Tableau 5.1. Ceci favorise donc notre approche puisque les hypergraphes d'entrée sont décomposés 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 de traitement en secondes, récapitulés dans le tableau 5.2, montrent que l'algorithme LOCAL-GENERATOR présente des temps plus intéressants que ceux obtenus par les algorithmes KS et MMCS. Notons que l'algorithme de [KS05] 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-GENERATOR et MMCS varie entre *709* et *1800* secondes. La différence de performances entre les tableaux 5.1 et 5.2 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 décomposer l'hypergraphe en plusieurs hypergraphes partiels.

De plus, le nombre de traverses minimales doit être 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. De plus, le gain en temps de traitement est aussi conditionné par le nombre des plus petites traverses minimales. En effet, pour ces dernières, notre algorithme n'effectue pas de test de la minimalité

et permet donc d'optimiser les temps de traitements nécessaires pour le calcul de toutes les traverses minimales.

5.7 Conclusion

Dans ce chapitre, 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 décomposer 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'hypergraphes renfermant des propriétés données. Cette approche a été publiée dans une conférence avec comité de lecture [JLB14b].

Conclusion générale et perspectives

Conclusion

Dans cette thèse, nous avons introduit trois nouvelles approches relatives aux traverses minimales. Deux d'entre elles s'intéressent au calcul de toutes les traverses minimales d'un hypergraphe alors que la troisième est basée sur un sous-ensemble précis de ces traverses minimales.

Cette dernière permet d'exploiter les traverses minimales pour identifier des acteurs-clés dans un réseau social représenté par un hypergraphe où chaque hyperarête correspond à une communauté du réseau. Pour déterminer ces acteurs importants, appelés multi-membres, notre approche propose d'extraire, à partir de l'hypergraphe d'entrée, les traverses minimales les plus petites. A partir de ces dernières, nous privilégions celles qui maximisent le nombre de sommets touchés. L'application de cette approche sur les réseaux sociaux a permis de vérifier que les traverses minimales multi-membres (TMM) que nous générons permettent de représenter l'ensemble du réseau avec le minimum de sommets. L'algorithme OM2D proposé et dédié au calcul des TMM affiche des performances intéressantes dans la mesure où il repose sur une notion importante, celle du nombre de transversalité de l'hypergraphe d'entrée. Cette idée clé permet de connaître le nombre de sommets que doivent renfermer les TMM avant même d'extraire ces derniers. Des expériences montrent que le calcul du nombre de transversalité, à travers un algorithme dédié et proposé au sein de

cette première approche, optimise sensiblement le calcul des plus petites traverses minimales.

Les contributions qui ont couronné cette première approche ont fait l'objet de trois publications. Une dans une revue internationale avec *impact factor* [JLB14a] et deux dans des conférences *avec comité de lecture* [JLB12b] [JLB12a].

C'est dans l'objectif de chercher une représentation concise et exacte de l'ensemble des traverses minimales qu'est proposée notre deuxième approche. S'appuyant sur la notion de l'irrédondance de l'information dans l'hypergraphe, notre approche classe les sommets en des sous-groupes suivant l'ensemble d'hyperarêtes auxquels ils appartiennent. Ceux appartenant au mêmes hyperarêtes sont classés ensemble et seulement l'un d'eux apparaîtra dans l'hypergraphe irrédondant que nous déduisons à partir de l'hypergraphe initial. Les traverses minimales irrédondantes de ce dernier sont les traverses minimales de l'hypergraphe irrédondant et représentent un ensemble concis qui permet de retrouver toutes les traverses minimales de l'hypergraphe initial. De manière évidente, plus l'ensemble des traverses minimales irrédondantes est restreint, plus cette approche se montre efficace, ce qui peut être quantifié à l'aide du taux de compacité que nous avons introduit et qui calcule le pourcentage de traverses minimales de l'hypergraphe initial qui sont générées sans être calculées directement.

Les traverses minimales irrédondantes nous permettent ainsi de proposer une représentation concise et exacte de l'ensemble des traverses minimales d'un hypergraphe donné. Cette représentation peut avoir des repercussions intéressantes dans divers domaines d'application. Nous avons choisi le problème de l'inférence des dépendances fonctionnelles pour l'illustrer, ce qui nous permis d'introduire une nouvelle couverture minimale de toutes les dépendances fonctionnelles d'une relation donnée.

Cette contribution a fait l'objet d'une publication dans une conférence *avec comité*

de lecture [JLB13].

Enfin, et toujours dans l'optique d'optimiser le calcul de toutes les traverses minimales d'un hypergraphe, nous avons proposé une troisième approche qui consiste à décomposer l'hypergraphe d'entrée en des hypergraphes partiels. Les traverses minimales de ces derniers sont calculées et leur produit cartésien permet de regénérer l'ensemble des traverses de l'hypergraphe par filtrage avec un test de minimalité. Le nombre d'hypergraphes partiels, consécutifs à la décomposition, n'est pas choisi au hasard. Il est égal au nombre de transversalité de l'hypergraphe initial afin d'éliminer les tests de la minimalité pour toutes les traverses constituées d'un nombre de sommet égal à ce nombre de transversalité. Notre approche a montré son intérêt, à travers une série d'expérimentations, mais pour un certain type d'hypergraphes caractérisés par un nombre de transversalité élevé, un nombre de traverses minimales extrêmement important et un nombre conséquent de traverses minimales de plus petite taille. Cette contribution a été validée par une conférence avec comité de lecture [JLB14b].

Perspectives

Si notre travail nous a permis d'obtenir des résultats encourageants, plusieurs perspectives et améliorations sont encore envisagées. La première des perspectives consiste à mettre davantage à profit les possibilités offertes par le nombre de transversalité d'un hypergraphe. Ainsi, il serait intéressant d'utiliser ce nombre de transversalité et les plus petites traverses minimales pour proposer une solution efficace au problème de la coloration d'hypergraphes.

La deuxième perspective est d'effectuer une série d'expérimentations détaillées sur l'application de notre représentation concise et exacte des traverses minimales au problème de l'inférence des dépendances fonctionnelles. Si nous avons montré que la couverture canonique et minimale que nous calculons est constituée d'un nombre

de dépendances fonctionnelles inférieur aux couvertures calculées par les méthodes existantes, il serait intéressant de vérifier, en pratique, si notre contribution permet d'optimiser singulièrement le calcul de ladite couverture d'une relation donnée.

Toujours dans l'optique de valoriser nos contributions, de nouvelles expérimentations sur les réseaux sociaux les plus utilisés dans le monde (Facebook, Twitter, Youtube, etc.), et donc nettement plus volumineux que ceux pris en compte dans l'étude expérimentale présentée dans ce manuscrit, sont nécessaires afin de jauger l'impact des TMM. L'utilisation d'algorithmes de détection des communautés permettra de construire les hypergraphes correspondants à ces réseaux.

Publications scientifiques acceptées

- M. Nidhal Jelassi, Christine Largeron et Sadok Ben Yahia. *Local-Generator : "diviser pour régner" pour l'extraction des traverses minimales d'un hypergraphe*. Extraction et la Gestion des Connaissances (EGC) 2014 : 245-256.
- M. Nidhal Jelassi, Christine Largeron et Sadok Ben Yahia. *Efficient unveiling of multi-members in a social network*. The Journal of Systems and Software (JSS)(94) 2014 : 30-38.
- M. Nidhal Jelassi, Christine Largeron et Sadok Ben Yahia. *Détection des traverses minimales par élimination de la redondance*. Extraction et Gestion des connaissances (EGC) 2013 : 169-174.
- M. Nidhal Jelassi, Christine Largeron et Sadok Ben Yahia. *A la recherche d'acteurs multi-communautaires dans un réseau social*. Conférence Francophone sur l'Apprentissage Automatique (CAP) 2012 : 238-252.
- M. Nidhal Jelassi, Christine Largeron et Sadok Ben Yahia. *Tmd-miner : Une nouvelle approche pour la détection des diffuseurs dans un système communautaire*. Extraction et Gestion des connaissances (EGC) 2012 : 423-428.

Liste des notations

H^t	L'hypergraphe transversal à H
H'	L'hypergraphe irrédondant associé à H
TM	Traverse minimale
IM_H	Matrice d'incidence correspondant à l'hypergraphe H
γ_H	Ensemble des traverses de H
$\mathcal{M}_H$	Ensemble des traverses minimales de H
$\tau(H)$	Nombre de transversalité de H
$Supp(X)$	Support de l'ensemble des sommets X
TMM	Traverse minimale multi-membre
$minsupp_u$	Support minimal de "utilisateurs" d'un tri-concept
EXTENT(x)	Extension du sommet x
DF	Dépendance fonctionnelle
F^+	Fermeture d'un ensemble de DFS
D_r	Ensemble de toutes les DFS satisfaites par une relation r
COVER(D_r)	Couverture canonique de D_r
Ag(t_i, t_j)	Ensemble en accord des tuples t_i et t_j
MAX(D_r, A)	Ensemble de tous les ensembles maximaux de A dans une relation r
CMAX(D_r, A)	Complément de MAX(D_r, A)
LHS(D_r, A)	Ensemble des prémisses de D_r pour un attribut A
Min	Minimalité au sens de l'inclusion

Bibliographie

[Ake78] S. B. Akers. Binary decision diagrams. *IEEE Trans. Computers*, 27(6) :509–516, 1978.

[ALTY08] N. Agarwal, H. Liu, L. Tang, and P. S. Yu. Identifying the influential bloggers in a community. In *Proceedings of the International Conference on Web Search and web Data Mining (WSDM '08)*, pages 207–218, Stanford, USA, 2008.

[AR94] R. Agrawal and S. Ramakrishnan. Fast algorithms for mining association rules in large databases. In *Proceedings of the 2nd International Conference on Very Large Data Bases (VLDB '94)*, pages 487–499, Santiago, Chili, 1994.

[AvG09] R. Abreu and A. J. C. van Gemund. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *Proceedings of the seventh Symposium on Abstraction, Reformulation, and Approximation (SARA'09)*, Minnesota, USA, 2009.

[BE92] S.P. Borgatti and M.G. Everett. Notions of position in social network analysis. In *Sociological methodology*, pages 1–35, 1992.

[BEGK03] E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA 2003)*, pages 556–567, Amsterdam, Netherlands, 2003.

[BEM08] E. Boros, K. Elbassioni, and K. Makino. On Berge multiplication for monotone boolean dualization. In *Proceedings of the 35th International colloquium on Automata, Languages and Programming, Part I*, ICALP '08, pages 48–59, 2008.

[Ber89] C. Berge. *Hypergraphs : Combinatorics of Finite Sets*. North-Holland, 3rd edition, 1989.

[BGKM03] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On maximal frequent and minimal infrequent sets in binary matrices. *Annals of Mathematics and Artificial Intelligence*, 39(3) :211–221, 2003.

[BHL+05] B. Benatallah, M. S. Hacid, A. Léger, C. Rey, and F. Toumani. On automating web services discovery. *The International Journal on Very Large Data Bases*, 14(1) :84–96, 2005.

[BI95] J.C. Bioch and T. Ibaraki. Complexity of identification and dualization of positive boolean functions. *Information and Computation*, 123(1) :50–63, 1995.

[BMR03] J. Bailey, T. Manoukian, and K. Ramamohanarao. 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)*, pages 485–488, Washington, USA, 2003.

[BRB90] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient implementation of a BDD package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC'90)*, pages 40–45, Florida, USA, 1990.

[CCL05] A. Casali, R. Cicchetti, and L. Lakhal. Essential patterns : A perfect cover of frequent patterns. In *Proceedings of the 7th International Conference on DaWaK*, pages 428–437, Copenhagen, Denmark, 2005.

[CH77] D. Cartwright and F. Harary. A graph theoretic approach to the investigation of system-environment relationships. *Journal of Mathematical Sociology*, 5 :87–111, 1977.

[CJB12] Ch.Trabelsi, N. Jelassi, and S. Ben Yahia. Scalable mining of frequent tri-concepts from folksonomies. In *Proceedings of the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2012*, pages 231–242, Kuala Lumpur, Malysia, May 2012.

[CWW10] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD'10)*, pages 1029–1038, Washington, USA, 2010.

[CWY09] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD '09)*, pages 199–208, Paris, France, 2009.

[CYZ10] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM'10)*, pages 88–97, Sydney, Australia, 2010.

[DL05] G. Dong and J. Li. Mining border descriptions of emerging patterns from dataset pairs. *Knowledge and Information Systems*, 8(2) :178–202, 2005.

[Dom05] P. Domingos. Mining social networks for viral marketing. *IEEE Intelligent Systems*, 20(1) :80–82, 2005.

[DQ13] N. Durand and M. Quafafou. 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'12*, pages 228–240, Lille, France, 2013.

[DR01] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66, New York, USA, 2001.

[DT99] J. Demetrovics and V. D. Thi. Describing candidate keys by hypergraphs. *Computers and Artificial Intelligence*, 18(2) :191–207, 1999.

[EDS07] E. Even-Dar and A. Shapira. A note on maximizing the spread of influence in social networks. In *Proceedings of the 3rd International Worshop on Internet and Network Economics (WINE'07)*, San Diego, USA, 2007.

[EG95] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6) :1278–1304, 1995.

[EG02] T. Eiter and G. Gottlob. Hypergraph transversal computation and related problems in logic and AI. In *Proceedings of the 4th European Conference on Logics in Artificial Intelligence*, JELIA '02, pages 549–564, 2002.

[Elb08] K. Elbassioni. On the complexity of monotone dualization and generating minimal hypergraph transversals. *Discrete Applied Mathematics*, 156(11) :2109–2123, 2008.

[FK96] M. L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21 :618–628, 1996.

[Fre79] L. C. Freeman. Centrality in social networks : Conceptual clarification. *Social Networks*, 1(3) :215–239, 1979.

[Gar06] G. C. Garriga. *Formal methods for mining structured objects.* Phd dissertation, Universitat Politècnica de Catalunya, 2006.

[Gas13] G. Gasmi. *Inférence des dépendances fonctionnelles en utilisant la fouille de données.* Thése de doctorat, Faculté des Sciences de Tunis, Université Tunis El Manar, 2013.

[GLL11] A. Goyal, W. Lu, and L.V.S Lakshmanan. Celf++ : optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference companion on World Wide Web (WWW '11)*, pages 47–48, Hyderabad, India, 2011.

[GMNR05] A. Gély, R. Medina, L. Nourine, and Y. Renaud. Uncovering and reducing hidden combinatorics in guigues-duquenne bases. In *Proceedings of The third International Conference on Formal Concept Analysis, ICFCA 2005*, pages 235–248, Lens, France, 2005. Springer-Verlag.

[Gra78] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6) :1420–1443, 1978.

[Hag08] M. Hagen. *Algorithmic and Computational Complexity Issues of MONET.* Phd dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2008.

[HBC07] C. Hébert, A. Bretto, and B. Crémilleux. A data mining formalization to improve hypergraph minimal transversal computation. *Fundamenta Informaticae*, 80(4) :415–433, 2007.

[HBN08] T. Hamrouni, S. Ben Yahia, and E. Mephu Nguifo. Succinct minimal generators : Theoretical foundations and applications. *International Journal on Foundations of Computer Science (IJFCS)*, 19(2) :271–296, 2008.

[Héb07] C. Hébert. *Extraction et usage de motifs minimaux en fouille de donnes, contribution au domaine des hypergraphes.* Thése de doctorat, Université de Caen, Basse Normandie, 2007.

[HKPt98] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *Proceedings of the 4th international IEEE international conference on Data Engineering*, pages 392–401, 1998.

[JG01] E. Muller J. Goldenberg, B. Libai. Talk of the network : A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3) :211–223, 2001.

[JLB12a] M.N. Jelassi, C. Largeron, and S. Ben Yahia. A la recherche d'acteurs multi-communautaires dans un réseau social. In *actes de la 12ème Conférence Francophone sur l'Apprentissage Automatique, CAP'12*, pages 238–252, 2012.

[JLB12b] M.N. Jelassi, C. Largeron, and S. Ben Yahia. Tmd-miner : Une nouvelle approche pour la détection des diffuseurs dans un système communautaire. In *actes de la 13ème Conférence Francophone sur l'Extraction et la Gestion des Connaissances, EGC'12*, pages 423–428, 2012.

[JLB13] M.N. Jelassi, C. Largeron, and S. Ben Yahia. Détection des traverses minimales par élimination de la redondance. In *actes de la 14ème Conférence Francophone sur l'Extraction et la Gestion des Connaissances, EGC'13*, pages 169–174, Toulouse, France, 2013.

[JLB14a] M.N. Jelassi, C. Largeron, and S. Ben Yahia. Efficient unveiling of multi-members in a social network. *The Journal of Systems and Software*, 94 :30–38, 2014.

[JLB14b] M.N. Jelassi, C. Largeron, and S. Ben Yahia. Local-generator : "diviser pour régner" pour l'extraction des traverses minimales d'un hypergraphe. In *actes de la 15ème Conférence Francophone sur l'Extraction et la Gestion des Connaissances, EGC'14*, pages 245–256, Toulouse, France, 2014.

[KKT03] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'03)*, pages 137–146, Washington, USA, 2003.

[KS05] D. J. Kavvadias and E. C. Stavropoulos. An efficient algorithm for the transversal hypergraph generation. *Journal of Graph Algorithms and Applications*, 9(2) :239–264, 2005.

[KS06] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *Proceedings of the 10th European Conference on Principles of Knowledge Discovery in Databases (PKDD'06)*, volume 4213, pages 259–271, Berlin, Allemagne, 2006.

[LKG+07] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD '07)*, pages 420–429, San Jose, Californie, USA, 2007.

[LPL00] S. Lopes, J. M. Petit, and L. Lakhal. Efficient discovery of functional dependencies and Armstrong relations. In *Proceedings of the 7th International Conference on Extending Database Technology*, pages 350–364, Konstanz, Germany, 2000.

[Mai83] D. Maier. *The theory of relational databases.* Commputer Science Press, 1983.

[Mar13] A. Mary. *Enumération des dominants minimaux d'un graphe.* Thése de doctorat, Université Blaise Pascal, Clermont-Ferrand, 2013.

[Mik07] P. Mika. Ontologies are us : A unified model of social networks and semantics. *Web Semantics : Science, Services and Agents on the World Wide Web*, 5(1) :5–15, March 2007.

[Mor34] J.L. Moreno. *Who shall survive ? : a new approach to the problem of Human Interrelations*, volume 58 of *Nervous and mental disease monograph series*. 1934.

[MP03] F. De Marchi and J. M. Petit. Zigzag : a new algorithm for mining large inclusion dependencies in database. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, ICDM'03, pages 27–34, Florida, USA, 2003.

[MR94] Heikki Mannila and Kari-Jouko Räihä. Algorithms for inferring functional dependencies from relations. *Data Knowledge Engineering*, 12(1) :83–99, 1994.

[MT97] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge discovery*, 1(3) :241–258, 1997.

[MU] K. Murakami and T. Uno. Hypergraph Dualization Repository (2013). {http://research.nii.ac.jp/~uno/dualization.html}. [Online ; accessed 19 Jun 2014].

[MU13] K. Murakami and T. Uno. Efficient algorithms for dualizing large-scale hypergraphs. In *Proceedings of the 15th Meeting on Algorithm Engineering and Experiments (ALENEX'13)*, pages 1–13, New Orleans, USA, 2013.

[OH10] T. Opsahl and B. Hogan. Growth mechanisms in continuously-observed networks : Communication in a facebook-like community. *CoRR*, abs/1010.2141, 2010.

[PT02] J. L. Pfaltz and C. M. Taylor. Scientific discovery through iterative transformations of concept lattices. In *Proceedings of the Workshop on Discrete Mathematics and Data Mining at 2nd SIAM Conference on Data Mining*, pages 65–74, Arlington, USA, 2002.

[RD02] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the of the 8th International Conference on Knowledge Discovery and Data mining (KDD '02)*, pages 61–70, Edmontom, Canada, 2002.

[Sch78] T. Schelling. *Micromotives and Macrobehavior*. 1978.

[Sco00] J. Scott. *Social Network Analysis : A Handbook*. Sage, 2000.

[STE07a] J. Scripps, P. N. Tan, and A. H. Esfahanian. Exploration of link structure and community-based node roles in network analysis. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM'07)*, pages 649–654, Omaha, USA, 2007.

[STE07b] J. Scripps, P. N. Tan, and A. H. Esfahanian. Node roles and community structure in networks. In *Proceedings of the 1st Workshop on Web Mining and Social Network Analysis (SNA-KDD'07)*, pages 26–35, San José, California, 2007.

[Tod13] T. Toda. Hypergraph transversal computation with binary decision diagrams. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, pages 91–102, Rome, Italy, 2013.

[TS05] D. V. Thi and H. N. Son. On the dense families in the relational datamodel. *ASEAN Journal on Science and Technology for Development*, 22(3) :241–249, 2005.

[WCSX10] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge discovery and Data mining (KDD'10)*, pages 1039–1048, New York, USA, 2010. ACM.

[WF94] S. Wasserman and K. Faust. *Social Network Analysis, methods and application*. Cambridge University Press, 1994.

Etude, représentation et applications des traverses minimales d'un hypergraphe.

Résumé : Cette thèse s'inscrit dans le domaine de la théorie des hypergraphes et s'intéresse aux traverses minimales des hypergraphes. L'intérêt pour l'extraction des traverses minimales est en nette croissance, depuis plusieurs années, et ceci est principalement dû aux solutions qu'offrent les traverses minimales dans divers domaines d'application comme les bases de données, l'intelligence artificielle, l'e-commerce, le web sémantique, etc. Compte tenu donc du large éventail des domaines d'application des traverses minimales et de l'intérêt qu'elles suscitent, l'objectif de cette thèse est donc d'explorer de nouvelles pistes d'application des traverses minimales tout en proposant des méthodes pour optimiser leur extraction. Ceci a donné lieu à trois contributions proposées dans cette thèse. La première approche tend à tirer profit de l'émergence du Web 2.0 et, par conséquent, des réseaux sociaux en utilisant les traverses minimales pour la détection des acteurs importants au sein de ces réseaux. La deuxième partie de recherche au cours de cette thèse s'est intéressé à la réduction du nombre de traverses minimales d'un hypergraphe. Ce nombre étant très élevé, une représentation concise et exacte des traverses minimales a été proposée et est basée sur la construction d'un hypergraphe irrédondant, d'où sont calculées les traverses minimales irrédondantes de l'hypergraphe initial. Une application de cette représentation au problème de l'inférence des dépendances fonctionnelles a été présentée pour illustrer l'intérêt de cette approche. La dernière approche s'est intéressée à la décomposition des hypergraphes en des hypergraphes partiels. Les traverses minimales de ces derniers sont calculées et leur produit cartésien permet de générer l'ensemble des traverses de l'hypergraphe. Les différentes études expérimentales menées ont montré l'intérêt de ces approches proposées.

Mots clés : Hypergraphe, Hypergraphe partiel, Traverse minimale, Multi-membre, Réseau social, Nombre de transversalité, Représentation concise, Irrédondance, Dépendance fonctionnelle, Couverture minimale.

Abstract: This work is part of the field of the hypergraph theory and focuses on hypergraph minimal transversal. The problem of extracting the minimal transversals from a hypergraph received the interest of many researchers as shown the number of algorithms proposed in the literature, and this is mainly due to the solutions offered by the minimal transversal in various application areas such as databases, artificial intelligence, e-commerce, semantic web, etc. In view of the wide range of fields of minimal transversal application and the interest they generate, the objective of this thesis is to explore new application paths of minimal transversal by proposing methods to optimize the extraction. This has led to three proposed contributions in this thesis. The first approach takes advantage of the emergence of Web 2.0 and, therefore, social networks using minimal transversal for the detection of important actors within these networks. The second part of research in this thesis has focused on reducing the number of hypergraph minimal transversal. A concise and accurate representation of minimal transversal was proposed and is based on the construction of an irredundant hypergraph, hence are calculated the irredundant minimal transversal of the initial hypergraph. An application of this representation to the dependency inference problem is presented to illustrate the usefulness of this approach. The last approach includes the hypergraph decomposition into partial hypergraph the "local" minimal transversal are calculated and their Cartesian product can generate all the hypergraph transversal sets. Different experimental studies have shown the value of these proposed approaches.

Key words: Hypergraph, Partial hypergraph, minimal transversal, Multi-member, Social network, transversality level, Concise representation, Irredundance, Functional dependency, Minimal cover.