l'université nantes angers le mans
COMMUNAUTÉ D'UNIVERSITÉS ET ÉTABLISSEMENTS

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Anthony COUTANT

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
sous le label de l'Université de Nantes Angers Le Mans

École doctorale : Sciences et technologies de l'information, et mathématiques

Discipline : Informatique et applications, section CNU 27
Unité de recherche : Laboratoire d'informatique de Nantes-Atlantique (LINA)

Soutenue le 5 novembre 2015
Thèse n° : ED 503

Modèles Relationnels Probabilistes et Incertitude de Références

Apprentissage de structure avec algorithmes de partitionnement

JURY

Rapporteurs :	**M. Christophe GONZALES**, Professeur des Universités, Université Paris VI
	M. Julien VELCIN, Maître de conférences (HDR), Université de Lyon II
Examinateurs :	**M. Rushed KANAWATI**, Maître de conférences, Université Paris XIII
	M. Eric GAUSSIER, Professeur des Universités, Université Joseph Fourier (Grenoble I)
Invité :	**M. Kamel MEKHNACHA**, Directeur R&D, ProbaYes
Directeur de thèse :	**M. Philippe LERAY**, Professeur des universités, Université de Nantes
Co-encadrant de thèse :	**M. Hoel LE CAPITAINE**, Maître de conférences, Université de Nantes

Remerciements

Le doctorat est un exercice long et difficile et cette thèse n'aurait certainement pas abouti sans la présence de nombreuses personnes à mes côtés.

Je remercie tout d'abord mes encadrants, M. Philippe Leray et M. Hoel Le Capitaine, qui m'ont accompagné durant ces trois années, pour leurs conseils, la confiance et l'autonomie qu'ils m'ont accordés, et leur disponibilité précieuse qui m'a permis de toujours avoir un appui lorsque j'en avais besoin.

Je voudrais également remercier mes compagnons lors des groupes de travail autour des modèles relationnels probabilistes et de la plate-forme PILGRIM, en doctorat ou à l'occasion de leur stage, notamment Mme. Rajani Chulyadyo et Mme. Mouna Ben Ishak, avec qui j'ai étroitement collaboré.

Plus généralement, merci aux membres des anciennes équipes *COnnaissances et Décisions* (COD) et *Gestion de Résumés et d'Information Multimédia* (GRIM), constituant désormais l'équipe *Data, User, Knowledge* (DUKe), pour les discussions passionnantes en réunion, aux détours de couloirs ou devant la machine à café, ainsi que pour l'ambiance de travail décontractée et chaleureuse, tant sur le plan de la recherche que sur le plan de l'enseignement sur le site de la Chantrerie de Nantes.

Je voudrais également remercier M. Christophe Gonzales et M. Julien Velcin d'avoir accepté d'être rapporteurs de ma thèse et M. Rushed Kanawati pour avoir consenti à en être examinateur. Je remercie de surcroît M. Eric Gaussier d'avoir fait partie de mon comité de suivi de thèse et me faire l'honneur de présider mon jury pour la soutenance. Ainsi que Mme. Nahla Ben Amor pour sa participation éclairée à mon comité de suivi durant ces années.

Sur un plan plus personnel, je souhaiterais remercier ma famille qui a cru en mon projet et m'a suivi dans mes choix. J'aimerais de plus remercier mes amis qui m'ont encouragé et offert des occasions de décompresser lorsque cela était nécessaire : Cédric, Nicolas, Frédéric, Freddy, Mylène, Grégory, Lionel, Véronique, Camille, Samantha, Jules, Clément, Bertrand, Amandine, Guillaume, Claire, Rodolphe, Marion, Simon, et tous les autres qui se reconnaîtront.

Enfin, Merci plus particulièrement à ma Maman et Marine dont le soutien au quotidien a été plus que déterminant.

Table des matières

Liste des tableaux

Table des figures

Liste des Algorithmes

Nomenclature

Clustering et Algèbre linéaire

$\bar{V}$ Matrice indicatrice du partitionnement P_V.

λ_i Paramètre de régularisation dans une fonction objectif pour en équilibrer les termes.

$\mathcal{G}(\mathcal{R}^j, \mathcal{I}) = (\mathcal{V}^j(\mathcal{I}), \mathcal{E}^j(\mathcal{I}))$ L'hyper graphe support du schéma de relation $\mathcal{R}^j$ pour l'instance $\mathcal{I}$.

$\mathcal{R}^*_{j\rightarrow}$ La séquence de schémas de relation pointés par les références successives de $fk(\mathcal{R}^j)$, avec dupliquas.

$\mathcal{R}^{j\rightarrow}$ La séquence de schémas de relation pointés par les références successives de $fk(\mathcal{R}^j)$, sans dupliquas.

$\|A\|_F^2$ Norme de Frobenius au carré de la matrice A.

$\tilde{R}_{12}$ Une matrice approximant R_{12} (généralement par produit de matrices facteur de dimensions inférieures).

$\underset{\mathbf{\Phi}}{\Delta}$ La divergence de Bregman calculée à partir de la fonction convexe $\mathbf{\Phi}$ entre deux matrices.

$\underset{f}{\Delta}$ La f-divergence calculée à partir de la fonction f entre deux matrices.

$\underset{I}{\Delta}$ La I divergence calculée entre deux matrices.

$\underset{KL}{\Delta}$ La KL divergence calculée entre deux matrices.

$\underset{pW}{\Delta}$ La norme $l-p$ calculée entre deux matrices et pondérée par une matrice W.

$\underset{p}{\Delta}$ La norme $l-p$ calculée entre deux matrices.

$A(ij)$ Élement de la matrice A situé à la i^{eme} ligne et j^{ieme} colonne.

$cut(\mathcal{G}, P_V)$ Coupe totale d'un graphe $\mathcal{G}$ dont les nœuds sont partitionnés en P_V.

D Matrice diagonale telle que $D(ii)$ contient le degré de l'individu i dans la matrice d'affinités dont D est issue.

G_i La matrice facteur pour le i^{eme} mode de données.

I Matrice identité.

L Matrice laplacienne.

$M[., c]$ Restriction de M à un ensemble c de ses colonnes.

$M[r, .]$ Restriction de M à un ensemble r de ses lignes.

$M_{\mathcal{I}}(\mathcal{R}_a^i)$ Matrice issue de la restriction de l'instance $\mathcal{I}$ au type d'association binaire $\mathcal{R}_a^i$.

$ncut(\mathcal{G}, P_V)$ Coupe normalisée d'un graphe $\mathcal{G}$ dont les nœuds sont partitionnés en P_V.

Q Mesure de modularité d'un graphe partitionné.

R_{12} Une matrice d'associations entre deux ensembles d'individus potentiellement disjoints.

$rcut(\mathcal{G}, P_V)$ Ratio de coupe d'un graphe $\mathcal{G}$ dont les nœuds sont partitionnés en P_V.

S Matrice facteur liant les G_i pour la tri factorisation non négative de matrices.

s_A Une fonction de similarités d'un ensemble de tuples utilisant une restriction de leurs attributs A.

$s_j(\mathcal{I}(\mathcal{R}^j)) = W_j(\mathcal{I})$ Une fonction de similarité évaluée pour tous les individus de la restriction au schéma $\mathcal{R}^j$ d'une instance $\mathcal{I}$. Le résultat est une matrice de similarités W_j.

$s_j(t1, t2)$ Une fonction de similarité évaluée pour deux tuples $t1$ et $t2$ appartenant au schéma de relation $\mathcal{R}^j$.

$tr(X)$ Trace de la matrice X.

W Matrice de poids ou de similarités entre individus d'un même ensemble.

X Une matrice.

X^t La matrice transposée de X.

Modèles Probabilistes

α Hyper paramètre singleton de Dirichlet représentant un nombre global d'individus synthétiques équivalents, pour le score Equivalent (Uniform) Bayesian Dirichlet.

α_{aijk} Hyper paramètre de Dirichlet pour la i^{eme} variable, rattachée au a^{ieme} schéma de relation, d'un MRP dans le k^{ieme} état de son domaine de définition sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

α_{aij} Somme des hyper paramètres de Dirichlet pour la i^{eme} variable, rattachée au a^{ieme} schéma de relation, d'un MRP dans tous ses états possibles sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

α_{ijk} Hyper paramètre de Dirichlet pour la i^{eme} variable d'un réseau bayésien dans le k^{ieme} état de son domaine de définition sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

α_{ij} Somme des hyper paramètres de Dirichlet pour la i^{eme} variable d'un réseau bayésien dans tous ses états possibles sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

$\boldsymbol{\alpha}$ Vecteur des hyper paramètres de Dirichlet.

Δs_{MRP} Fonction de calcul incrémental de score pour un MRP et ses extensions.

Δs_{RB} Fonction de calcul incrémental de score pour un RB.

$\gamma(\mathcal{G}_j, W_{\mathcal{R}^{j\rightarrow}}, \Phi_{\mathcal{R}^{j\rightarrow}})$ Une fonction de coût pour un partitionnement de $\mathcal{G}_j$, étant données des matrices de similarités des individus impliqués.

γ_{IR} La fonction d'évaluation d'un MRP-IR avec a priori sur les fonctions de partition, sur les structures et les paramètres.

$\hat{\theta}$ Paramètre de modèle graphique probabiliste optimisant le maximum de vraisemblance du modèle.

$\hat{P}(A)$ Estimateur d'une fonction de probabilités P sur A.

$\mathbf{u}$ Vecteur de valeurs d'une séquence de variables aléatoires.

$\mathcal{G}(\mathcal{M}, \mathcal{I}) = (V_{\mathcal{M},\mathcal{I}}, pa_{\mathcal{I}}, \Theta_{\mathcal{I}})$ Le réseau bayésien à plat (RBP) d'un MRP $\mathcal{M}$ pour l'instance $\mathcal{I}$ composé de son ensemble de variables aléatoires, sa fonction de parents, et ses paramètres.

$\mathcal{M} = (\mathcal{S}, \Phi, \Theta_{\mathcal{S},\Phi})$ La définition complète d'un MRP-IR ou MRP-IC $\mathcal{M}$ composé de structure graphique $\mathcal{S}$, son ensemble de fonctions de partition Φ et son ensemble de paramètres $\Theta_{\mathcal{S},\Phi}$.

$\mathcal{M} = (\mathcal{S}, \Theta)$ La définition complète d'un MRP comprenant sa structure graphique $\mathcal{S}$ et ses paramètres Θ.

$\mathcal{S} = (V, pa)$ La structure graphique d'un modèle probabiliste définie sur l'ensemble de variables (ou nœuds) V dont la liste des dépendances est déterminée par la fonction de parents pa.

$\mathcal{V}$ L'ensemble des variables d'un MRP ou d'une extension.

$\Phi_{\mathcal{R}^{j\rightarrow}}$ L'ensemble des fonctions de partition pour les contraintes de références du schéma $\mathcal{R}^j$.

$\phi_{a,i}$ Fonction de partition associée à la variable sélecteur $S_{a,i}$.

Ψ Un espace de fonctions d'agrégations.

Θ L'ensemble des paramètres des distributions de probabilités d'un modèle graphique probabiliste.

a Scalaire indiquant une valeur d'une variable aléatoire unique.

$A_{a,i}$ i^{eme} variable aléatoire d'un MRP associée à un attribut descriptif du a^{ieme} schéma relationnel.

$C^S_{a,i}$ Variable cluster source correspondant à la variable de référence $R_{a,i}$ dans un MRP-IC. Équivalent d'une variable sélecteur dans un MRP-IR.

$C^T_{k,a,i}$ Variable cluster cible correspondant à la variable de cluster source $C^S_{i,j}$ dont la contrainte de référence associée pointe vers le schéma de relation $\mathcal{R}^k$.

$c_{a,i}$ Nombre de clusters de la fonction de partition $\phi_{a,i}$.

$comp(\mathcal{M})$ La version complétée du MRP-IR ou MRP-IC $\mathcal{M}$.

$dim(\mathcal{S})$ La dimension d'une structure de modèle graphique probabiliste $\mathcal{S}$, c.-à-d. son nombre de paramètres indépendants.

N_{aijk} Fréquence dans un jeu de données où une variable, rattachée au a^{ieme} schéma de relation, d'un MRP est dans le k^{ieme} état de son domaine de définition sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

N_{aij} Somme des fréquences dans un jeu de données pour la i^{eme} variable, rattachée au a^{ieme} schéma de relation, d'un MRP dans n'importe quel état de son domaine de définition sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

N_{ijk} Fréquence dans un jeu de données où une variable d'un réseau bayésien est dans le k^{ieme} état de son domaine de définition sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

N_{ij} Somme des fréquences dans un jeu de données pour une variable d'un réseau bayésien dans n'importe quel état de son domaine de définition sachant que les parents de cette variable dans le graphe sont dans leur j^{ieme} état.

$nmi(A, B)$ L'information mutuelle normalisée des variables A et B.

$P(A)$ Distribution de probabilités de A.

pa Fonction de parents définissant la structure graphique d'un modèle probabiliste.

$R_{a,i}$ i^{eme} variable aléatoire d'un MRP associée à une contrainte de référence du a^{ieme} schéma relationnel.

$S_{a,i}$ Variable sélecteur correspondant à la variable de référence $R_{a,i}$.

$V_{a,i}$ Variable aléatoire de n'importe quelle forme $A_{a,i}$, $R_{a,i}$, et soit $S_{a,i}$ pour les MRP-IR, soit $C^S_{a,i}$ ou $C^T a, i, j$ pour les MRP-IC.

$x_{t,i,j}$ L'instanciation dans un RBP pour le tuple t de la variable aléatoire $X_{i,j}$ du MRP correspondant.

Paradigme Relationnel et Théorie des Ensembles

$\setminus$ Division ensembliste.

$\cap$ Intersection ensembliste.

$\cup$ Union ensembliste.

$\emptyset$ Ensemble vide.

$\in$ Appartenance ensembliste.

$\langle A_1, \ldots, A_k \rangle$ Une séquence

$\mathbf{R} = \{\mathcal{R}^i\}_{i \leq m}$ Un schéma relationnel composé de m schémas de relation.

$\mathcal{I} \in inst(\mathbf{R})$ Une instance de $\mathbf{R}$.

$\mathcal{I}(\mathcal{R})$ La restriction d'une instance $\mathcal{I}$ à un de ses schémas de relation $\mathcal{R} \in \mathbf{R}$.

$\mathcal{P}(S)$ L'ensemble des parties de l'ensemble S.

$\mathcal{R} = (\mathcal{H}, \mathcal{C})$ Un schéma de relation défini par une séquence d'attributs $\mathcal{H}$ et un ensemble de contraintes $\mathcal{C}$

$\mathcal{R}^i[A_i] \subseteq \mathcal{R}^j[A_j]$ Une dépendance d'inclusion du sous-ensemble A_i du schéma de relation $\mathcal{R}^i$ envers le sous-ensemble A_j du schéma de relation $\mathcal{R}^j$

$\pi = \pi(\mathcal{I})$ Le squelette relationnel de l'instance $\mathcal{I}$ pour un MRP, ou son squelette objet pour un MRP-IR et un MRP-IC.

$\pi_A(R)$ La projection d'une relation R à un sous-ensemble de ses attributs A, c.-à-d. l'ensemble des restrictions de ses tuples sur A.

$\pi_A(t)$ La projection d'un tuple t à un sous-ensemble de ses attributs A, c.-à-d. sa restriction sur A.

$\rightarrow$ Implication logique.

$\sigma = (\sigma_i)_{i \leq l}$ Une chaîne de références de longueur l.

$\Sigma(\mathbf{R})$ L'ensemble infini des chaînes de référence dans $\mathbf{R}$

$\sigma_{\mathcal{C}}(R)$ La sélection des tuples d'une relation satisfaisant la contrainte $\mathcal{C}$.

$\sigma_i = (\mathcal{R}^i, f_i)$ Un élément direct d'une chaîne de référence.

$\sigma_i = (\mathcal{R}^i, f_i)^{-1}$ Un élément inversé d'une chaîne de référence.

$\subset$ Inclusion ensembliste.

$\subseteq$ Inclusion ensembliste avec possible égalité.

$\times$ Produit cartésien.

$\{A_1, \ldots, A_k\}$ Un ensemble

$A = (N, D)$ Un attribut avec le nom N et le domaine D

$attr(\mathcal{R})$ La séquence d'attributs d'un schéma de relation

D Un domaine de données

$dom(\sigma_i)$ Le domaine d'un élément de chaîne de références.

$dom(A)$ Le domaine de l'attribut A

$fk(\mathcal{R})$ La séquence de contraintes d'intégrité référentielles d'un schéma de relation $\mathcal{R}$

$fkattr(\mathcal{R}, f_i)$ L'ensemble des attributs source de la contrainte d'intégrité référentielle f_i pour un schéma de relation $\mathcal{R}$

$fktarget(\mathcal{R}, f_i)$ Le schéma de relation cible de la contrainte d'intégrité référentielle f_i du schéma de relation $\mathcal{R}$

H Un en-tête, c.-à-d. une séquence d'attributs avec des noms différents

$id(S)$ La fonction d'agrégation identité, qui retourne l'ensemble S lui-même.

$inst(\mathbf{R})$ L'ensemble des instances du schéma relationnel $\mathbf{R}$

$pk(\mathcal{R})$ L'ensemble des attributs de clé primaire d'un schéma de relation $\mathcal{R}$

$R = (H, T)$ Une relation avec un en-tête H et un ensemble de tuples T

$ran(\sigma_i)$ Le co-domaine d'un élément de chaîne de références.

$ref_{\mathbf{R}}(\mathcal{R}, \sigma)$ Le schéma de relation atteint en partant de $\mathcal{R} \in \mathbf{R}$ et suivant la chaîne de références σ dans le schéma de base de données $\mathbf{R}$

T Un ensemble de tuples

t Un tuple

$t[U]$ La restriction d'un tuple t à ses valeurs d'attributs U

Divers

Γ La fonction Gamma généralisant la fonction factorielle aux nombres réels.

$\propto$ Opérateur de proportionnalité.

1

Introduction

Sommaire

1.1 Contexte

Nous sommes entourés de données. Celles-ci peuvent avoir la forme de descriptions d'objets du monde qui nous entoure, comme la couleur des yeux d'une personne, la longueur d'un solide, la température de fusion d'un élément, les coordonnées géographiques d'un lieu, la durée d'un événement ou encore n'importe quelle donnée taxonomique d'une espèce vivante. Elles peuvent également décrire des concepts plus abstraits comme ceux d'événements ponctuels ou continus, associant potentiellement divers objets du monde réel, p.ex. lorsqu'une personne consomme un aliment, lit un livre, aime un artiste, ou est amie avec quelqu'un. En langage logique, nous parlons de *fait* pour définir une donnée qui est alors assimilée conceptuellement à une réalité observée dans le monde considéré, que ce dernier soit réaliste ou non.

Avec l'avènement des systèmes informatisés, des réseaux et l'augmentation des capacités de traitement et de stockage des ordinateurs, la quantité de données disponibles est aujourd'hui considérable et augmente plus rapidement chaque jour. À titre d'illustration, la quantité de données générées en 2013 était de 4.4 Zeta octets (10^{21} octets) selon l'institut IDC[1]. Le nombre d'objets connectés croissant également exponentiellement, l'estimation de la quantité de données générées en 2020 avec l'avènement de l'internet des objets serait de 44 Zeta octets, soit près de 10 fois plus que 7 ans auparavant[2].

Fouiller ces données est devenu une activité critique aux enjeux majeurs, tant sur le plan sociétal qu'économique, et le haut volume des données a deux conséquences principales. D'une part, elle rend indispensable l'utilisation de techniques automatisées pour y arriver, car l'information de valeur est souvent cachée à l'intérieur de cette masse d'information. D'autre part, elle rend pertinente l'exploitation d'estimateurs statistiques grâce à la grande redondance des informations. Aussi, l'utilisation de méthodes d'apprentissage automatique est de plus en plus importante et fait l'objet de nombreux travaux de recherche.

L'apprentissage automatique [15] est un domaine de recherche visant à définir des méthodes efficaces permettant à une machine de simuler des mécanismes d'apprentissage afin d'être en mesure d'adapter son comportement à des situations différentes, sans programmation explicite de toutes les actions à entreprendre en fonction des cas rencontrés. Le produit de tels algorithmes est généralement un ou plusieurs modèles explicatifs des données, appelés *connaissances*, sous la forme de fonctions ou de distributions de probabilités sur le domaine de définition des individus considérés. L'induction, l'apprentissage visant à généraliser à partir de cas particuliers, est de loin le type d'apprentissage le plus couramment étudié et a conduit à une littérature foisonnante depuis les dernières décennies. Deux types d'apprentissages inductifs peuvent être identifiés en fonction de l'usage que nous souhaitons faire de tels algorithmes : l'apprentissage génératif a pour objectif de découvrir des connaissances générales dans le domaine de définition des données ; l'apprentissage discriminatif a pour but d'optimiser les résultats de prédictions d'un modèle pour une utilisation, ou requête, précise.

Même si de nombreux algorithmes d'apprentissage automatique produisent des

[1] http://www.idc.fr/
[2] http://www.emc.com/collateral/analyst-reports/idc-digital-universe-2014.pdf

connaissances déterministes, comme en discrimination et en régression, les connaissances à découvrir sont souvent incertaines. Les modèles graphiques probabilistes [105] sont des modèles inductifs permettant de modéliser, apprendre et raisonner dans des contextes génératifs et incertains. De nombreux formalismes ont été proposés dans cette famille tels que les réseaux bayésiens [152], les réseaux de Markov [152] et les réseaux de dépendances [82]. Leur objectif est toujours le même : apprendre une distribution de probabilités génératrice des individus d'un jeu de données tabulaire, sous hypothèse d'indépendance et de distribution identique des différents individus du jeu de données (i.i.d.).

Toutefois, cette hypothèse i.i.d. des individus d'un jeu de données tabulaire est souvent irréaliste. En effet, selon cette hypothèse, aucun individu ne peut avoir d'influence sur les autres. Or, les individus ne sont généralement pas de simples enregistrements de données juxtaposés, mais interagissent les uns avec les autres, générant de nombreuses données d'interactions, associatives, entre eux. Considérer abusivement l'hypothèse i.i.d. contredit ainsi les phénomènes dits d'*autocorrélation* souvent observés dans le monde réel, p.ex. le fait que les goûts culturels d'une personne dépendent de ceux partagés par ses amis. Les connaissances obtenues sur un individu peuvent alors donner de l'information sur les autres et les individus ne peuvent donc pas être considérés comme i.i.d. De nombreux travaux dans la littérature [137, 125] ont en outre montré l'importance de relâcher cette hypothèse même dans un contexte plus vraisemblablement i.i.d pour améliorer les performances prédictives au sein d'un même individu.

Il est important de noter que l'hypothèse i.i.d. est *de facto* très répandue dans les algorithmes d'apprentissage automatique, bien au-delà des modèles graphiques probabilistes.

L'apprentissage relationnel statistique [70] est le domaine de l'apprentissage automatique lié à la représentation de connaissances, le raisonnement et l'apprentissage dans des contextes de jeux de données *multi relationnels* à connaissances incertaines. Les jeux de données considérés sont constitués de multiples objets appelés *entités* de différents types liés les uns aux autres par différents types d'*associations*. L'apprentissage et le raisonnement dans les jeux de données multi relationnels requièrent l'utilisation d'algorithmes spécifiques, prenant en compte la nature massivement auto corrélée de telles données, rendant alors l'hypothèse i.i.d., faite par de nombreux algorithmes d'apprentissage automatique, inadaptée. De nombreuses applications de l'apprentissage relationnel statistique ont été étudiées, dont le classement relationnel [107], le partitionnement relationnel [123], la discrimination collective [71], la prédiction de liens [132] et la résolution d'entités [13].

Les modèles graphiques probabilistes de second ordre [100] sont des approches adaptées pour la représentation de connaissance dans le contexte relationnel incertain. Ils consistent en une version factorisée des modèles graphiques probabilistes énoncés précédemment en ce sens qu'ils représentent à eux seuls une potentielle infinité de distributions de probabilités. Ceux-ci peuvent toutefois être instanciés en l'une de ces distributions, à partir d'une configuration de données particulière. Ainsi, un modèle factorisé et un jeu de données permettent la génération déterministe d'une loi jointe de distributions pour les individus de ce jeu de données. Comme pour leurs pendants non relationnels, des algorithmes ont été développés pour apprendre et raisonner dans les modèles factorisés.

De nombreux modèles graphiques probabilistes existent dans un contexte tabulaire et de nombreuses versions factorisées de ces derniers ont en toute logique été proposées dans la littérature, telles que les modèles relationnels probabilistes [62], les réseaux logiques de Markov [160], et les réseaux de dépendance relationnels [139]. Certains sont des versions factorisées de modèles dirigés, et d'autres de modèles non dirigés. Cette thèse se focalise sur la première catégorie. Les modèles dirigés ont en effet l'avantage d'être significativement plus rapides à apprendre que les modèles non dirigés, grâce à l'utilisation de scores d'évaluation localement décomposables. De plus, ils ne nécessitent pas de réaliser des normalisations coûteuses des paramètres appris à chaque itération, contrairement aux modèles non dirigés, puisque ceux-ci définissent à tout moment des distributions de probabilités conditionnelles. Toutefois, les modèles factorisés dirigés requièrent de garantir l'acyclicité de toutes leurs instanciations éventuelles, un problème difficile dans le cas général. Des solutions moins coûteuses pour ce problème impliquent d'ajouter des contraintes sur les modèles factorisés durant l'apprentissage qui fournissent des conditions suffisantes pour maintenir l'invariant, au prix d'un biais de représentation pouvant exclure des modèles plus complexes, mais valides.

Les modèles relationnels probabilistes [62] sont des modèles graphiques probabilistes de second ordre généralisant les réseaux bayésiens au contexte relationnel. Dans leur forme principale, les modèles relationnels probabilistes avec incertitude d'attributs permettent de définir des dépendances probabilistes entre les descriptions internes des individus de différents types, supposant les associations entre eux connues. Des extensions [67, 65] existent pour également considérer une incertitude d'associations : les modèles relationnels probabilistes avec incertitude de références font l'hypothèse que le domaine des types d'entités et des types d'association sont connus. L'objectif est alors de trouver la probabilité qu'une association référence une entité particulière parmi l'ensemble des possibilités ; les modèles relationnels probabilistes avec incertitude d'existence définissent plutôt une distribution de probabilités conditionnelle d'existence pour chaque association possible. Alors que la majorité des modèles graphiques probabilistes de second ordre sous contrainte d'incertitude de liens s'expriment selon un paradigme d'incertitude d'existence, le paradigme d'incertitude de références est intéressant puisqu'il requiert l'usage explicite de partitions dans le but de s'abstraire des jeux de données. Par conséquent, ce modèle soulève explicitement le problème de la découverte de bonnes fonctions de partitionnement dans le contexte d'incertitude de liens pour des jeux de données multi relationnels, et de bons algorithmes pour découvrir ces fonctions lors de l'apprentissage de la structure du modèle. Malheureusement, la littérature est peu détaillée au sujet de ces modèles et leur application est très sommaire, ne permettant pas de se faire une opinion sur les stratégies de partitionnement à adopter et sur les heuristiques d'apprentissage de structure à considérer.

L'objectif de cette thèse est d'étudier en détail cette extension des modèles relationnels probabilistes dans le but de trouver une méthode d'apprentissage conjoint des fonctions de partition et du modèle offrant de bonnes performances, et de rechercher des algorithmes de partitionnement donnant de bons résultats dans ce contexte de données. Notre travail inclut ainsi l'analyse détaillée de ces modèles, l'identification des contraintes à prendre en compte pour le partitionnement dans ce contexte, l'exploration de la littérature sur les algorithmes de partitionnement respectant ces contraintes,

l'identification éventuelle de limites intrinsèques à ces modèles, et le cas échéant la proposition d'extensions pour s'extraire de ces dernières. Nous cherchons notamment ici à savoir si partitionnement et apprentissage de structure peuvent mutuellement profiter l'un de l'autre et si leur apprentissage conjoint apporte un bénéfice sur le résultat final.

1.2 Contributions

Nous distinguons trois types de contributions dans cette thèse : des contributions théoriques, expérimentales et logicielles.

1.2.1 Contributions théoriques

Nous proposons dans cette thèse plusieurs contributions théoriques. La première prend la forme d'une explication détaillée des modèles d'intérêt, détaillant des points laissés même obscurs dans la littérature. Cette contribution comprend également certaines propositions comme autant de choix qui ne sont pas faits dans les articles originaux, p.ex. la définition d'un a priori sur les fonctions de partition pour l'évaluation des modèles, et la description de règles de calculs des distributions de probabilités dans les instanciations de ceux-ci. En outre, cette contribution comprend l'identification de limites pour ces modèles.

Notre seconde contribution théorique correspond aux deux extensions que nous proposons pour corriger les limites identifiées préalablement et aux stratégies de partitionnement que nous proposons, allant au-delà de ce qu'il est possible de faire dans les modèles historiques. Nous nous focalisons notamment sur l'utilisation de fonctions de partitionnement relationnel, permettant de grouper conjointement les ensembles d'entités impliqués dans une même association, permettant ainsi de profiter de l'information topologique des associations elles-mêmes, non exploitée sinon. Nous proposons également un algorithme d'apprentissage de structure glouton pour l'apprentissage de nos extensions. Cette contribution a mené à deux publications : la première a été présentée lors de la conférence FLAIRS 27 en 2014 [41] ; la seconde a été présentée lors de la conférence IJCNN 2015 [42].

Enfin, pour faire suite à notre recherche d'algorithmes de partitionnement relationnel et après avoir considéré les avantages et inconvénients de plusieurs méthodes, nous avons découvert une équivalence entre deux familles de méthodes, à savoir d'une part les méthodes de factorisation non négatives régularisées par des matrices laplaciennes, et d'autre part les méthodes de coupes minimales dans des graphes augmentés d'arcs de similarités. Cette équivalence permet d'envisager un partitionnement relationnel présentant la précision de calcul des méthodes de la première famille, avec les temps de calcul rapides de certaines méthodes de la seconde. Cette contribution a également conduit à la publication d'un article présenté lors de la conférence ESANN 2015 [40].

1.2.2 Contributions expérimentales

Nous apportons dans cette thèse des validations expérimentales pour les contributions théoriques énoncées précédemment. Ces contributions expérimentales permettent ainsi de confirmer l'intérêt de réaliser du partitionnement relationnel pour apprendre des modèles plus fidèles aux données, tant sur le plan de l'apprentissage de paramètres que de l'apprentissage de structure. De plus, les différents algorithmes de partitionnement que

nous utilisons au fur et à mesure des expériences, après avoir constaté les limites des résultats précédemment obtenus, nous permettent de comparer leurs comportements et de mesurer leur adéquation dans le contexte d'un apprentissage de modèles relationnels probabilistes avec incertitude de références.

1.2.3 Contributions logicielles

Les contributions de cette thèse comprennent enfin des valorisations logicielles importantes. Aucun outil n'existant pour l'apprentissage de structure des modèles relationnels probabilistes et leurs extensions, nous avons dû construire un cadre logiciel permettant de définir ces modèles, de raisonner avec eux, et de les apprendre à partir de données. Ce travail a mené à la réalisation d'une bibliothèque complète, appelée *PILGRIM Relational*, où nous proposons la majorité des fonctionnalités de la littérature sur les modèles relationnels probabilistes, ainsi que l'ensemble des fonctionnalités provenant de nos contributions théoriques et expérimentales. La réalisation de ce projet s'est faite en collaboration avec d'autres membres de l'équipe de recherche et nous parlerons essentiellement dans ce manuscrit de notre propre apport, comprenant la définition de l'architecture globale du projet, de nombreuses implémentations des concepts de base, ainsi que l'implémentation des extensions avec incertitude de référence et tous les concepts associés.

1.3 Organisation du manuscrit

Ce manuscrit est organisé de la façon suivante :

Le chapitre 2 présente les concepts généraux qui sont nécessaires à la compréhension de cette thèse. Nous commençons par décrire les concepts liés à la définition de domaines de données mono relationnels, ou tabulaires, c.-à-d. ne comportant qu'une seule relation. Nous définissons ensuite les concepts de dépendance fonctionnelle et probabiliste avant d'aborder l'hypothèse i.i.d. faite dans les algorithmes d'apprentissage non relationnels. Nous définissons ensuite les réseaux bayésiens, le raisonnement à partir de ces modèles ainsi que l'apprentissage de ceux-ci à partir de données. Puis, nous abordons les problèmes que pose la contrainte i.i.d. et motivons le besoin d'aller vers des solutions multi relationnelles. Nous définissons alors les concepts permettant de décrire des jeux de données relationnels et finissons par définir un premier modèle de réseaux bayésiens étendus, les *réseaux bayésiens orientés objet*, mais donnons les limites de ces modèles justifiant le besoin des modèles relationnels probabilistes, que nous définissons également ainsi que les principes pour y raisonner et en réaliser l'apprentissage à partir de données.

Le chapitre 3 présente en détail les modèles relationnels probabilistes dans le contexte d'incertitude de références. Ce chapitre donne des définitions précises des concepts impliqués spécifiques à cette extension et explicite même des points non détaillés dans la littérature par des définitions supplémentaires ainsi que divers exemples. Les notions de variables sélecteur et leur lien avec les fonctions de partition sont des points très importants de ce chapitre et font l'objet d'une attention particulière. Comme pour leur version de base, le raisonnement et l'apprentissage à partir de données pour cette extension sont également détaillés et nous finissons par présenter les différentes limites de ces modèles et de leur apprentissage, tant sur le plan des contraintes inhérentes aux fonctions de partition, où nous regrettons l'impossibilité d'utiliser des méthodes de co

partitionnement, que celles induites par le biais de représentation des structures.

Le chapitre 4 s'appuie sur les limites énoncées au chapitre précédent relatives aux fonctions de partition et notamment à l'absence de support des méthodes de co-partitionnement. Nous y présentons ainsi diverses méthodes faisant partie de la famille que nous appelons *partitionnement relationnel* permettant de grouper simultanément les différents ensembles d'entités impliqués dans un type d'association. Trois types d'algorithmes nous intéressent particulièrement dans ce chapitre, à savoir les méthodes de factorisation non négatives de matrices, les méthodes de coupe de graphe, ainsi que les méthodes de détection de communautés basées sur l'optimisation de critères de modularité. Le chapitre se termine par notre contribution relative à l'équivalence des méthodes de factorisation de matrices binaires avec régularisation laplacienne et des méthodes de coupe minimale dans un graphe augmenté d'informations de similarités.

Le chapitre 5 est le premier à proposer un nouveau type de modèle relationnel probabiliste, visant à effacer les limites des modèles du chapitre 3 relatives aux contraintes sur les fonctions de partition. Ce modèle nous sert principalement à confirmer l'hypothèse que nous faisons au sujet de l'importance d'utiliser des méthodes de partitionnement relationnel pour l'apprentissage de modèles avec incertitude de références. Après avoir décrit cette nouvelle extension, nous réalisons des expériences permettant de confirmer que l'apprentissage de modèles avec des méthodes de partitionnement relationnel obtient de meilleurs résultats que l'apprentissage de modèles selon l'approche historique décrite dans la littérature, utilisant le produit cartésien d'un ensemble d'attributs pris dans le type d'entité concerné pour chaque fonction de partitions.

Le chapitre 6 va plus loin et contient une nouvelle proposition d'extension plus complète des modèles du chapitre 3 visant à corriger l'ensemble des problèmes énoncés dans ce chapitre. Cette nouvelle extension, appelée *modèles relationnels probabilistes avec incertitude de clusters*, est définie de façon précise et nous abordons une nouvelle fois les principes de raisonnement et d'apprentissage dans ces modèles. Nous confirmons l'intérêt de ce modèle par des expérimentations, à la fois sur des données synthétiques, et sur des données réelles.

Pour finir, le chapitre 7 présente les contributions logicielles faites durant cette thèse dans le but de valoriser le travail de compréhension des modèles existants avant nous ainsi que nos contributions, et pour être en mesure de réaliser les diverses expériences des chapitres précédents. Nous énumérons l'ensemble des solutions existantes de programmation probabiliste puis nous identifions leurs propriétés et exhibons l'absence d'outils permettant de réaliser de l'apprentissage de structure de modèles relationnels probabilistes, motivant la création de notre propre outil. Nous abordons ainsi ensuite le sujet principal du chapitre, à savoir le projet PILGRIM, et plus particulièrement sa brique relationnelle à laquelle nous avons contribué de façon conséquente. Nous présentons les différentes parties constituant cet outil et donnons divers points de documentation simplifiés, avec quelques diagrammes de classe de haut niveau.

2

Des réseaux bayésiens aux modèles relationnels probabilistes

Sommaire

2.1 Introduction

L'*apprentissage relationnel statistique* [70] est un domaine à la frontière de l'apprentissage symbolique et de l'apprentissage en environnement incertain. Son objectif est d'étudier les moyens permettant de modéliser des connaissances, de les apprendre et de s'en servir pour raisonner dans un contexte impliquant des données complexes, généralement en interaction les unes avec les autres, bruitées, et intrinsèquement incertaines (c.-à-d. souffrant d'une incertitude inhérente au domaine qu'elles matérialisent). Ce domaine a connu un essor considérable lors de ces deux dernières décennies et a mené à de nombreuses applications telles que le clustering multi relationnel [175, 2, 135], la classification collective [137, 69, 130], la prédiction de liens [176, 157] et la résolution d'entités [167, 13].

Les modèles graphiques probabilistes de second ordre [100] sont une approche adaptée pour la représentation de l'incertain dans un contexte relationnel. Ils proposent des versions factorisées de modèles graphiques probabilistes provenant du monde propositionnel, en ce sens qu'ils peuvent être dans un second temps instanciés en un de ces modèles graphiques probabilistes, à partir d'un jeu de données. Ainsi, les modèles graphiques probabilistes de second ordre sont des patrons pour des ensembles de lois jointes de distributions de probabilités et un modèle de second ordre mêlé à un jeu de données détermine une loi unique dans cet ensemble. Comme pour leurs versions propositionnelles, des algorithmes ont été développés pour apprendre et raisonner avec ces modèles.

De nombreux modèles graphiques de second ordre existent dans la littérature. Cette thèse se focalise plus particulièrement sur les *modèles relationnels probabilistes*, ou *réseaux bayésiens relationnels* [62, 67, 66] qui, comme le dernier nom l'indique, sont une extension relationnelle des réseaux bayésiens. Aussi, ce chapitre est essentiellement centré sur ces modèles, et une section est dédiée en fin de chapitre aux alternatives du domaine.

Plus précisément, nous présentons dans ce chapitre les concepts généraux conduisant naturellement aux modèles relationnels probabilistes. Nous commençons par aborder le cas de données mono relationnelles et définissons les concepts de dépendances fonctionnelles et probabilistes, avant d'aborder le sujet des réseaux bayésiens adaptés à ce type de données. Ensuite, nous expliquons le besoin amenant au paradigme multi relationnel incertain, abordant les limites du monde propositionnel sur le plan symbolique d'une part et décrivant les phénomènes d'autocorrélation mettant à mal les hypothèses d'indépendance des distributions sur le plan statistique d'autre part. Nous continuons alors ce chapitre par la définition des concepts multi relationnels pour arriver à une première extension des réseaux bayésiens dans ce contexte, appelés *réseaux bayésiens orientés objet* [106], et discutons les limites de ces modèles amenant aux modèles relationnels probabilistes, que nous définissons finalement.

2.2 Paradigme mono relationnel

Historiquement, la majorité des algorithmes d'apprentissage s'est concentrée sur des jeux de données tabulaires, plats, où chaque individu est considéré indépendant et identiquement distribué par rapport aux autres à partir d'une distribution de données génératrice. Dans le paradigme de représentation relationnel, un jeu de données dans ce

contexte est un ensemble de tuples, chacun représentant un individu différent, décrit selon certaines contraintes. Nous définissons dans cette section les différents concepts du paradigme relationnel que nous utiliserons dans cette thèse, en nous concentrant sur le contexte d'un jeu de données constitué d'une seule relation. Nous abordons enfin l'apprentissage dans ces jeux de données, en commençant par les environnements certains pour finir sur les environnements incertains et la description des réseaux bayésiens.

Le paradigme de représentation relationnel des données a été initialement introduit par Codd [36]. Nous nous restreignons dans ce chapitre aux définitions uniquement nécessaires pour cette thèse et référons le lecteur vers l'ouvrage de référence de Date [44] pour une description plus détaillée des concepts relationnels.

2.2.1 Généralités

Nous présentons tout d'abord les différents concepts généraux relatifs à la description de données, dans le cadre de jeux de données tabulaires mono relationnels.

L'unité élémentaire d'un jeu de données est la donnée. Toute donnée est une unité d'information pouvant prendre plusieurs valeurs, selon son *domaine de définition* énumérant par intention ou extension l'ensemble des possibilités.

Définition 2.1. *Un domaine (de données) D est un ensemble fini ou infini de valeurs possibles pour une unité d'information ou élément d'un jeu de données.*

En règle générale, les individus d'un jeu de données sont décrits par plusieurs unités élémentaires d'information, sous forme de *tuples*, c.-à-d. de séquences de données prises dans leurs différents domaines de données.

Définition 2.2. *Un n-tuple $t = \langle t[i] \rangle_{1 \leq i \leq n}$ est une séquence de n éléments, chacun pouvant avoir un domaine de données $dom(t[i])$ différent.*

Par extension, nous définissons le domaine d'un tuple comme le produit cartésien des domaines de ses éléments : $dom(t) = dom(t[1]) \times \ldots \times dom(t[n])$.

Un *attribut* correspond à un domaine de définition nommé, permettant d'associer à ce dernier une sémantique intelligible.

Définition 2.3. *Un attribut A est une paire (N, D) avec N son nom, et D son domaine.*

Nous étendons la fonction dom définie préalablement pour les tuples aux attributs, c.-à-d. ici $dom(A) = D$. Par extension, nous définissons également cette fonction aux séquences d'attributs : $dom(\langle A_i \rangle_{1 \leq i \leq k}) = dom(A_1) \times \ldots \times dom(A_k)$.

Plusieurs attributs aux noms distincts peuvent être regroupés en séquence. Nous appelons généralement *en-tête* de tels regroupements, utilisés pour énumérer les informations à renseigner pour chaque individu d'un jeu de données.

Définition 2.4. *Un en-tête H est une séquence d'attributs ayant des noms différents deux à deux. Son degré est la longueur de la séquence, c.-à-d. sa cardinalité dénotée $|H|$.*

Différents individus sont regroupés dans des *relations*. Une relation est un ensemble de tuples ordonnés similairement, c.-à-d. de façon à ce que les éléments à la même position dans chacun des tuples décrivent un même concept.

Définition 2.5. *Une relation n-aire est une paire (H, T) où $H = \langle A_i \rangle_{i \leq n}$ est un en-tête de degré n et T est une collection finie de n-tuples $T = \{t_j\}_{1 \leq j \leq m}$, chaque t_j étant une fonction de H vers $dom(H)$.*

Nous notons $t[U]$ la restriction du tuple t à la séquence d'attributs $U \subseteq H$.

Finalement, une relation doit généralement respecter un ensemble de contraintes définies par un *schéma de relation* pour être valide.

Définition 2.6. *Un schéma de relation $\mathcal{R}$ est une paire $(\mathcal{H}, \mathcal{C})$ où $\mathcal{H}$ est un en-tête et $\mathcal{C}$ est un ensemble de contraintes définies sur les éléments de $\mathcal{H}$. Une contrainte est une formule logique qui peut être valide pour une relation, c.-à-d. vraie pour tout tuple de cette relation, ou non. Dans le second cas, la relation viole la contrainte.*

Une relation $R(H, T)$ peut être vue comme une instanciation d'un schéma de relation si leurs séquences d'attributs sont identiques ($H = \mathcal{H}$) et si tous ses tuples valident les contraintes $\mathcal{C}$. Dans la suite, nous définissons une fonction $attr$ retournant la séquence d'attributs d'un schéma de relation.

2.2.2 Dépendance fonctionnelle

Une contrainte très répandue dans les schémas de relation est celle de *dépendance fonctionnelle*. Cette contrainte décrit un déterminisme entre plusieurs attributs du schéma de relation, de telle sorte que la connaissance d'une partie des valeurs d'attributs pour n'importe quel tuple détermine de façon certaine et non ambiguë la valeur d'autres attributs pour ces tuples.

Définition 2.7. *Étant donné un schéma de relation $\mathcal{R} = (\mathcal{H}, \mathcal{C})$ et deux sous-ensembles de son en-tête $\mathcal{H}_x, \mathcal{H}_y \subset \mathcal{H}$, une dépendance fonctionnelle de $\mathcal{H}_x$ vers $\mathcal{H}_y$ est une contrainte du schéma qui certifie que pour toute relation R instanciant $\mathcal{R}$, et toute paire de tuples (t_1, t_2) de R, nous avons $t_1[H_x] = t_2[H_x] \rightarrow t_1[H_y] = t_2[H_y]$.*

Le concept de dépendance fonctionnelle est au cœur de nombreux algorithmes d'apprentissage automatique et de la connaissance recherchée par ces algorithmes.

Nous pouvons également définir le concept plus général de *dépendance fonctionnelle conditionnelle* [17] apportant un contexte à une dépendance fonctionnelle, qui n'est alors vraie que dans certaines conditions décrites par des valeurs spécifiques d'attributs.

Définition 2.8. *[17] Étant donné un schéma de relation $\mathcal{R} = (\mathcal{H}, \mathcal{C})$ et deux sous-ensembles de son en-tête $\mathcal{H}_x, \mathcal{H}_y \subset \mathcal{H}$, et un tuple patron t_p, une dépendance fonctionnelle conditionnelle de $\mathcal{H}_x$ vers $\mathcal{H}_y$ sachant t_p, est une contrainte du schéma qui certifie que pour chaque relation R instanciant $\mathcal{R}$, et toute paire de tuples (t_1, t_2) de R, nous avons $t_1 \leq t_p, t_2 \leq t_p, t_1[H_x] = t_2[H_x] \rightarrow t_1[H_y] = t_2[H_y]$. Nous avons $t \leq t_p$ si pour chaque $A \in \mathcal{H}$ nous avons soit $t[A] = t_p[A]$ ou $t_p[A] = \_$ (A n'est alors pas restreint pour cette dépendance).*

2.2.3 Dépendance probabiliste

Une dépendance probabiliste relâche le concept de dépendance fonctionnelle dans le cas d'un environnement incertain. Une dépendance probabiliste entre deux ensembles d'attributs indique l'existence d'une corrélation statistique, non forcément déterministe, entre les ensembles de variables aléatoires qui en sont issus, c.-à-d. présentant les mêmes domaines de définition. Cela signifie que les valeurs prises par les deux sous-ensembles d'attributs ne sont pas indépendantes l'une de l'autre.

Définition 2.9. *Considérant un schéma de relation $\mathcal{R} = (\mathcal{H}, \mathcal{C})$, et l'ensemble de variables aléatoires $H = \{H_i | \mathcal{H}_i \in \mathcal{H} \wedge dom(H_i) = dom(\mathcal{H}_i)\}$ issu de $\mathcal{H}$, on dit qu'il existe une dépendance probabiliste entre deux sous-ensembles d'attributs disjoints $\mathcal{H}_x \subseteq \mathcal{H}$ et $\mathcal{H}_y \subseteq \mathcal{H}$ ssi nous avons pour leurs sous-ensembles de variables aléatoires associées $H_x \subseteq H$ et $H_y \subseteq H$: $P(H_x) \times P(H_y) \neq P(H_x, H_y)$.*

Pour des raisons de simplicité, nous confondrons par la suite un ensemble d'attributs et l'ensemble de variables aléatoires qui en est issu. Nous définirons ainsi directement des distributions de probabilités sur des ensembles d'attributs. À titre d'exemple, nous pourrons parler de loi jointe de distribution $P(\mathcal{H})$ pour l'en-tête du schéma de relation défini plus haut.

La manipulation de distributions de probabilités permet d'effectuer des opérations formellement définies et éprouvées depuis de nombreuses années, telles que la composition ou le conditionnement.

Considérons à titre d'exemple deux règles d'implication $A = a \rightarrow B = b$ et $B = b \rightarrow C = c$ ayant chacune une *confiance* (c.-à-d. une fréquence de tuples vérifiant la règle parmi les tuples concernés par la prémisse) de 0.7 dans le jeu de données. Doit-on considérer $A = a \rightarrow C = c$ comme une règle valide ? Dans le cas des algorithmes apprenant des règles strictes, il est possible de déduire qu'un tuple t ayant une valeur $t[A] = a$ a une valeur $t[C] = c$, alors que si l'on assimile la confiance à une mesure de probabilité, $A = a \rightarrow C = c$ aura elle-même une mesure de probabilité de 0.49 seulement. Les dépendances probabilistes permettent donc de quantifier de façon plus précise le degré de croyance apporté aux prédictions.

2.2.4 Hypothèse d'indépendance des tuples (i.i.d.)

Une hypothèse importante faite par la majorité des algorithmes d'apprentissage automatique fonctionnant sur des jeux de données mono relationnels est l'idée que leurs individus sont indépendamment et identiquement distribués (i.i.d.). Ces algorithmes admettent l'existence d'une distribution de probabilités génératrice de chaque individu, considérant ses seules valeurs d'attributs, qu'il faut essayer d'approximer, généralement par des mécanismes inductifs. Comme nous le verrons par la suite, cette hypothèse est très forte et non nécessairement vérifiée dans les jeux de données.

2.3 Réseaux bayésiens

Un réseau bayésien (RB) [152] est un modèle graphique probabiliste représentant la loi jointe de distributions d'un ensemble de variables aléatoires H issu d'un ensemble d'attributs $\mathcal{H}$.

Définition 2.10. *[152] Un réseau bayésien $\mathcal{M} = (\mathcal{S}, \Theta)$ est composé d'un graphe dirigé sans circuit (DAG) $\mathcal{S} = (V, pa)$ et d'un ensemble de paramètres Θ. Les nœuds du graphe sont les variables aléatoires $V = \{v_1, v_2, \ldots, v_n\}$ du domaine et la fonction pa associe à chaque variable $v \in V$ ses parents $pa(v) \subset V$ de façon à décrire un DAG. Cette structure permet de déterminer les dépendances et indépendances conditionnelles entre toutes les variables aléatoires considérées. Chaque nœud du graphe est finalement associé à un paramètre $\theta_i \in \Theta$ déterminant sa distribution de probabilités conditionnelle (CPD) sachant ses parents dans le graphe. Un réseau bayésien représente une factorisation de la loi jointe de distributions $P(V)$. En effet, nous pouvons écrire :*

$$P(V) = \prod_{i=1}^{n} P(v_i|pa(v_i)) \ .$$

Un réseau bayésien peut être défini à la main, p.ex. par l'expert d'un domaine, dans le but de faire des prédictions, mais il peut également être appris à partir d'une relation $R = (H, T)$. Dans ce second cas, l'ensemble des variables aléatoires considérées correspond à celui issu de l'en-tête H ou un de ses sous-ensembles. Comme énoncé précédemment pour des raisons de simplicité, nous confondrons par la suite un attribut et la variable aléatoire qui en est issue. Aussi, nous pourrons définir le graphe d'un réseau bayésien $\mathcal{S} = (H, pa)$ et définir des distributions $P(A_i|pa(A_i))$ pour $A_i \in H$.

Il est important de noter que les réseaux bayésiens supposent que les individus sont i.i.d. car les dépendances probabilistes entre les différentes variables sont contextuelles aux attributs d'un même individu et il est donc impossible d'inférer sur les valeurs d'attributs d'un individu à partir des informations disponibles pour un autre. La vraisemblance $P(R|\Theta, \mathcal{S})$ d'une relation $R = (H, T)$ pour un réseau bayésien $\mathcal{M} = (\mathcal{S}, \Theta)$ s'obtient ainsi en réalisant le produit des vraisemblances de chaque individu :

$$P(R|\Theta, \mathcal{S}) = \prod_{t \in T} \prod_{A_i \in H} P\left(t\left[A_i\right] \mid t\left[pa(A_i)\right] \right) . \tag{2.1}$$

Un exemple de structure de réseau bayésien dans le contexte de la recommandation de films est donné dans la Figure 2.1. Cette structure détermine les dépendances et indépendances probabilistes entre les différents goûts d'une personne donnée en terme de films et de réalisateurs, dans le but de recommander des films qui n'ont pas encore été vus, mais qui ont une grande probabilité d'être appréciés par cette personne.

2.3.1 Inférence dans les réseaux bayésiens

Réaliser de l'inférence dans un réseau bayésien revient à essayer de calculer la distribution de probabilités d'un sous-ensemble de ses variables H_s sachant que certaines valeurs sont connues pour un autre sous-ensemble H_e de ses variables, avec $H_e \cap H_s = \emptyset$. Formellement, cela revient à calculer $P(H_s|H_e = \mathbf{u_e}, \mathcal{S}, \Theta_{\mathcal{S}})$, où $\mathbf{u_e}$ est le vecteur de valeurs pour H_e, appelé *évidence*.

De nombreux algorithmes ont été proposés au fil des années pour réaliser de l'inférence probabiliste dans les RBs en propageant les informations de l'évidence vers les autres variables du modèle jusqu'à atteindre les variables d'intérêt H_s. Afin d'être efficace, cette propagation se fait en exploitant la structure du RB considéré, permettant

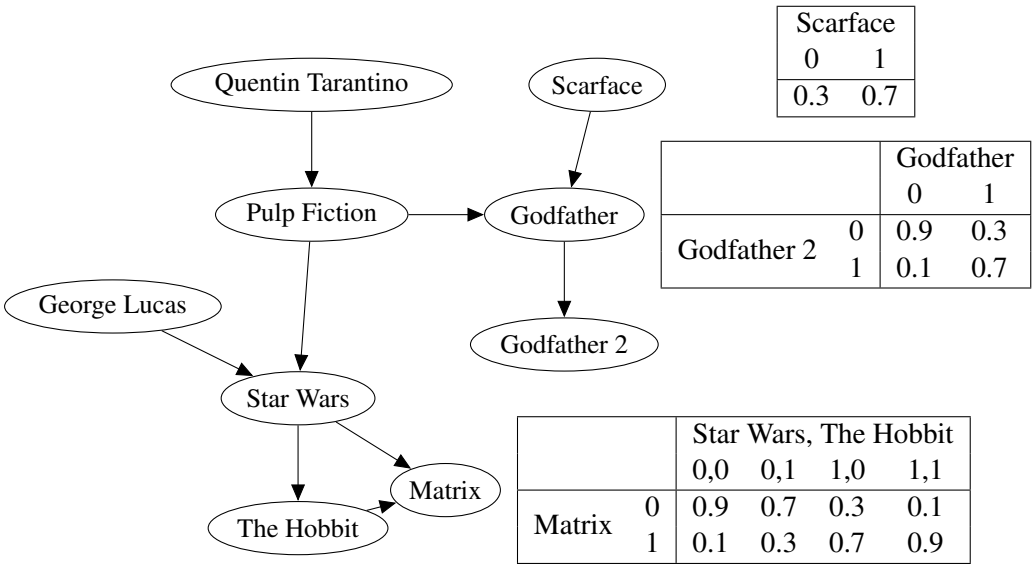


FIGURE 2.1 – Exemple de structure de réseau bayésien dans un contexte de recommandation de films. Les variables sont ici booléennes et expriment le goût d'une personne particulière pour un film ou un réalisateur. Chaque variable est associée à une distribution de probabilités définie sachant les valeurs de ses variables parentes dans le graphe de structure. Plusieurs distributions sont proposées à titre d'exemple sous la forme de tables de probabilités conditionnelles.

généralement de simplifier le problème posé grâce aux diverses indépendances conditionnelles qu'il décrit entre ses variables. Un premier algorithme limité aux structures arborescentes, appelé *Message Passing* (MP), a été proposé par Pearl [153] et Kim et Pearl [99]. Celui-ci a ensuite été étendu pour des structures moins spécifiques, p. ex. par Lauritzen et Spiegelhalter [113]. Une autre méthode d'inférence exacte célèbre est basée sur l'élimination successive de variables dans les calculs de distributions [187].

Les méthodes énoncées jusqu'ici permettent de réaliser de l'inférence exacte. Toutefois, l'inférence dans les réseaux bayésiens dans le cas général est connue pour être NP-difficile [37]. C'est pourquoi, des méthodes approchées d'inférence dans les RBs ont été développées pour gérer les cas où le nombre de variables est important ou lorsque la dimension du modèle est très grande, tel que *loopy belief propagation* [89] étendant l'algorithme MP au cas de réseaux bayésiens avec circuit. Pour de plus amples informations à ce sujet, nous redirigeons le lecteur vers des ouvrages traitant du sujet de façon détaillée [112, 43].

2.3.2 Apprentissage de réseaux bayésiens

L'apprentissage de réseaux bayésiens à partir de données se décompose en deux sous-tâches distinctes : l'estimation des paramètres étant donné une structure de graphe fixée, et l'apprentissage de la structure elle-même.

i Estimation des paramètres

Étant donné une relation $R = (H, T)$ et une structure fixée $\mathcal{S} = (H, pa)$, l'évaluation des paramètres peut être réalisée en considérant les diverses fréquences dans la

relation. Soit N_{ijk}, le nombre de tuples de la relation dont l'attribut $A_i \in H$ est à sa k-ième valeur et dont les attributs $pa(A_i)$ sont à leur j-ième valeur, et $N_{ij} = \sum_k N_{ijk}$, il est possible d'utiliser ces deux fréquences pour estimer la distribution $P(\theta_i|R) = P(A_i|pa(A_i), R)$. La manière la plus simple de réaliser cette estimation est selon le maximum de vraisemblance :

$$\hat{P}(A_i = a_k|pa(A_i) = \mathbf{u_j}, R) = \frac{N_{ijk}}{N_{ij}}$$

Toutefois, cette estimation est très sensible au bruit dans les données, et donc ne garantit pas de bonnes prédictions pour de nouveaux tuples. Il est aisé de remarquer notamment que dans le cas où $N_{ijk} = 0$, l'évènement aléatoire $A_i = k|pa(A_i) = j$ sera considéré comme impossible. Une solution plus robuste au bruit est de réaliser une estimation bayésienne des distributions. En effet, considérant maintenant une distribution a priori sur tous les paramètres possibles pour θ_i, c.-à-d. $P(\theta_i)$, décrivant une croyance sur les valeurs plausibles de la distribution, nous pouvons écrire grâce au théorème de Bayes :

$$P(\theta_i|R) \propto P(R|\theta_i)P(\theta_i)$$

où $P(R|\theta_i)$ est la vraisemblance probabiliste du jeu de données sachant un paramètre déterminé. Si les distributions à apprendre sont des tables de probabilités conditionnelles (CPT), un choix possible pour $P(\theta_i)$ est une distribution de Dirichlet avec des hyper paramètres α_{ijk} pour chaque N_{ijk} précédemment défini. Dans ce cas, l'a posteriori $P(\theta_i|R)$ est aussi une distribution de Dirichlet d'hyper paramètres $\alpha_{ijk} + N_{ijk}$ et les paramètres sont alors estimés de la façon suivante :

$$\hat{P}(A_i = a_k|pa(A_i) = \mathbf{u_j}, R) = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}}.$$

Les hyper paramètres de l'a priori peuvent être assimilés à des fréquences concernant les individus d'un échantillon imaginaire. Plus les valeurs des α_{ijk} augmentent, et plus les distributions apprises seront proches de l'a priori. Au contraire, plus le nombre d'individus réels augmente, plus les distributions apprises se rapprocheront du maximum de vraisemblance. Ainsi, l'approche bayésienne permet d'éviter un sur apprentissage lorsque le nombre d'individus est faible, tout en faisant de plus en plus confiance aux statistiques obtenues sur le jeu de données lorsque le nombre d'individus augmente.

ii Apprentissage de structure

Plusieurs approches d'apprentissage de structure de réseaux bayésiens peuvent être identifiées.

Les approches par contraintes [171, 180] ont pour but de reconstruire un réseau bayésien à partir d'un ensemble de résultats de tests d'indépendance des variables aléatoires. La première phase consiste à identifier les indépendances (conditionnelles) entre variables, grâce à des tests statistiques comme le χ^2 ou l'information mutuelle. Des biais de recherche sont généralement introduits afin d'éviter des problèmes liés à un trop grand nombre de tests à effectuer, par exemple le nombre maximal de parents que peut avoir un nœud. Ensuite, l'ensemble des indépendances découvertes est utilisé pour construire une structure, si possible de dimension minimale, c.-à-d. avec le moins de degrés de liberté possible.

L'apprentissage de structure d'un réseau bayésien peut également être vu comme un problème d'optimisation où le but est de trouver le modèle maximisant une fonction objectif [32, 33, 81, 34]. Comme l'espace des modèles possibles pour un en-tête donné est très grand, un calcul exhaustif des scores de tous les modèles est intraitable en pratique. Aussi, les algorithmes d'apprentissage selon cette approche utilisent généralement des méthodes approchées, à base de méta heuristiques, dont la plus populaire est sans doute la méthode de recherche gloutonne. Cette méta heuristique fonctionne en trois étapes : 1) à partir d'une hypothèse (ici une structure de RB) courante, obtenue suite aux itérations précédentes, définir un ensemble d'hypothèses voisines ; 2) calculer pour chacune de ces hypothèses son score vis-à-vis de la fonction objectif ; 3) choisir l'hypothèse voisine maximisant le gain de score par rapport à l'hypothèse courante (on s'arrête si aucun voisin n'améliore le score) et retourner en 1.

iii Exploration de l'espace de structure

L'algorithme de recherche gloutonne tente d'améliorer un modèle $\mathcal{M}$ qui a été déterminé comme le meilleur candidat jusqu'à l'itération actuelle par la maximisation d'une fonction objectif parmi d'autres candidats voisins de ce modèle. À chaque itération, le voisinage d'un modèle est obtenu par l'application à ce modèle de différents opérateurs de voisinage. Ces opérateurs sont généralement au nombre de trois : ajout ou suppression d'un parent à une variable, et inversion d'une relation parent-enfant. Étant donnés deux modèles, ceux-ci sont dits voisins dans le contexte de la recherche gloutonne ssi l'exécution d'un seul des opérateurs susmentionnés permet de passer de l'un à l'autre.

iv Évaluation des candidats

En apprentissage de structure, nous sommes intéressés par la découverte d'un modèle le plus fidèle possible à la distribution de données réelle. Cette distribution étant inconnue, il faut trouver une façon de comparer les modèles afin de savoir lequel est le meilleur pour cette tâche. Formellement, nous voulons trouver un modèle $\mathcal{M} = (\mathcal{S}, \Theta)$ maximisant $P(\mathcal{S}|R) \propto P(R|\mathcal{S})P(\mathcal{S})$. Le premier terme est la vraisemblance marginale de la loi jointe de distribution sur l'ensemble des paramètres de la structure correspondante :

$$P(R|\mathcal{S}) = \int_{\Theta} P(R|\Theta, \mathcal{S})P(\Theta|\mathcal{S})\, d\Theta$$

où $P(R|\Theta, \mathcal{S})$ est défini dans (2.1), et $P(\Theta|\mathcal{S})$ est l'a priori sur les paramètres. Afin d'avoir un score décomposable, il faut que chaque a priori satisfasse les propriétés d'*indépendance globale des paramètres* et de *modularité des paramètres*. Ces propriétés sont par exemple assurées par les a priori de Dirichlet. Dans ce cas, $P(R|\mathcal{S})$ a une forme close :

$$P_{Dir}(R|\mathcal{S} : \boldsymbol{\alpha}) = \prod_{A_i \in H} \prod_{\mathbf{u}_j \in dom(pa(A_i))} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{a_k \in dom(A_i)} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \quad (2.2)$$

où $\boldsymbol{\alpha}$ est le vecteur des hyper paramètres α_{ijk}.

Le score utilisant (2.2) est appelé Bayesian Dirichlet (BD) [38] :

$$score_{BD}(\mathcal{S} : R, \boldsymbol{\alpha}) = log\,(\ P(\mathcal{S})P_{Dir}(R|\mathcal{S} : \boldsymbol{\alpha})\ )\,. \tag{2.3}$$

Buntime [23] a proposé une variante du score BD, appelée *Equivalent Uniform BD* (BDeu) dans le cas où les coefficients de Dirichlet sont obtenus par la formule suivante :

$$\alpha_{ijk} = \frac{\alpha}{|dom(A_i)| \times |dom(pa(A_i))|},$$

où α représente un nombre global d'individus synthétiques équivalents.

Heckerman et coll. [83] ont ensuite proposé une variante du score BDeu, appelé *Equivalent BD* (BDe) dans le cas où les coefficients de Dirichlet s'obtiennent de la façon suivante :

$$\alpha_{ijk} = \alpha \times P(A_i = a_k, pa(A_i) = \mathbf{u_j}|\mathcal{S}_c),$$

où $\mathcal{S}_c$ est la structure non dirigée complète, c.-à-d. où toutes les paires de nœuds sont connectées.

Une autre approximation de (2.3) peut être faite en considérant la limite de (2.2) lorsque le nombre d'individus tend vers l'infini. En effet, dans ce cas, nous pouvons écrire :

$$logP(R|\mathcal{S}) = logP(R|\hat{\theta}_{\mathcal{S}}) - \frac{log|T|}{2}dim(\mathcal{S}) + c$$

où $dim(\mathcal{S})$ est la dimension de la structure, c.-à-d. le nombre de paramètres indépendants dans $\mathcal{S}$, $P(R|\hat{\theta}_{\mathcal{S}})$ est le maximum de vraisemblance du modèle sachant les données, $|T|$ est le nombre d'individus et c est majoré par 1.

Cette équivalence a mené à la définition du score *Bayesian Information Criterion* [163] (BIC) :

$$score_{BIC}(\mathcal{S} : R) = logP(R|\hat{\theta}_{\mathcal{S}}) - \frac{log|T|}{2}dim(\mathcal{S}).$$

Le caractère localement décomposable des scores décrits dans cette section permet de calculer de façon très efficace les scores d'un modèle sachant le score d'un de ses voisins, en calculant uniquement l'impact sur le score de l'application de l'opérateur de voisinage.

2.4 Paradigme multi relationnel

Le paradigme mono relationnel est très répandu dans les communautés d'apprentissage automatique et de fouille de données. Toutefois, les hypothèses induites ne sont pas toujours vérifiées et introduisent des biais importants de représentation. En outre, de nombreux jeux de données réels ne peuvent pas du tout être représentés par une unique relation. Cette section relâche les hypothèses précédentes et aborde le cas de jeux de données multi relationnels.

2.4.1 Limites de l'hypothèse i.i.d. et autocorrélation

L'hypothèse d'indépendance et de distribution identique des données pour chaque individu d'un jeu de données mono relationnel est souvent irréaliste. En effet, par définition, selon cette hypothèse, aucun individu ne peut avoir d'influence sur les autres. Ceci contredit les phénomènes d'*autocorrélation* souvent observés dans le monde réel. En

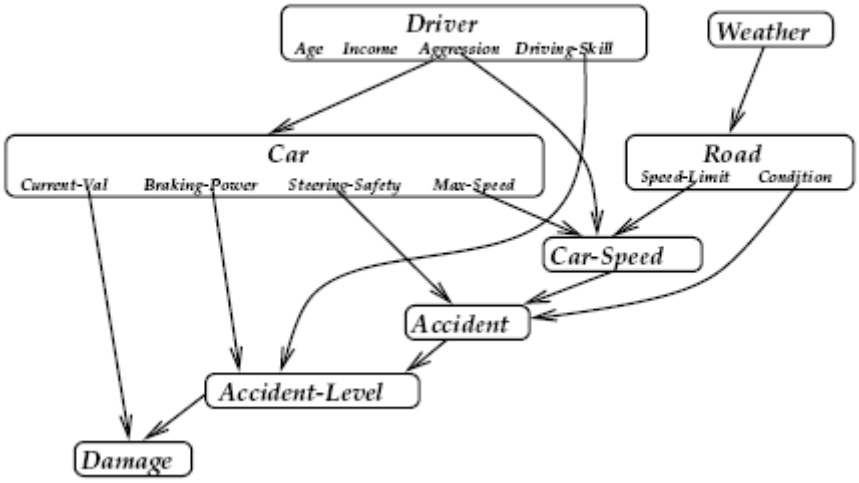


FIGURE 2.2 – Une structure de RBOO dans un contexte d'accident de voiture. *Image provenant de [106].*

effet, considérons par exemple le cas de films visionnés et appréciés par une personne. Sous hypothèse i.i.d., nous pourrions essayer de trouver une distribution représentant les goûts d'une personne pour un film en fonction de caractéristiques propres à cette personne et à ce film. Or, une personne a plus de chances d'apprécier un film si un nombre conséquent de ses pairs l'a apprécié également. De façon symétrique, une personne a plus de chances d'apprécier un film si elle a déjà apprécié d'autres films partageant les mêmes caractéristiques. Aussi, les connaissances obtenues sur un individu peuvent donner de l'information sur les autres et les individus ne peuvent donc pas être considérés comme i.i.d. D'autres exemples de jeux de données où ce phénomène joue un rôle important incluent les données génétiques, où une dépendance entre deux gênes dépend du type de chaque gêne et des autres dépendances de chacun. De nombreux travaux dans la littérature [137, 125] ont en outre montré l'importance de faire une inférence collective pour améliorer les performances individuelles par rapport au seul usage des méthodes d'inférence dans un contexte i.i.d.

2.4.2 Réseaux bayésiens orientés objet

Au-delà de l'hypothèse i.i.d., un autre désavantage des modèles mono relationnels est leur inadéquation pour modéliser des domaines larges et complexes [126]. En effet, dans de tels modèles, la structure est fixée à l'avance. Elle spécifie une configuration précise du domaine considéré. De plus, aucune partie n'est réutilisable, et il faut nécessairement recopier de façon explicite les régularités de structure ou de paramètres.

Les réseaux bayésiens orientés objet (RBOO) ont été proposés pour résoudre ces problèmes [106, 7]. Ces modèles appliquent différents principes du paradigme de représentation objet aux réseaux bayésiens, tels que l'encapsulation, la composition et l'héritage.

L'élément de base d'un RBOO est l'objet. Un objet $o \in O$, où O est l'ensemble des objets, est composé d'attributs A_o qui sont eux-mêmes des objets. Trois types d'attri-

buts existent : les entrées (I_o), sorties (O_o), et les objets encapsulés (E_o). Chaque objet est lié à une *fonction stochastique* de ses entrées vers ses sorties, utilisant des dépendances internes entre attributs faisant intervenir les objets encapsulés. Ce processus est défini de façon récursive pour chaque objet attribut d'un autre.

La réutilisation de type est possible en RBOO à travers la notion de classe. Les classes permettent de définir des types génériques, décrits à partir d'autres classes, et peuvent être instanciées par les objets. Les classes peuvent également être définies à partir d'autres classes par héritage (une relation de type "'est-un"') ou composition (une relation de type "'partie de"').

Pour être complètement défini, un RBOO doit décrire un ensemble de classes C, représentant tous les types complexes des objets du domaine considéré. De plus, un objet principal, sans attribut d'entrée, appelé *situation*, définit l'instanciation spécifique des classes dans laquelle on souhaite raisonner. Il est composé d'objets de types basiques ou complexes dans C et est lui-même associé à une fonction stochastique. Un exemple de RBOO est donné dans la figure 2.2.

Deux stratégies d'inférence peuvent être utilisées pour propager les croyances dans les modèles. La première consiste à transformer le RBOO en un RB classique. Les types complexes n'étant présents que pour structurer le modèle et n'ayant pas de sémantique propre, la transformation consiste en l'aplatissement du RBOO pour obtenir un RB $B = (G, \theta)$ où $G = (V, E)$ et V correspond à tous les attributs de types simples du RBOO, et $E = \{e_1, e_2, \ldots, e_n\}$ est l'ensemble des arcs reliant une paire de noeuds de s vers t avec $s, t \in V$ s'il existe un lien dans le RBOO de U_s à V_t où $(U, V) \in \{(O, I), (E, O), (I, E), (I, O)\}$. Les CPD sont aisément obtenues via les fonctions stochastiques des attributs de type simple du RBOO. Un RBOO complètement défini décrit une unique loi jointe de distribution sur les attributs de type simple du modèle et cette loi jointe est celle obtenue à partir de B comme pour un RB classique. Il est intéressant de noter que cette stratégie n'utilise pas les régularités de structure du RBOO considéré et les algorithmes d'inférence des RB, qu'ils soient exacts ou approchés, peuvent être utilisés.

La deuxième stratégie peut mener à des gains de performance notables puisqu'elle utilise les régularités de structure du modèle. En effet, puisque dans un RBOO, un objet communique avec les autres à travers son interface (ses entrées et sorties), nous pouvons en déduire que dans le RB obtenu par aplatissement, chaque variable d'un objet X sera conditionnellement indépendant des variables des autres objets sachant O_X et I_X. À partir de cette observation, Koller définit dans [106] une méthode permettant de transformer un RBOO en un réseau bayésien à sections multiples [185] (MSBN). Un MSBN partitionne le RB aplati obtenu à partir du RBOO en un ensemble de sous-réseaux non disjoints. Chaque intersection entre deux sous-réseaux est appelé *d-sepset* ayant la propriété de rendre localement indépendants les deux sous-réseaux concernés s'ils sont isolés du reste du RB. Dans cette méthode, un sous-réseau est créé pour chaque objet complexe. Plus précisément, le MSBN d'un RBOO contient un sous-réseau $\sum_X$ pour chaque objet X du RBOO. Le sous-réseau $\sum_X$ contient les ensembles d'attributs O_X, I_X, les variables simples de E_X et tous les attributs non encapsulés (dits *publics*) de chaque variable complexe dans X. Les d-sepsets correspondent aux interfaces entre objets. Si l'on considère finalement un arbre T défini par l'ensemble des objets du RBOO tels que Y est un parent de X si X est un attribut

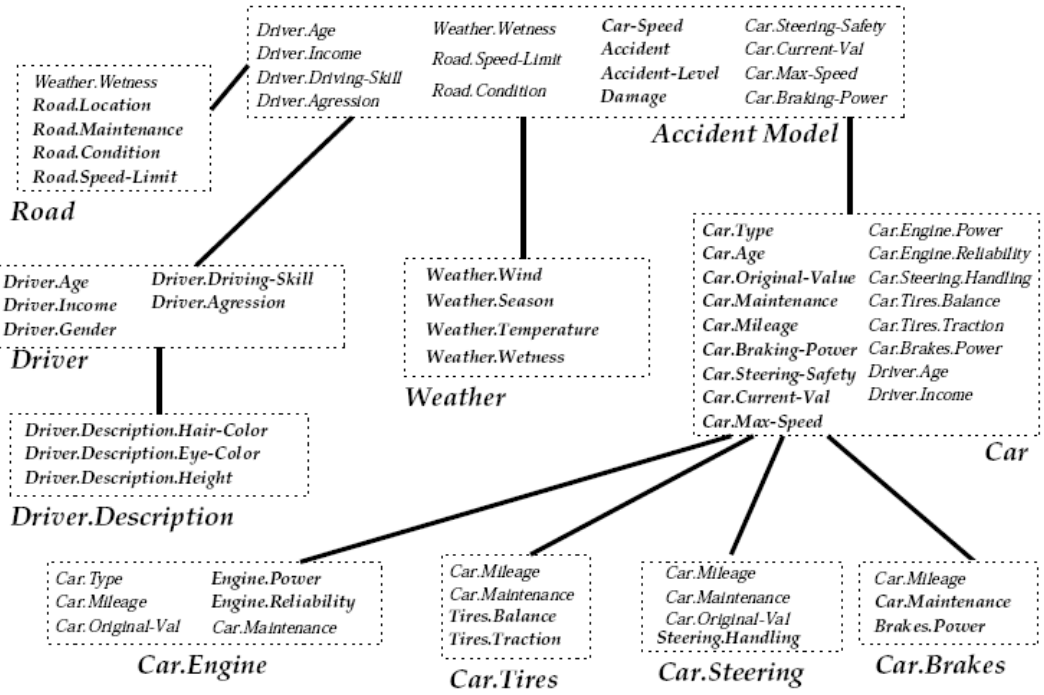


FIGURE 2.3 – L'hyper arbre d'un MSBN dans le contexte de l'accident de voiture. *Image provenant de [106].*

complexe de Y, nous pouvons ensuite organiser le MSBN comme un hyper arbre où il existe un hyperlien de $\sum_Y$ à $\sum_X$ si Y est un parent de X dans T. La figure 2.3 donne un exemple d'un tel MSBN. Ceci offre de grands avantages pour l'inférence : d'une part, la structure du RBOO peut être utilisée pour trouver de bonnes partitions dans le MSBN ; d'autre part, la gestion d'un cache permet la réutilisation de calculs pour les multiples objets d'une même classe. Pour des modèles avec une structure très répétitive, le gain peut être considérable.

En ce qui concerne l'apprentissage des RBOO pour finir, un algorithme a été proposé par Langseth et Bangsø pour estimer les paramètres d'un objet de situation et son ensemble de classes [111]. Du côté de l'apprentissage de structure, le seul travail publié à notre connaissance est celui de Bangsø et Langseth [6] sur un ensemble particulier d'instanciations de classes, avec éventuellement une caractérisation des variables en terme d'entrée ou de sortie.

Ainsi, un RBOO apporte plusieurs avantages par rapport aux RB classiques, incluant la possibilité de réutiliser des types et de de représenter des modèles où les individus ne sont plus i.i.d. Néanmoins, ces modèles ont encore quelques désavantages. En effet, même si ces modèles permettent une forme de réutilisabilité, ils ne permettent pas de découvrir des connaissances générales entre les différentes classes, en dehors de la situation précise étudiée via l'objet principal. Par conséquent, la moindre modification de l'objet de situation ou d'une classe donnée demandera de mettre à jour le RBOO par un expert ou un nouvel apprentissage complet du modèle. Les données et le modèle ne sont ici pas suffisamment découplés pour rendre le modèle réutilisable par une famille d'instances différentes. Les modèles relationnels probabilistes améliorent les possibilités de généralisation dans ce sens.

2.4.3 Schéma de base de données, contraintes multi relationnelles

Une étape de généralisation supplémentaire dans le cadre des jeux de données multi relationnels par rapport aux RBOO peut consister à découpler le domaine de définition des données et les données considérées pour faire de l'inférence ou de l'apprentissage. Par cette abstraction, nous pouvons alors nous intéresser à la découverte générale de connaissances dans ce domaine de définition pour différentes configurations de données. Une première étape nécessaire est d'étendre la définition de schéma pour le cas multi relationnel. Pour rappel, les connaissances définies ou apprises dans le cas mono relationnel se présentaient sous la forme de dépendances entre les différents attributs de l'en-tête d'un schéma, en faisant abstraction des tuples d'une relation particulière. Dans le cas multi relationnel, nous définissons le concept de schéma de base de données, avec un objectif similaire.

Définition 2.11. *Un schéma de base de données, ou schéma relationnel, est composé d'un ensemble de schémas de relation* $\mathbf{R} = \{\mathcal{R}^i\}_{i \leq m}$.

Dans ce contexte, l'extension du concept d'instance est naturelle.

Définition 2.12. *Soit un schéma de base de données* $\mathbf{R} = \langle \mathcal{R}^i = (\mathcal{H}_i, \mathcal{C}_i) \rangle_{i \leq m}$. *Nous appelons instance* $\mathcal{I}$ *de ce schéma une séquence de relations* $\langle R_i = (H_i, T_i) \rangle_{i \leq m}$ *telle que* $\forall i \leq m : H_i = \mathcal{H}_i$ *et les contraintes* C_i *sont valides pour* T_i.

Pour des raisons de simplicité, nous définissons l'ensemble $inst(\mathbf{R})$ des instances de $\mathbf{R}$. Nous notons également $\mathcal{I}(\mathcal{R}^i)$ la restriction d'une instance $\mathcal{I} \in inst(\mathbf{R})$ à un de ses schémas de relation $\mathcal{R}^i \in \mathbf{R}$.

Dans un paradigme multi relationnel, les contraintes des différents schémas de relations peuvent concerner d'autres schémas de relation. De nouvelles contraintes sont alors pertinentes. Nous les présentons ci-après.

La première catégorie de contraintes est liée à la notion d'unicité d'information permettant d'identifier un tuple particulier à partir d'un sous-ensemble de ses informations. Nous commençons par aborder le cas le plus général, via le concept de *super clé*.

Définition 2.13. *Étant donné un schéma de base de données et un de ses schémas de relation* $\mathcal{R} = (\mathcal{H}, \mathcal{C})$, *une contrainte de super clé* $c \in C$ *définit un sous-ensemble* H_k *de* $\mathcal{H}$ *identifiant un tuple de n'importe quelle instance de façon certaine, c.-à-d. une dépendance fonctionnelle de la forme* $H_k \rightarrow H \backslash H_k$.

Nous définissons ensuite la notion de *clé candidate*, qui n'est autre qu'une super clé minimale dans l'ensemble des attributs.

Définition 2.14. *Une contrainte de clé candidate* $c' \in C$ *est une contrainte de super clé minimale pour toute instance de* $\mathcal{R}$, *c.-à-d. un sous-ensemble* H_k *de* $\mathcal{H}$ *de telle sorte que* $H_k \rightarrow H \backslash H_k$ *et qu'il n'existe pas* $H'_k \subset H_k$ *tel que* $H'_k \rightarrow H \backslash H'_k$.

Plusieurs clés candidates peuvent exister dans un schéma de relation et il est souvent nécessaire de faire un choix parmi elles pour définir une clé candidate principale qui sera utilisée par d'autres contraintes, notamment pour créer des associations entre tuples. La clé candidate principale d'un schéma de relation est appelée *clé primaire*.

Définition 2.15. *Une contrainte de clé primaire $c'' \in C$ définit la clé candidate principale choisie pour identifier de façon unique un tuple pour toute instance du schéma de relation. Elle est unique pour le schéma de relation.*

La clé primaire de $\mathcal{R}$ est utilisée comme cible de toute contrainte d'intégrité référentielle ayant $\mathcal{R}$ pour cible (cf. Définition 2.17). Pour des raisons de simplicité, nous définissons la fonction pk qui associe à tout schéma de relation l'ensemble de ses attributs de clé primaire.

La seconde catégorie de contraintes permet de définir des inclusions de domaines entre différents ensembles d'attributs. De façon générale, nous pouvons ainsi définir une *dépendance d'inclusion* entre deux ensembles d'attributs, indiquant que le domaine du premier ensemble est inclus dans le domaine du second.

Définition 2.16. *Étant donné un schéma de base de données $\mathbf{R}$ et deux de ses schémas de relation $\mathcal{R}^i = (\mathcal{H}_i, \mathcal{C}_i)$ et $\mathcal{R}^j = (\mathcal{H}_j, \mathcal{C}_j)$, une dépendance d'inclusion entre $A_i \subseteq \mathcal{H}_i$ et $A_j \subseteq \mathcal{H}_j$ contraint chaque instance $\mathcal{I} \in inst(\mathbf{R})$ à avoir pour tout tuple t_i de $\mathcal{I}(\mathcal{R}^i)$ sa restriction $t_i[A_i]$ égale à la restriction $t_j[A_j]$ d'un tuple t_j de $\mathcal{I}(\mathcal{R}^j)$.*

La dépendance d'inclusion est particulièrement intéressante dans le cas où le second ensemble d'attributs est une clé primaire d'un schéma de relation $\mathcal{R}^i$. Dans ce cas, le premier ensemble d'attributs est dit lié par une *contrainte d'intégrité référentielle* à $\mathcal{R}^i$.

Définition 2.17. *Étant donné un schéma de base de données $\mathbf{R}$ et un schéma de relation $\mathcal{R}^i = (\mathcal{H}_i, \mathcal{C}_i)$, l'ensemble d'attributs $A_r \subseteq \mathcal{H}$ **référence** un schéma de relation $\mathcal{R}^j$ ssi il existe une contrainte d'inclusion entre A_r et $pk(\mathcal{R}^j)$.*

Cela signifie que pour toute instance $\mathcal{I} \in inst(\mathbf{R})$, la restriction $t_a[A_j]$ de chaque tuple t_a de $\mathcal{I}(\mathcal{R}^i)$ identifie un tuple $t_b \in \mathcal{I}(\mathcal{R}^j)$ de façon unique. Pour des raisons de simplicité, nous définissons la fonction fk qui à tout schéma de relation associe la séquence de ses contraintes d'intégrité référentielles. Nous définissons également les fonctions $fktarget$ et $fkattr$ associant respectivement à tout couple (schéma de relation, contrainte d'intégrité référentielle) la relation cible et l'ensemble des attributs source impliqués pour cette contrainte.

2.4.4 Squelette relationnel

L'inférence dans un MRP nécessite une connaissance partielle sur le jeu de données afin de générer le RBP. Cette connaissance partielle consiste en l'énumération des tuples existants, ainsi que les associations existant entre eux par le biais de contraintes référentielles, appelée *squelette relationnel*.

Définition 2.18. *Le squelette relationnel d'une instance $\mathcal{I}$, noté $\pi(\mathcal{I})$, est l'union des restrictions des tuples de $\mathcal{I}$ à leurs attributs impliqués soit dans une contrainte de clé primaire, soit dans une contrainte d'intégrité référentielle.*

Lorsqu'aucune ambiguïté ne sera possible, nous appellerons $\pi = \pi(\mathcal{I})$ le squelette relationnel considéré.

2.4.5 Chaîne de références

Les contraintes d'intégrité référentielles définissent des dépendances entre les divers schémas de relation qui peuvent être composées de façon à définir des chemins, appelés *chaînes de références*, dans le schéma de base de données. Ceux-ci sont particulièrement importants dans l'apprentissage de modèles dans un contexte multi relationnel pour construire des vues contextuelles de données dans le but de réaliser divers calculs de fréquences dans les bases de données.

Définition 2.19. *Étant donné un schéma de base de données* $\mathbf{R}$*, une chaîne de références* $\sigma = (\sigma_i)_{i \leq l}$ *de longueur* l *est une séquence telle que* $\sigma_i = (\mathcal{R}^i, f_i)$ *(élément de chaîne direct) ou* $\sigma_i = (\mathcal{R}^i, f_i)^{-1}$ *(élément inversé) avec* $\mathcal{R}^i \in \mathbf{R}$*,* $f_i \in fk(\mathcal{R}^i)$ *et* $\forall i < l$ *:*

- $\sigma_i = (\mathcal{R}^i, f_i), \sigma_{i+1} = (\mathcal{R}^{i+1}, f_{i+1}) \rightarrow ref_{\mathbf{R}}(\mathcal{R}^i, f_i) = \mathcal{R}^{i+1}$;
- $\sigma_i = (\mathcal{R}^i, f_i), \sigma_{i+1} = (\mathcal{R}^{i+1}, f_{i+1})^{-1} \rightarrow ref_{\mathbf{R}}(\mathcal{R}^i, f_i) = ref_{\mathbf{R}}(\mathcal{R}^{i+1}, f_{i+1})$;
- $\sigma_i = (\mathcal{R}^i, f_i)^{-1}, \sigma_{i+1} = (\mathcal{R}^{i+1}, f_{i+1}) \rightarrow \mathcal{R}^i = \mathcal{R}^{i+1}$;
- $\sigma_i = (\mathcal{R}^i, f_i)^{-1}, \sigma_{i+1} = (\mathcal{R}^{i+1}, f_{i+1})^{-1} \rightarrow ref_{\mathbf{R}}(\mathcal{R}^{i+1}, f_{i+1}) = \mathcal{R}^i$;

où la fonction $ref_{\mathbf{R}} : \mathbf{R} \times \Sigma(\mathbf{R}) \rightarrow \mathbf{R}$ *associe à chaque couple* $(\sigma, \mathcal{R})$ *le schéma de relation d'arrivée après avoir appliqué la chaîne à partir de* $\mathcal{R}$*, et où* $\Sigma(\mathbf{R})$ *définit l'ensemble des chaînes de référence possibles pour* $\mathbf{R}$*.*

Il est important de noter que l'ensemble $\Sigma(\mathbf{R})$ est infini. Une chaîne de références peut en effet être vue comme un chemin dans le multigraphe non dirigé où les nœuds sont les schémas de relation et une arête étiquetée existe entre deux nœuds pour chaque contrainte d'intégrité référentielle entre eux. Un exemple simple pour illustrer ce fait consiste à construire la chaîne de référence $\langle (\mathcal{R}^i, f_i), (\mathcal{R}^i, f_i)^{-1}, (\mathcal{R}^i, f_i), (\mathcal{R}^i, f_i)^{-1}, \ldots \rangle$.

Si nous notons $dom(\sigma_i)$, le domaine d'un élément d'une chaîne de référence, et $ran(\sigma_i)$ son co domaine, nous avons les égalités suivantes :

- $\sigma_i = (\mathcal{R}^i, f_i) \rightarrow dom(\sigma_i) = \mathcal{R}^i, ran(\sigma_i) = ref_{\mathbf{R}}(\mathcal{R}^i, f_i)$;
- $\sigma_i = (\mathcal{R}^i, f_i)^{-1} \rightarrow dom(\sigma_i) = ref_{\mathbf{R}}(\mathcal{R}^i, f_i), ran(\sigma_i) = \mathcal{R}^i$;

Ceci nous permet de redéfinir une chaîne de références de la façon alternative suivante.

Définition 2.20. *Étant donné un schéma de base de données* $\mathbf{R}$*, une chaîne de références* $\sigma = (\sigma_i)_{i \leq l}$ *de longueur* l *est une séquence où* $\sigma_i = (\mathcal{R}^i, f_i)$ *ou* $\sigma_i = (\mathcal{R}^i, f_i)^{-1}$*, avec* $\mathcal{R}^i = (H_i, T_i) \in \mathbf{R}$*,* $f_i \in fk(\mathcal{R}^i)$ *et* $\forall i < l, ran(\sigma_i) = dom(\sigma_{i+1})$*.*

2.4.6 Exemple

Un exemple de schéma relationnel pour 5 schémas de relations et 4 contraintes d'intégrité référentielles est montré dans la Figure 2.4. Nous appelons par la suite ce schéma relationnel par l'acronyme *UMA* (pour *User, Movie, Actor*). La Table 2.1 présente un exemple d'instance pour ce domaine de données et la Table 2.2 donne son squelette relationnel.

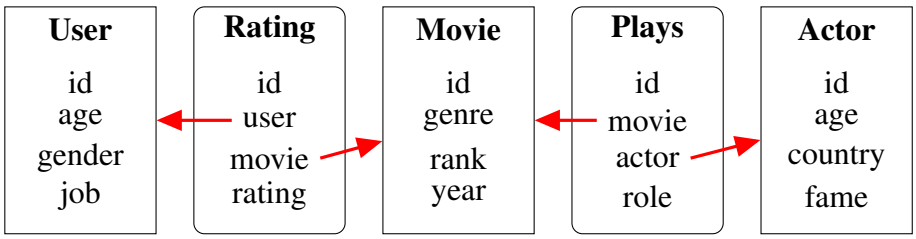


FIGURE 2.4 – Schéma de base de données avec 5 schémas de relations et 4 contraintes d'intégrité référentielles.

User			
id	age	gender	job
u1	"20-30"	"M"	"computing"
u2	"20-30"	"F"	"business"

Movie			
id	genre	rank	year
m1	"drama"	"top"	"1990s"
m2	"comedy"	"top"	"1970s"

Actor			
id	age	country	fame
a1	"30-40"	"France"	"low"
a2	"10-15"	"China"	"high"

Rating			
id	user	movie	rating
r1	u1	m1	4
r2	u2	m2	1

Plays			
id	movie	actor	role
p1	m1	a2	"main"
p2	m1	a1	"second"

TABLE 2.1 – Exemple d'instance du domaine UMA.

User			
id	age	gender	job
u1	?	?	?
u2	?	?	?

Movie			
id	genre	rank	year
m1	?	?	?
m2	?	?	?

Actor			
id	age	country	fame
a1	?	?	?
a2	?	?	?

Rating			
id	user	movie	rating
r1	u1	m1	?
r2	u2	m2	?

Plays			
id	movie	actor	role
p1	m1	a2	?
p2	m1	a1	?

TABLE 2.2 – Squelette relationnel de l'instance de la Table 2.1 du domaine UMA.

2.4.7 Opérations élémentaires d'algèbre relationnelle

Pour finir, nous définissons maintenant trois opérations élémentaires d'algèbre relationnelle : la *projection*, la *sélection*, ainsi que la *jointure*. Ces opérations sont communément utilisées pour explorer des données multi relationnelles et construire des relations par intention, appelées *vues*, permettant par la suite de réaliser de plus amples opérations, comme des comptages.

Définition 2.21. *Soit $R = (H, T)$ une relation et $H_p \subseteq H$ un ensemble d'attributs. La projection π_{H_p} d'un tuple $t_i \in T$ est sa restriction $t_i[H_p]$. Par extension, la projection π_{H_p} d'une relation R est une nouvelle relation $R_{H_p} = (H_p, T_{H_p})$ telle que :*

$$T_{H_p} = R[H_p] = \{\pi_{H_p}[t_i] | t_i \in T\}.$$

Définition 2.22. *Soit $R = (H, T)$ une relation et $\mathcal{C}$ une contrainte de sélection, c.-à-d. une formule logique sous la forme d'une combinaison de contraintes de comparaison et de connecteurs logiques. La sélection $\sigma_{\mathcal{C}}$ de R est une nouvelle relation $R_{\mathcal{C}} = (H, T_{\mathcal{C}})$ où :*

$$T_{\mathcal{C}} = \{t_i | t_i \textit{ satisfait } \mathcal{C}\}.$$

Définition 2.23. *Soient $R_1 = (H_1, T_1)$ et $R_2 = (H_2, T_2)$, deux relations. Soient $H_{j1} \subseteq H_1$ et $H_{j2} \subseteq H_2$, des sous-ensembles d'attributs pour chacune de ces relations. Une jointure de R_1 et R_2 sur H_{j1} et H_{j2} est une nouvelle relation $R_J = (H_J, T_J)$ définie de la façon suivante :*

$$H_J = H_1 \cup H_2,$$

$$T_J = \{\, t = \langle t_1, t_2 \rangle \mid (t_1, t_2) \in T_1 \times T_2,\ t1[H_{j1}] = t2[H_{j2}] \,\}.$$

2.4.8 Modèles relationnels probabilistes

De façon similaire aux RBOO, les modèles relationnels probabilistes (MRP) [62] sont une famille de modèles permettant de raisonner dans un contexte multi relationnel incertain où l'hypothèse d'indépendance des tuples n'est pas vérifiée. Toutefois, les MRP ont pour objectif la modélisation d'une connaissance plus générale que les RBOO pour un domaine de définition donné. En effet, là où un RBOO oblige la définition d'un objet de situation à partir duquel les dépendances probabilistes entre variables sont définies ou apprises, un MRP a pour objectif de découvrir des dépendances probabilistes entre les différents attributs d'un schéma de base de données. L'apprentissage d'un MRP se fait toujours grâce à l'utilisation d'une instance du schéma de base de données, mais les connaissances découvertes en font abstraction. Ainsi, un MRP défini sur un schéma de base de données est compatible avec toutes les configurations de données qui respectent les contraintes de ce schéma. Ce dernier peut alors être vu comme un contrat à respecter entre le MRP et l'instance pour garantir leur compatibilité.

Le découplage entre MRP et instance est très intéressant puisqu'il permet en théorie de transférer un savoir acquis suite à l'apprentissage sur une instance d'un schéma de base de données à une autre instance pour raisonner sur de nouveaux tuples. Cet usage des MRP suppose l'existence d'une certaine généralisation des connaissances entre différentes bases de données, ce qui n'est malheureusement pas toujours vérifié. Plus vraisemblablement, le découplage permet d'apprendre des connaissances à partir

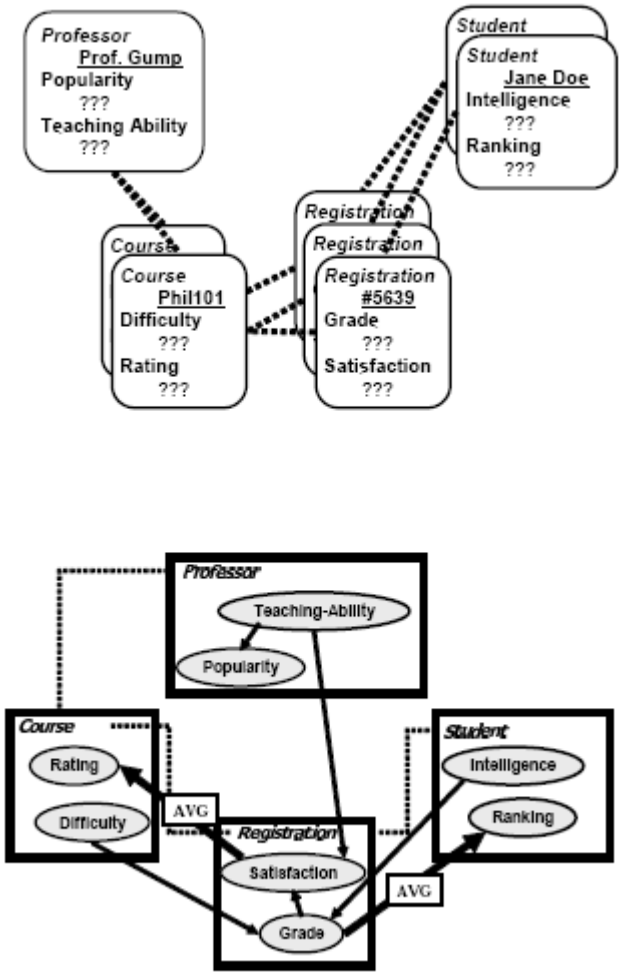


FIGURE 2.5 – (haut) Squelette relationnel pour une instance dans le contexte des inscriptions étudiantes (bas) La structure de MRP pour ce domaine. *Images provenant de [65]*

d'une base de données pour ensuite réaliser des inférences sur les états futurs de cette même base de données, sous hypothèse plus raisonnable à intervalles de temps courts de stationnarité des distributions de probabilités, ou encore de réaliser un apprentissage sur un ensemble de tuples avec données manquantes en ne prenant que ceux étant complètement définis, selon le principe de l'analyse des cas disponibles (*available case analysis* dans la littérature), pour ensuite réaliser une inférence sur les tuples restants.

Les MRP sont définis formellement de la façon suivante.

Définition 2.24. *[62] Étant donné un schéma de base de données* $\mathbf{R} = \langle \mathcal{R}^i \rangle_{i \leq m}$*, un MRP* $\mathcal{M} = (\mathcal{S}, \Theta)$ *est composé d'une structure* $\mathcal{S}$ *et d'un ensemble de paramètres* Θ*. La structure est composée d'un ensemble de variables aléatoires* $\mathcal{V}$ *et d'une fonction de parents* $pa : \mathcal{V} \rightarrow \mathcal{P}(\Psi \times \Sigma(\mathbf{R}) \times \mathcal{V})$ *où* Ψ *est un ensemble de fonctions d'agrégation (contenant la fonction* id *d'identité) et* $\Sigma(\mathbf{R})$ *est l'ensemble des chaînes de références qu'il est possible de définir sur* $\mathbf{R}$*. L'ensemble* $\mathcal{V}$ *contient une variable aléatoire* $A_{i,j}$ *pour chaque attribut* $\mathcal{A}_j \in att(\mathcal{R}^i)$ *avec* $dom(A_{i,j}) = ran(\mathcal{A}_j)$*. Finalement, nous avons* $\Theta = \{\theta_{i,j} = P(A_{i,j}|pa(A_{i,j}))\}$.

La fonction pa doit respecter certaines contraintes pour définir un MRP valide. D'abord, la fonction doit être telle que toute instanciation future du MRP donnera lieu à un RB valide, donc sans circuit. De plus, l'image de toute variable $A_{i,j} \in \mathcal{V}$ par la fonction pa ne doit contenir que des triplets $(\psi, \sigma, A_{a,b}) \in pa(A_{i,j})$ tels que si σ est une chaîne de références non vide, alors ψ doit être une fonction d'agrégation différente de l'identité et $dom(A_{a,b}) = ran(\sigma)$; sinon, ψ doit être la fonction d'identité id et $dom(A_{a,b}) = dom(A_{i,j})$.

Il est important de noter que les RB sont des MRP particuliers. En effet, considérant un schéma de base de données $\mathbf{R}_1$ avec un seul schéma de relation, sans contrainte d'intégrité référentielle, tout MRP défini sur $\mathbf{R}_1$ est un RB. En effet, sans contrainte référentielle, toutes les chaînes de référence pour l'ensemble des parents définis dans le MRP doivent nécessairement être nulles, impliquant que les fonctions d'agrégations doivent également être égales à id. Chaque triplet parent fait donc référence à un attribut singleton du même tuple pour toute instance, et l'on revient à la définition d'un RB où chaque tuple d'une instance est indépendant des autres.

2.4.9 Instanciation d'un MRP

Considérant un MRP valide et une instance compatible, il est possible de générer un unique RB, appelé *réseau bayésien à plat* (RBP) de l'instance pour le MRP considéré.

Définition 2.25. *[62] Soit $\mathcal{M} = ((\mathcal{V}, pa), \Theta)$, un MRP valide et une instance $\mathcal{I} \in inst(\mathbf{R})$ compatible, le réseau bayésien à plat $\mathcal{G}(\mathcal{M}, \mathcal{I}) = (V_\mathcal{I}, pa_\mathcal{I}, \Theta_\mathcal{I})$ est défini de la façon suivante :*

1. *une variable aléatoire $a_{e,i,j} \in V_\mathcal{I}$ existe pour tout $A_{i,j}$ de $\mathcal{M}$, avec $dom(a_{e,i,j}) = dom(A_{i,j})$ pour tout $e \in \pi(\mathcal{R}^i)$;*
2. *pour tout $A_{i,j}$ et chacun de ses parents $(\psi, \sigma, V_{a,b}) \in pa(V_{i,j})$, nous avons :*
$$\forall s \in \mathcal{I}(\mathcal{R}^i) \forall t \in \pi(\mathcal{R}^a) \ : \ (a_{s,i,j}, \psi, \sigma, a_{t,a,b}) \in pa_\mathcal{I};$$
3. *une CPD $P(a_{e,i,j}|pa(a_{e,i,j})) \in \Theta_\mathcal{I}$ est définie pour tout $a_{e,i,j} \in V_\mathcal{I}$ avec :*
$$P(a_{e,i,j}|pa(a_{e,i,j})) = P(A_{i,j}|pa(A_{i,j})).$$

.

Il est important de noter que toutes les instanciations d'une même variable aléatoire du MRP dans le RBP sont identiques. Elles sont dites *liées* au même paramètre.

Un RBP est l'instanciation d'un MRP qui décrit une unique distribution jointe de probabilités pour les individus de $\mathcal{I}$ en utilisant la connaissance décrite par $\mathcal{M}$. La fonction de vraisemblance associée à cette distribution nous donne sa sémantique :

$$P(\mathcal{I}|\mathcal{M}, \pi) = \prod_{\mathcal{R}^i \in \mathbf{R}} \prod_{\mathcal{A}_j \in att(\mathcal{R}^i)} \prod_{e \in \mathcal{I}(\mathcal{R}^i)} P(a_{e,i,j}|pa(a_{e,i,j})) \tag{2.4}$$

Dans le cas où le modèle est simple et le squelette est petit, l'inférence exacte dans les MRP peut être faite directement sur les RBP, par l'utilisation de techniques d'inférence dédiées aux RB [152, 45]. Raisonner de façon exacte est également possible quand le modèle contient de nombreuses régularités. Dans ce cas, l'utilisation

de techniques proches de celles utilisées pour les RBOO peut donner de bons résultats, impliquant la définition de MSBN et l'utilisation d'un cache applicatif. Certains travaux se sont également focalisés sur des approches plus élaborées de génération partielle de RBP [156, 168] utilisant les régularités du RBP pour en produire une version factorisée plus petite. Toutefois, cette généralisation partielle exacte n'est pas toujours intéressante, puisque nombre de configurations de données ne présentent pas ou peu de régularités exactes. Ainsi, en examinant localement les configurations de données des instanciations d'une même variable du MRP, celles-ci sont généralement sensiblement différentes, en terme de nombre de parents par exemple. Aussi, des travaux se sont intéressés à la génération de RBP partiels approchés [169]. D'une façon plus générale, lorsque le MRP engendre des RBP d'une certaine dimension, l'inférence exacte est trop complexe et il faut alors utiliser des algorithmes approchés (cf. 2.3.1).

2.4.10 MRP garantis acycliques

Un MRP est un modèle général sur un domaine de définition défini par un schéma de base de données. Toutefois, son intérêt premier provient de sa capacité à être instancié pour un jeu de données particulier, celle-ci étant nécessaire pour effectuer un raisonnement sur une base de données spécifique. Aussi faut-il pouvoir garantir qu'un MRP pourra être instancié pour tout jeu de données respectant le schéma de base de données. Ce problème est indécidable dans le cas général [91] et il est donc impossible de discriminer de façon exacte l'ensemble des MRP que l'on peut instancier de son complément. Cependant, il est possible de définir des conditions suffisantes à respecter au niveau du MRP pour assurer que tout RBP obtenu par instanciation sera valide, c.-à-d. acyclique. Un MRP est dit *garanti acyclique* s'il respecte ces conditions.

La vérification de cette propriété pour un MRP nécessite la construction d'un graphe, appelé *graphe de dépendances*. Le graphe de dépendance $\mathcal{G}_{\mathcal{M}}$ d'un MRP $\mathcal{M} = ((\mathcal{V}, pa), \Theta)$ contient :

- un noeud pour chaque variable aléatoire dans $\mathcal{V}$;
- un arc de $A_{i,j}$ vers $A_{a,b}$ si $\exists \psi, \sigma | (\psi, \sigma, A_{a,b}) \in pa(A_{i,j})$.

Un MRP est ainsi garanti acyclique si son graphe de dépendance est garanti acyclique. Certains MRP dont le graphe de dépendance n'est pas acyclique sont aussi garantis acycliques. Par exemple, dans un contexte génétique, le codage d'un gêne pour un individu dépend entre autres du codage du même gêne pour ses parents, etc. Au niveau conceptuel, le schéma de base de données contient un schéma de relation représentant les personnes et un second schéma de relation contenant deux contraintes d'intégrité référentielles liant les individus par lien de parenté. Toutefois, il est impossible que la dépendance probabiliste entre le codage d'un gêne pour un individu et celui de ses parents entraîne un quelconque cycle dans le RBP d'une instance associée, car la relation de parenté est arborescente. Il est possible de considérer ce genre de situation dans les MRP, en acceptant les cycles dans le graphe de dépendance qui n'induisent pas de chaînes de référence non garanties acycliques, mais ce contexte ne sera pas détaillé ici (cf. [62]).

2.4.11 Apprentissage d'un MRP

Comme pour les RB, l'apprentissage d'un MRP se découpe en deux phases : l'apprentissage des paramètres d'une part et l'apprentissage de la structure d'autre part.

i Estimation des paramètres

L'estimation des paramètres d'un MRP est proche de celle définie précédemment pour les RB. Chaque variable $A_{i,j}$ d'un MRP possède en effet un ensemble de parents $pa(A_{i,j})$ tel que la conjonction de variables $A_{i,j} \cup pa(A_{i,j})$ forme une relation R pour tout jeu de données compatible $\mathcal{I}$. Cette relation est une vue de la base de données originale, contenant un unique tuple pour chaque tuple de la relation originale $\mathcal{I}(\mathcal{R}^i)$. Aussi, chaque variable $A_{i,j}$ donne lieu à la création de son propre jeu de données mono relationnel dans lequel il est possible de calculer les fréquences nécessaires à l'estimation des paramètres, et ce de la même façon que pour un RB. Formellement, nous pouvons ainsi définir selon l'approche par maximum de vraisemblance :

$$\hat{P}(A_{a,i} = a_k | pa(A_{a,i}) = \mathbf{u_j}, \mathcal{I}) = \frac{N_{aijk}}{N_{aij}} \tag{2.5}$$

où N_{aijk} (resp. N_{aij}) est le nombre de tuples dont l'attribut $A_{a,i}$ est dans l'état a_k (resp. dans n'importe quel état) alors que les parents de cet attribut dans le graphe sont dans l'état $\mathbf{u_j}$.

Une estimation bayésienne pour ces paramètres, en utilisant comme précédemment un a priori de Dirichlet pour le cas de variables multinomiales, nous donne le résultat suivant :

$$\hat{P}(A_{a,i} = a_k | pa(A_{a,i}) = \mathbf{u_j}, \mathcal{I}) = \frac{N_{aijk} + \alpha_{aijk}}{N_{aij} + \alpha_{aij}} \tag{2.6}$$

où les α_{aijk} sont les hyper paramètres de l'a priori de Dirichlet correspondant aux N_{aijk}.

ii Apprentissage de structure

L'apprentissage d'un MRP tel que décrit dans la littérature [62, 65, 66] se fait essentiellement via recherche gloutonne, comme pour les RB. Les opérateurs de voisinage utilisés sont les mêmes que pour les RB, à savoir l'ajout ou la suppression d'un parent à une variable, ainsi que l'inversion d'une relation parent / enfant. Toutefois, les versions relationnelles de ces opérateurs demandent des vérifications supplémentaires en amont avant d'être applicables. Ces vérifications s'appliquent à chaque ajout d'un parent (lors de l'utilisation de l'opérateur d'ajout ou d'inversion). D'une part, le parent $(\phi, \sigma, A_{a,b})$ d'une variable $A_{i,j}$ doit être tel qu'il soit possible d'aller de $\mathcal{R}^i$ à $\mathcal{R}^a$ en suivant la chaîne de référence σ. D'autre part, l'agrégateur ϕ ne peut pas être nul si la chaîne σ contient une partie de chaîne inversée. En effet, dans ce cas, cela signifie qu'une relation $1, n$ a été rencontrée, et qu'à chaque tuple de $\mathcal{R}^i$ ne correspond plus nécessairement un seul tuple dans la relation localement construite pour les calculs de fréquence, empêchant toute estimation de paramètres dans ce cadre.

Une deuxième différence oppose l'apprentissage de structure d'un RB et celle d'un MRP selon l'approche gloutonne. En effet, le nombre de parents potentiels pour une variable est très dépendant du nombre de chaînes de références qu'il est possible de construire à partir de celle-ci. Or, l'ensemble des chaînes de références sur un schéma de base de données contenant au moins une contrainte de référence est infini. De ce

fait, il est impossible de générer la liste exhaustive des voisins pour un MRP candidat. Par conséquent, il faut introduire un biais de recherche dans l'algorithme glouton.

Appliquant le principe de parcimonie, encourageant le choix d'un modèle simple lorsque plusieurs modèles sont performants, le biais de recherche doit privilégier les structures les plus simples. En ce qui concerne les chaînes de références, un modèle est dit simple si ces chaînes sont courtes. Aussi, le biais doit encourager les chaînes les plus courtes ou pénaliser les chaînes en fonction de leur longueur. Ce biais se matérialise de deux façons dans les algorithmes d'apprentissage de MRP : d'une part, la recherche de voisinage se fait toujours à une itération pour une longueur fixe de chaîne de référence et le parcours de l'espace des modèles se fait tant qu'il est possible d'améliorer le score de la structure en conservant une longueur de chaîne fixe pour l'exploration des voisinages. Une fois alors qu'une convergence est obtenue, la longueur de chaîne considérée est incrémentée et l'apprentissage continue ; d'autre part, la fonction objectif permettant d'évaluer les modèles est pondérée de façon à pénaliser les structures en fonction de la longueur de l'ensemble de leurs chaînes de référence. L'algorithme complet de recherche gloutonne pour un MRP est donné par l'Algorithme 1. La fonction $neighbors(\mathcal{M}, s)$ retourne l'ensemble des voisins de $\mathcal{M}$ pour une taille de chaîne de référence fixée à s. De son côté, la fonction $best(\mathcal{M}^{\mathcal{C}})$ retourne le meilleur modèle parmi un ensemble de candidats selon une fonction d'évaluation.

Algorithme 1 Algorithme de recherche gloutonne pour l'apprentissage de MRP

Paramètres : $\mathcal{M}_{init}$: modèle de départ
Retourne : $\overset{*}{\mathcal{M}}$: optimum local obtenu à partir de $\mathcal{M}_{init}$

```
s ← −1
M*_{≤s} ← M_init
Répéter
    s ← s + 1
    M*_{≤s} ← M*_{≤s−1}
    Répéter
        M^C_{≤s} ← neighbors(M*_{≤s}, s)
        M*_{≤s} ← best(M^C_{≤s})
    Jusqu'à M*_{≤s} converge
Jusqu'à M*_{≤s} = M*_{≤s−1}
Retourner M*_{≤s}
```

Au-delà des approches gloutonnes, un algorithme hybride mêlant approche par contraintes et approche gloutonne a très récemment été proposé pour l'apprentissage de MRP par Ben Ishak [11], adaptant aux MRP l'algorithme *Max-Min Hill Climbing* de Tsamardinos et coll. [177].

2.4.12 Évaluation des candidats

En apprentissage de structure de MRP, comme pour les RB, nous sommes intéressés par la découverte d'un modèle le plus fidèle possible à la distribution de données réelle. Cette distribution étant inconnue, il faut trouver une façon de comparer les modèles afin de savoir lequel est le meilleur pour cette tâche. Formellement, nous voulons trouver

un modèle $\mathcal{M} = (\mathcal{S}, \Theta)$ maximisant $P(\mathcal{S}|\mathcal{I}, \pi) \propto P(\mathcal{I}|\mathcal{S}, \pi)P(\mathcal{S}|\pi)$. Le premier terme est la vraisemblance marginale de la loi jointe de distribution sur l'ensemble des paramètres de la structure correspondante :

$$P(\mathcal{I}|\mathcal{S}, \pi) = \int_{\Theta} P(\mathcal{I}|\Theta, \mathcal{S}, \pi)P(\Theta|\mathcal{S}, \pi)d\Theta$$

où $P(\mathcal{I}|\Theta, \mathcal{S}, \pi)$ est défini dans (2.4), et $P(\Theta|\mathcal{S}, \pi)$ est l'a priori sur les paramètres. Afin d'avoir un score décomposable, il faut pour rappel que chaque a priori satisfasse les propriétés d'indépendance globale des paramètres et de modularité des paramètres. Ces propriétés sont par exemple assurées par les a priori de Dirichlet. Dans ce cas, $P(\mathcal{I}|\mathcal{S}, \pi)$ a une forme close :

$$P(\mathcal{I}|\mathcal{S}, \pi) = \prod_{\mathcal{R}^a \in \mathbf{R}} \prod_{V_i \in \mathcal{V}(\mathcal{R}^a)} \prod_{\mathbf{u}_j \in dom(pa(V_i))} \frac{\Gamma(\alpha_{aij})}{\Gamma(\alpha_{aij} + N_{aij})} \prod_{k \in dom(V_i)} \frac{\Gamma(\alpha_{aijk} + N_{aijk})}{\Gamma(\alpha_{aijk})}.$$

Le second terme est l'a priori sur la structure du graphe, étant donné le squelette relationnel π. Dans le cas des RB, cet a priori est généralement uniforme et n'apparaît donc pas explicitement. Dans le cas des MRP, à cause du nombre infini de chaînes de références qu'il est possible de construire pour un schéma de base de données, l'a priori uniforme n'est pas raisonnable.

Supposant que la structure du graphe ne dépend pas du squelette relationnel, nous pouvons réécrire l'a priori de structure en $P(\mathcal{S})$. Cet a priori est choisi de façon à pénaliser les parents impliquant de longues chaînes de références. Cette propriété repose sur une information locale pour chaque variable aléatoire et satisfait donc la propriété de modularité structurelle, permettant la décomposabilité du score. Plus précisément, la littérature [66] définit l'a priori de structure de telle façon que $log(P(\mathcal{S})) \propto -K_r$, où K_r dénote la somme totale des tailles de chaînes de références pour l'ensemble des variables du modèle.

2.5 Autres modèles probabilistes

De nombreux modèles graphiques probabilistes existent dans un contexte propositionnel au-delà des réseaux bayésiens, comme les réseaux de Markov [152] et les réseaux de dépendance [82]. C'est donc tout naturellement que des adaptations du second ordre de ces modèles ont été proposées dans un contexte relationnel ou de logique des prédicats, tels que les programmes bayésiens logiques [97], BLOG [131], les réseaux logiques de Markov [160], les réseaux de Markov relationnels [174], les réseaux bayésiens relationnels de Jaeger [90], et les réseaux de dépendance relationnels [139]. Ces modèles sont décrits brièvement avec d'autres dans [100]. À ces modèles peuvent être ajoutés les DAPER [84, 127], ainsi que les réseaux somme-produit relationnels [136].

Deux catégories de modèles probabilistes peuvent être identifiées, comme pour leurs homologues propositionnels, dépendant de la nature dirigée ou non des graphes construits. Chaque famille possède des avantages et inconvénients par rapport aux autres. Comme rappelé dans [100], les modèles dirigés sont généralement plus rapides à apprendre que les modèles non dirigés grâce à leurs fonctions de score décomposable

localement pour chaque variable aléatoire. De plus, ils ne nécessitent pas de normalisation coûteuse des paramètres à chaque itération puisque ces paramètres représentent déjà des distributions de probabilité conditionnelles. Finalement, ils sont généralement plus adéquats pour exprimer certaines formes de connaissance particulières comme les liens de causalité entre variables. En revanche, la contrainte d'acyclicité est sujette à débat et peut poser de sérieux problèmes de modélisation dans certains contextes. Aussi, certains rejettent l'usage de tels modèles au profit de modèles non dirigés.

De façon symétrique, les modèles non dirigés permettent une plus grande flexibilité de modélisation au prix de temps de calcul plus importants. En règle général, l'apprentissage de tels modèles est effectué de façon approchée, où le calcul du maximum de vraisemblance est remplacé par celui du maximum de pseudo-vraisemblance [12]. Calculer ce dernier rapidement nécessite d'utiliser des méthodes d'inférence approchée, tel que celles de type *Markov chain Monte Carlo* (MCMC), la plus célèbre étant *l'échantillonnage de Gibbs* [173].

Parmi les modèles dirigés, le choix des MRP naît entre autres du constat que la majorité des données sont stockées dans des systèmes de gestion de base de données relationnelles. Les MRP fonctionnant directement à partir d'un schéma de base de données, ils sont plus facilement interprétables par les experts d'un domaine et ne demandent pas l'ajout d'une couche de représentation. De plus, les systèmes de base de données relationnels sont des technologies éprouvées, bénéficiant de dizaines d'années de recherche et permettant de réaliser les différentes interrogations aux données nécessaires à l'apprentissage de façon correcte, simple et robuste. En outre, les MRP proposent des méthodes d'apprentissage dans le contexte d'incertitude de structure entre tuples, problématique au coeur de cette thèse et de la littérature récente existe pour leur utilisation concrète dans des cadres applicatifs professionnels [24].

2.6 Conclusion

Nous avons présenté dans ce chapitre les concepts généraux nécessaires à la compréhension de nos travaux. Nous avons ainsi commencé par définir le paradigme de représentation mono relationnel ou propositionnel, où un jeu de données est constitué d'un ensemble de tuples identiquement et indépendamment distribués, avant d'aborder les concepts de dépendance fonctionnelle puis probabiliste dans ce contexte pour enfin aborder la définition, l'inférence ainsi que l'apprentissage de réseaux bayésiens. En nous appuyant sur les limites des représentations propositionnelles, notamment les contraintes fortes de l'hypothèse i.i.d. opposées aux phénomènes d'autocorrélation généralement observés dans les jeux de données réels, nous justifions l'intérêt d'envisager un paradigme multi relationnel de représentation de données. Nous décrivons ainsi deux extensions des réseaux bayésiens dans ce contexte. D'une part, les réseaux bayésiens orientés objet proposent une généralisation sous la forme de structure en classes permettant le partage de distributions entre instances de cette même classe. Ils nécessitent toutefois la définition d'un objet de situation où les instances de classe sont énumérées et conditionnent le savoir qu'il est possible d'apprendre, rendant impossible le découplage complet entre données et modèle. D'autre part, les modèles relationnels probabilistes permettent un degré de généralisation supplémentaire puisqu'ils sont directement définis sur la description des types de données existants et non plus sur une configuration de données particulière.

Nous considérons à la fin de ce chapitre le seul cas des modèles relationnels probabilistes sous incertitude d'attributs, avec connaissance complète des liens entre les

individus d'une instance. Nous considérons dans le prochain chapitre le cas plus complexe des modèles relationnels probabilistes avec incertitude d'attributs et de liens, plus particulièrement dans un contexte d'incertitude de références, où la liste des tuples de chaque type est connue, mais pas leurs valeurs d'attributs et de références entre eux.

3

Modèles relationnels probabilistes et incertitude de référence

‘

Sommaire

3.1 Introduction

Dans leur version la plus simple, les modèles relationnels probabilistes (MRP) représentent une connaissance impliquant les différents attributs d'un schéma de base de données, reposant sur l'hypothèse que pour toute instance de ce schéma, le squelette relationnel peut être obtenu de façon complète. Ceci implique que toute instance doit connaître le nombre de tuples pour chaque relation et que chaque contrainte d'intégrité référentielle pour tout tuple concerné est certaine. En d'autres termes, cela signifie que les références entre tuples sont connues. Inférer sur les instanciations de tels MRP est alors restreint aux valeurs des attributs descriptifs de ces tuples.

L'hypothèse de certitude de références entre tuples n'est pas toujours vérifiée. À titre d'exemple, dans un contexte de recommandation, nous ne connaissons a priori pas toutes les associations susceptibles d'exister entre des utilisateurs et les produits en vente et la tâche de recommandation vise justement à inférer la valeur d'associations inconnues à partir des informations certaines sur les entités et associations du domaine. Dans ce type de contextes où nous souhaitons pouvoir inférer sur les références entre tuples, il nous faut avoir recours à des versions étendues des MRP [67, 66].

Le but principal des MRP avec *incertitude de référence* est de retrouver les références entre ensembles de tuples dont les extensions sont connues, mais pour lesquels les valeurs de référence sont inconnues. À titre d'exemple, nous pouvons considérer un système spécialisé dans l'analyse d'articles scientifiques anciens, dont les références sont illisibles, mais où le nombre de citations est connu ou peut être estimé, ayant pour objectif de retrouver les paires $cite(a, b)$ indiquant que a cite b. Dans le cas où le nombre de références est connu, mais pas leurs valeurs effectives, l'extension d'incertitude de référence est un bon candidat pour résoudre le problème.

Il est important de noter que le paradigme d'incertitude de référence peut également être utilisé dans des contextes où le nombre réel de liens entre objets est inconnu. Cette information n'est d'ailleurs parfois pas pertinente et peut-être fixée arbitrairement par un expert. Ainsi, un problème de recommandation d'objets pour des utilisateurs peut être vu comme un problème d'inférence de valeurs de références entre des utilisateurs et des objets, étant donné un nombre prédéfini de liens correspondant au nombre de recommandations à proposer pour chaque utilisateur.

Ce chapitre est dédié à la description des MRP avec incertitude de référence, ainsi qu'à leur critique. Il est important de noter que ces modèles n'ont été que très peu détaillés dans la littérature [65, 67, 66] et n'ont jusqu'ici été que très peu utilisés. Une contribution de cette thèse, proposée dans ce chapitre, est l'explicitation de ces modèles sur des points non détaillés dans la littérature.

3.2 Entités et Associations

Afin de faciliter le discours dans la suite de ce chapitre, et plus généralement dans ce document, il nous faut apporter des précisions sur la description d'un schéma de base de données dans le cadre d'une application aux MRP avec incertitude de référence. Nous partitionnons en effet dorénavant un schéma de base de données en deux sous-ensembles de schémas de relations, appelés respectivement *types d'entités* et *types d'associations*.

Définition 3.1. *Soit* $\mathbf{R}$ *un schéma de base de données. Nous pouvons diviser l'ensemble de ses schémas de relation en deux sous-ensembles,* $\mathcal{R}_e$ *et* $\mathcal{R}_a$*. Un schéma de*

relation $\mathcal{R} \in \mathcal{R}_e$ est appelé type d'entités et ne comporte aucune contrainte de référence, tandis qu'un schéma de relation $\mathcal{R} \in \mathcal{R}_a$ est appelé type d'association et comporte au moins une contrainte de référence. Chaque tuple d'une instance de type d'entité (resp. d'association) est appelé entité (resp. association).

3.3 Incertitude de référence

Un jeu de données relationnel est composé de tuples instanciant différents schémas de relation. Dans le cas de MRP orientés attributs, tels que définis dans le chapitre précédent, les modèles appris prennent la forme d'un ensemble de dépendances probabilistes et de distributions de probabilités conditionnelles entre les différents attributs des différents schémas de relation de la base de données. À partir d'un tel modèle, une prédiction sur une nouvelle instance du schéma de base de données considéré se fait obligatoirement en considérant les références entre tuples de cette instance comme connaissance *a priori*. Les références sont dans ce contexte supposées exactes et complètes, et un MRP permet alors d'inférer les valeurs d'attributs des différents tuples en fonction des valeurs des attributs de ses tuples directement ou indirectement voisins dans le squelette relationnel de l'instance.

Il est possible d'étendre le degré d'incertitude sur les jeux de données en considérant à la fois une incertitude d'attributs, mais aussi une incertitude structurelle entre tuples, c.-à-d. sur leurs associations entre eux. Une manière de modéliser cette incertitude de structure est le *paradigme d'incertitude de référence*. Dans ce contexte, un MRP définit des dépendances probabilistes et des distributions de probabilités conditionnelles entre les différents attributs, mais aussi entre les différentes variables de référence des différents schémas de relation de la base de données. Une prédiction à partir d'un tel modèle se fait sans connaissance préalable des valeurs de références entre tuples, ou sur une connaissance partielle de celles-ci, mais sachant toujours une énumération complète de l'ensemble des associations pour chaque type. Un objectif de prédiction peut à la fois être une valeur d'un attribut de tuple ou une valeur de référence pour une association particulière, dans le dernier cas sous la forme d'un identifiant de tuple du schéma de relation concerné.

3.4 Squelette objet

Contrairement au squelette relationnel d'une instance décrivant l'ensemble des tuples existant ainsi que les associations entre eux, où les valeurs de toutes les références sont connues, l'information disponible pour inférer dans une instance de schéma de base de données dans un contexte d'incertitude de référence est réduite. Le *squelette objet* d'une instance décrit ainsi l'ensemble des entités et des associations existantes, sans toutefois connaître pour ces dernières les valeurs de leurs références.

Définition 3.2. *Le squelette objet d'une instance $\mathcal{I}$, noté $\pi(\mathcal{I})$, est l'union des restrictions des tuples de $\mathcal{I}$ à leurs seuls attributs impliqués dans une contrainte de clé primaire.*

Intuitivement, alors qu'un squelette relationnel décrit un hyper graphe dont les nœuds sont les entités (sans valeur pour les attributs) et où il existe un hyper arc pour

User			
id	age	gender	job
u1	?	?	?
u2	?	?	?

Movie			
id	genre	rank	year
m1	?	?	?
m2	?	?	?

Actor			
id	age	country	fame
a1	?	?	?
a2	?	?	?

Rating			
id	user	movie	rating
r1	?	?	?
r2	?	?	?

Plays			
id	movie	actor	role
p1	?	?	?
p2	?	?	?

TABLE 3.1 – Exemple de squelette objet pour une instance du domaine UMA.

chaque association reliant les entités qu'elle référence, le squelette objet décrit un ensemble de nœuds non liés entre eux.

Lorsqu'aucune ambiguïté ne sera possible, nous appellerons $\pi = \pi(\mathcal{I})$ le squelette objet considéré.

La Table 3.1 présente le squelette objet d'une instance du domaine UMA comprenant 2 entités et associations de chaque type. Le squelette objet de l'instance consiste en la seule conservation des identifiants de tuple pour chaque schéma de relation.

3.5 Fonctions de partition

Soit un schéma de base de données $\mathbf{R} = \mathcal{R}_e \cup \mathcal{R}_a$, un type d'association $\mathcal{R}_a^i \in \mathcal{R}_a$ et une contrainte de référence $f \in fk(\mathcal{R}_a^i)$ avec $ran(f) = \mathcal{R}^j$. Pour une instance $\mathcal{I} \in inst(\mathbf{R})$, tout tuple $t \in \mathcal{I}(\mathcal{R}_a^i)$ doit contenir une valeur pour f comprise dans $pk(\mathcal{I}(\mathcal{R}^j))$. Comme les valeurs de références font partie des éléments à inférer dans un MRP avec incertitude de référence, il faut définir une distribution de probabilités pour chaque contrainte de référence sur l'ensemble de son domaine. Une approche simpliste est de définir la distribution de probabilités directement sur l'ensemble des valeurs possibles $pk(\mathcal{I}(\mathcal{R}^j))$. Néanmoins, cela présente deux désavantages majeurs. D'une part, cela requiert un paramètre pour chaque entité du type correspondant au domaine de la contrainte de référence considérée, amenant des problèmes calculatoires et une pénurie de données pour le calcul de statistiques suffisantes pour faire de l'apprentissage robuste. D'autre part, cela empêche la conservation de la propriété de couplage faible souhaitée entre le modèle et les instances, puisque l'ensemble des valeurs possibles dépend d'une instance spécifique.

Une approche plus générique est définie dans les MRP avec incertitude de référence par l'utilisation de *fonctions de partition*. Le rôle d'une telle fonction est de fournir une connaissance résumée des tuples d'une relation, en les regroupant dans les mêmes parties d'une partition en fonction de leurs similarités deux à deux. Cela permet de conserver un couplage faible entre le MRP et une instance, tout en limitant la complexité amenée par ces distributions. Spécifier une fonction de partition peut être fait de différentes façons. La façon la plus triviale peut être de définir un cluster pour chaque valeur dans le produit cartésien des domaines des attributs des tuples considérés, regroupant ceux qui sont exactement identiques à l'égard de cette description.

Un modèle probabiliste sur les valeurs de $pk(\mathcal{I}(\mathcal{R}^j))$ pour f est donc défini en spécifiant une distribution sur les parties d'une partition, encodant la probabilité qu'une valeur de référence pour f soit prise dans une partie par rapport aux autres. Une fois le groupe choisi, choisir un tuple spécifique à l'intérieur de celui-ci est réalisé selon un tirage aléatoire suivant une loi de probabilités uniforme restreinte aux individus du groupe.

3.6 MRP avec incertitude de référence

Un MRP avec incertitude de référence (MRP-IR) est défini formellement de la façon suivante.

Définition 3.3. *[65] Étant donné un schéma de base de données $\mathbf{R} = \langle \mathcal{R}^i \rangle_{i \leq m}$, un MRP-IR $\mathcal{M} = (\mathcal{S}, \Phi, \Theta_{\mathcal{S},\Phi})$ est composé d'une structure graphique $\mathcal{S}$, d'un ensemble de fonctions de partition Φ et d'un ensemble de paramètres $\Theta_{\mathcal{S},\Phi}$. La structure graphique est composée d'un ensemble de variables aléatoires $\mathcal{V}$ et d'une fonction de parents $pa : \mathcal{V} \rightarrow \mathcal{P}(\Psi \times \Sigma(\mathbf{R}) \times \mathcal{V})$ où Ψ est un ensemble de fonctions d'agrégation (contenant la fonction d'identité* id*) et $\Sigma(\mathbf{R})$ est l'ensemble des chaînes de références que l'on peut définir sur $\mathbf{R}$.*

Pour chaque schéma de relation $\mathcal{R}^i \in \mathbf{R}$, nous définissons une variable aléatoire $A_{i,j}$ pour chaque attribut $\mathcal{A}_j \in att(\mathcal{R}^i)$ avec $dom(A_{i,j}) = ran(\mathcal{A}_j)$. De plus, pour chaque type d'association $\mathcal{R}^i_a \in \mathcal{R}_a \subseteq \mathcal{R}$, nous ajoutons une variable aléatoire de référence $R_{i,n}$ et une variable aléatoire sélecteur $S_{i,n}$ pour chaque contrainte de référence $r_n \in fk(\mathcal{R}^i_a)$ avec $ran(r_n) = \mathcal{R}^k$.

Pour chaque variable aléatoire sélecteur $S_{i,n}$, obtenue pour la contrainte de référence $r_n \in fk(\mathcal{R}^i_a)$ avec $ran(r_n) = \mathcal{R}^k$, nous définissons la fonction de partition $\phi_{i,n} : \mathcal{I}(\mathcal{R}^k) \rightarrow \{1, \ldots, c_{i,n}\}$, où $c_{i,n}$ dénote la cardinalité ou nombre de groupes de la partition, sur les tuples de $\mathcal{R}^k$ pour une instance $\mathcal{I}$. L'identité suivante doit être vérifiée : $dom(S_{i,n}) = ran(\phi_{i,n}) = \{1, \ldots, c_{i,n}\}$. La fonction de partition possède également un ensemble d'attributs $\mathcal{A}(\phi_{i,n}) \subseteq att(\mathcal{R}^k)$ qui sont utilisés pour réaliser le partitionnement.

Par souci de concision, nous dénotons par $V_{i,j}$ une variable pouvant être de la forme $A_{i,j}$, $R_{i,j}$ ou $S_{i,j}$.

La fonction pa doit respecter les mêmes contraintes que pour les MRP avec incertitude d'attributs pour définir un MRP-IR valide, c.-à-d. elle doit être telle que toute instanciation future du MRP-IR donnera lieu à un RB valide, donc acyclique, et l'image de toute variable $V_{i,j} \in \mathcal{V}$ par la fonction pa ne doit contenir que des triplets $(\psi, \sigma, V_{a,b}) \in pa(V_{i,j})$ tels que si σ est une chaîne de références non vide, alors ψ doit être une fonction d'agrégation différente de l'identité et $dom(V_{a,b}) = ran(\sigma)$; sinon, ψ doit être la fonction d'identité id et $dom(V_{a,b}) = dom(V_{i,j})$. De plus, les variables de référence sont sujettes à deux contraintes supplémentaires. Ainsi, nous avons :

$$\forall i, j : pa(R_{i,j}) = \{(id, \emptyset, S_{i,j})\},$$

indiquant qu'une variable de référence a pour unique parent sa variable sélecteur, et :

$$\forall i, j \ \nexists a, b, \sigma, \phi | (\phi, \sigma, R_{i,j}) \in pa(V_{a,b})$$

contraignant les variables de référence à ne pas avoir d'enfants dans $\mathcal{S}$.

Étant donné une structure graphique de MRP-IR $\mathcal{S}$ et un ensemble de fonctions de partitions Φ, les paramètres $\Theta_{\mathcal{S},\Phi}$ de ce modèle sont composés de distributions de probabilités conditionnelles pour chaque variable $V \in \mathcal{V}$ sachant ses parents $pa(V)$, les distributions des variables de référence étant contraintes à des lois de probabilités conditionnellement uniformes sachant un groupe, où la probabilité de choisir un individu de ce groupe est la même pour tous les individus du groupe, et 0 pour les individus d'autres groupes.

La figure 3.1 (a) montre un exemple de MRP-IR pour le schéma de base de données UMA, défini ici pour des raisons de simplicité sur 3 schémas de relation uniquement. Cet exemple contient une dépendance agrégée entre les deux types d'entités, et des dépendances intra relation faisant intervenir des noeuds sélecteurs et des noeuds attributs.

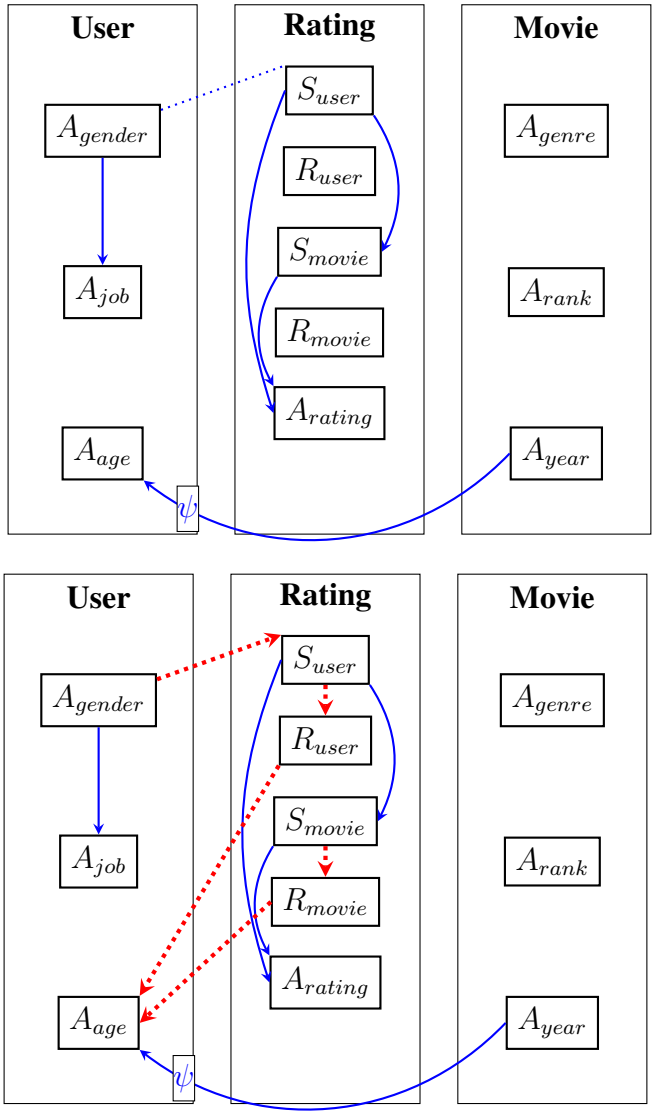


FIGURE 3.1 – (haut) Exemple de MRP-IR pour 3 schémas de relation du domaine UMA. (bas) MRP-IR complété correspondant. Le graphe décrit par ce MRP-IR est le graphe de dépendance.

3.7 MRP-IR complété et garantie d'acyclicité

La définition d'un MRP-IR entraîne l'existence automatique de dépendances induites entre les différentes variables qui le composent. Ces nouvelles dépendances sont nécessaires à l'inférence sur un jeu de données particulier et sont automatiquement ajoutées dans le RBP correspondant. L'évaluation des dépendances induites suit des règles bien précises, et il peut être intéressant de générer un MRP-IR *complété* à partir d'un MRP-IR défini par l'expert ou appris, mêlant les dépendances originales du modèle et les dépendances dérivées des règles de complétion d'un modèle.

Définition 3.4. *Soit un MRP-IR* $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$*. Nous appelons modèle complété de* $\mathcal{M}$*, le modèle* $comp(\mathcal{M}) = (comp(\mathcal{S}), \Phi, \Theta)$*, avec* $comp(\mathcal{S}) = (\mathcal{V}, comp(pa))$*, tel que :*

- $pa \subseteq comp(pa)$ *;*
- *si* $(\psi, \sigma, V_p) \in pa(V_c)$ *et* $\sigma = \langle \sigma_i \rangle_{i \leq l}$*, alors nous avons :*

$$\forall i \leq l : (id, s_i, R_{a,n}) \in comp(pa)(V_c) \mid \sigma_i = (\mathcal{R}^a, f_n) \vee \sigma_i = (\mathcal{R}^a, f_n)^{-1}$$

  où id *est l'agrégateur identité (qui n'agrège rien) et où nous avons :*

  - $s_i = (\sigma_1, \ldots, \sigma_i)$ *si* σ_i *est inversée,*
  - $s_i = (\sigma_1, \ldots, \sigma_{i-1})$ *si* σ_i *est directe ;*

- *si* $\phi_{i,f} \in \Phi$*, alors pour tout attribut de fonction de partition* $\mathcal{A}_j \in \mathcal{A}(\phi_{i,f})$*,* $(id, \emptyset, A_{a,j}) \in comp(pa)(S_{i,f})$ *où* $ref(\mathcal{R}^i, f) = \mathcal{R}^a$ *et* $\emptyset$ *est la séquence vide ;*
- $R_{i,f} \in \mathcal{V} \rightarrow (id, \emptyset, S_{i,f}) = comp(pa)(R_{i,f})$.

Un MRP-IR est dit complété ssi $comp(\mathcal{M}) = \mathcal{M}$.

Au-delà de son intérêt pour comprendre les dépendances réelles entre variables, ce MRP-IR complété peut également être utilisé pour vérifier que toute instance du schéma de base de données considéré entraînera la génération d'un RBP garanti acyclique, en vérifiant simplement l'acyclicité de sa structure graphique.

Définition 3.5. *Un MRP-IR* $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$ *est garanti acyclique si* $comp(\mathcal{S})$ *est un graphe acyclique.*

La figure 3.1 (b) montre la version complétée du MRP-IR défini dans la figure 3.1 (a). Nous pouvons observer que la dépendance agrégée entraîne l'ajout de deux dépendances pour chaque variable de référence impliquée dans la séquence de références entre les types d'entités $User$ et $Movie$. De plus, nous pouvons noter l'ajout de dépendances entre les sélecteurs et les variables de référence correspondantes, ainsi qu'entre un attribut de fonction de partition et le sélecteur correspondant. Le reste des dépendances est identique aux dépendances définies dans la figure 3.1 (a). Ce MRP-IR est garanti acyclique, puisque la structure du graphe du MRP-IR complété est acyclique.

3.8 Inférence

3.8.1 Instanciation d'un MRP-IR

L'inférence dans une instance se fait comme pour les MRP orientés attributs par la génération d'un RBP, dont les règles de génération sont les suivantes. Considérant une instance $\mathcal{I} \in inst(\mathbf{R})$ et un MRP-IR complété $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$ pour le même schéma de base de données, nous pouvons construire un RBP unique $\mathcal{G}(\mathcal{M}, \mathcal{I}) = (V_\mathcal{I}, pa_\mathcal{I}, \Theta_\mathcal{I})$ pour le squelette objet $\pi(\mathcal{I}) = \pi$.

Définition 3.6. *[62] Soit $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$, un MRP-IR complété valide et une instance $\mathcal{I} \in inst(\mathbf{R})$ compatible, le réseau bayésien à plat $\mathcal{G}(\mathcal{M}, \mathcal{I}) = (V_\mathcal{I}, pa_\mathcal{I}, \Theta_\mathcal{I})$ est défini de la façon suivante :*

1. *une variable aléatoire $a_{e,i,j} \in V_\mathcal{I}$ existe pour tout $A_{i,j}$ dans $\mathcal{M}$, avec $dom(a_{e,i,j}) = dom(A_{i,j})$ pour tout $e \in \pi(\mathcal{R}^i)$;*
2. *une variable référence $r_{e,i,f}$ existe pour tout $R_{i,f}$ dans $\mathcal{M}$, avec $dom(r_{e,i,f}) = \pi(\mathcal{R}^i)$ pour tout $e \in \pi(\mathcal{R}^i)$;*
3. *une variable sélecteur $s_{e,i,f} \in V_\mathcal{I}$ existe pour tout $S_{i,f}$ dans $\mathcal{M}$ et tout $e \in \pi(\mathcal{R}^i)$;*
4. *pour tout $V_{i,j}$ et chacun de ses triplets parents $(\psi, \sigma, V_{a,b}) \in pa(V_{i,j})$, nous avons :*
$$\forall s \in \mathcal{I}(\mathcal{R}^i) \forall t \in \pi(\mathcal{R}^a) \ : \ (\psi, \sigma, v_{t,a,b}) \in pa_\mathcal{I}(v_{s,i,j});$$
5. *une distribution de probabilités conditionnelles $P(v|pa(v)) \in \Theta_\mathcal{I}$ existe pour tout $v \in V_\mathcal{I}$ avec $P(v|pa(v)) = P(V|pa(V))$ pour la variable aléatoire correspondante dans $\mathcal{M}$.*

Un RBP est l'instanciation d'un MRP-IR $\mathcal{M}$ qui décrit une loi jointe de distribution unique sur les individus de $\mathcal{I}$ utilisant les dépendances probabilistes définies dans $\mathcal{M}$. Nous pouvons alors calculer la vraisemblance d'une instance, sachant son squelette objet et le modèle :

$$P(\mathcal{I}|\mathcal{M}, \pi) = \prod_{\mathcal{R}^i \in \mathbf{R}} \prod_{\mathcal{A}_j \in att(\mathcal{R}^i)} \prod_{e \in \mathcal{I}(\mathcal{R}^i)} P(a_{e,i,j}|pa(a_{e,i,j}))$$
$$\prod_{\substack{f \in fk(\mathcal{R}^i), \\ ran(f) = \mathcal{R}^k}} P(s_{e,i,f}|pa(s_{e,i,f})) \cdot P(r_{e,i,f}|pa(r_{e,i,f})). \tag{3.1}$$

Un exemple de RBP pour le MRP-IR défini dans la figure 3.1 est proposé dans la figure 3.2 pour un jeu de données composé de 2 tuples de chaque type d'entité et chaque type d'association. Nous pouvons observer que pour une dépendance donnée dans le MRP-IR, tous les tuples des types d'entité ou d'associations impliqués par cette dépendance voient leurs variables aléatoires dépendantes deux à deux. Ceci est également vrai pour les dépendances induites. Ces dépendances peuvent être interprétées de la façon suivante. Puisque nous ne connaissons pas à l'avance les valeurs de référence entre les différents tuples, l'inférence impliquant des variables dépendantes doit prendre en compte tous les tuples potentiellement impliqués dans un chemin allant de la variable parent à la variable enfant de cette dépendance.

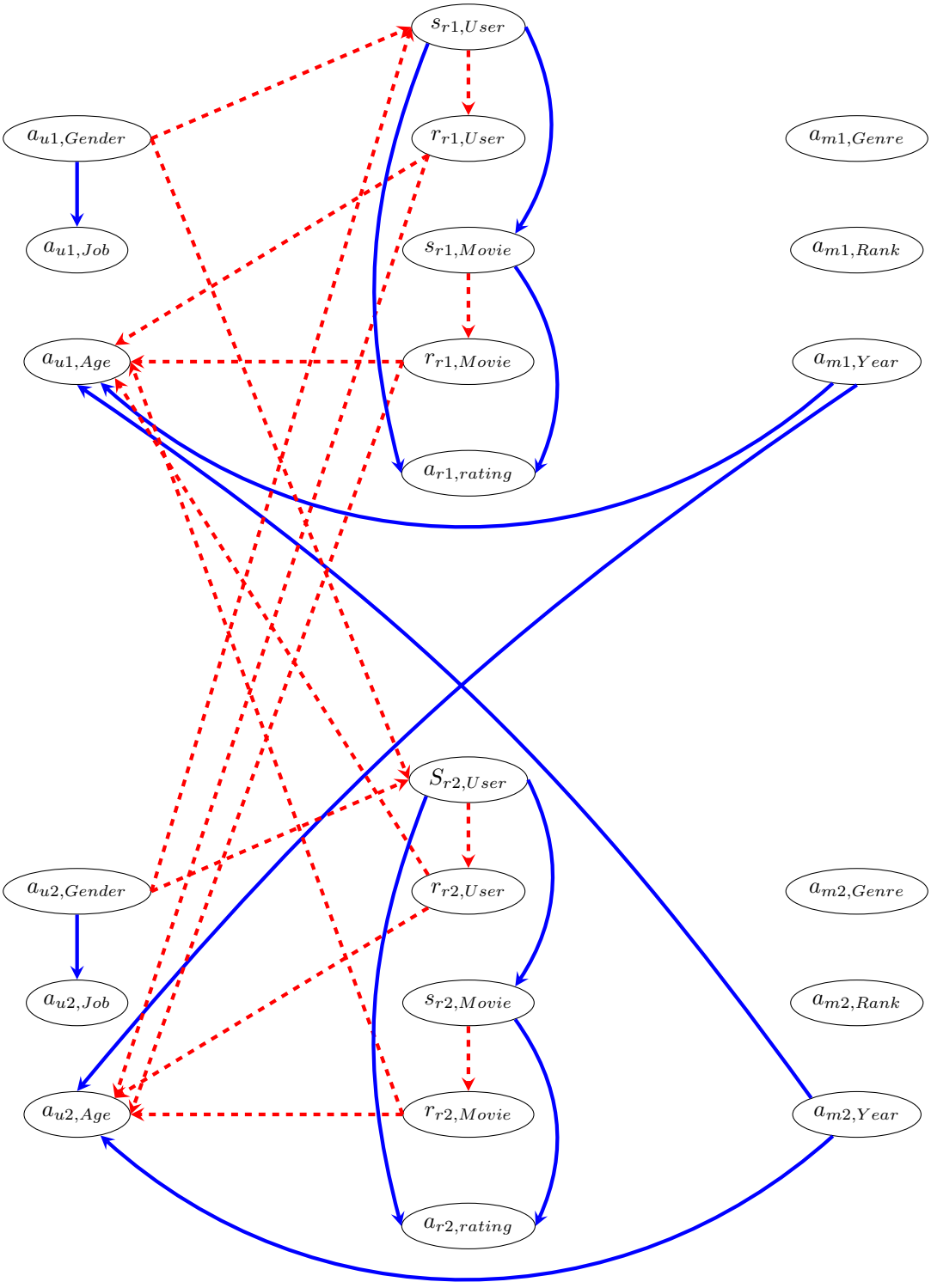


FIGURE 3.2 – Exemple de RBP pour le MRP-IR défini sur UM.

	$\psi(A_{Movie,Year})$		
	?	$y-$	$y+$
$A_{User,Age} = a-$	0.5	0.4	0.3
$A_{User,Age} = a+$	0.5	0.6	0.7

(a)

		a =				**a** =	
b	ψ	$a-$	$a+$	**b**	ψ	$a-$	$a+$
$\mathbf{y-}$ $y-$ $\mathbf{m1}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $y-$ $\mathbf{m1}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	$y+$	0.3	0.7
$\mathbf{y-}$ $y-$ $\mathbf{m1}$ $m1$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6	$\mathbf{y+}$ $y-$ $\mathbf{m1}$ $m1$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7
$\mathbf{y-}$ $y-$ $m1$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $y-$ $m1$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $y-$ $m1$ $m1$ $u2$ $u2$	?	0.5	0.5	$y+$ $y-$ $m1$ $m1$ $u2$ $u2$	?	0.5	0.5
$\mathbf{y-}$ $\mathbf{y-}$ $\mathbf{m1}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $\mathbf{y-}$ $\mathbf{m1}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	?	0.5	0.5
$\mathbf{y-}$ $y-$ $\mathbf{m1}$ $m2$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6	$\mathbf{y+}$ $y-$ $\mathbf{m1}$ $m2$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7
$y-$ $\mathbf{y-}$ $m1$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6	$y+$ $\mathbf{y-}$ $m1$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6
$y-$ $y-$ $m1$ $m2$ $u2$ $u2$	?	0.5	0.5	$y+$ $y-$ $m1$ $m2$ $u2$ $u2$	?	0.5	0.5
$\mathbf{y-}$ $\mathbf{y-}$ $\mathbf{m2}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $\mathbf{y-}$ $\mathbf{m2}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	?	0.5	0.5
$y-$ $\mathbf{y-}$ $\mathbf{m2}$ $m1$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6	$y+$ $\mathbf{y-}$ $\mathbf{m2}$ $m1$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6
$\mathbf{y-}$ $y-$ $m2$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $y-$ $m2$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $y-$ $m2$ $m1$ $u2$ $u2$	?	0.5	0.5	$y+$ $y-$ $m2$ $m1$ $u2$ $u2$	?	0.5	0.5
$y-$ $\mathbf{y-}$ $\mathbf{m2}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	$y-$	0.4	0.6	$y+$ $\mathbf{y-}$ $\mathbf{m2}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	$y-$	0.4	0.6
$y-$ $\mathbf{y-}$ $\mathbf{m2}$ $m2$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6	$y+$ $\mathbf{y-}$ $\mathbf{m2}$ $m2$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6
$y-$ $\mathbf{y-}$ $m2$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6	$y+$ $\mathbf{y-}$ $m2$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6
$y-$ $y-$ $m2$ $m2$ $u2$ $u2$	?	0.5	0.5	$y+$ $y-$ $m2$ $m2$ $u2$ $u2$	?	0.5	0.5
$\mathbf{y-}$ $\mathbf{y+}$ $\mathbf{m1}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $\mathbf{y+}$ $\mathbf{m1}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	$y+$	0.3	0.7
$\mathbf{y-}$ $y+$ $\mathbf{m1}$ $m1$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6	$\mathbf{y+}$ $y+$ $\mathbf{m1}$ $m1$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7
$\mathbf{y-}$ $y+$ $m1$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $y+$ $m1$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $y+$ $m1$ $m1$ $u2$ $u2$	?	0.5	0.5	$y+$ $y+$ $m1$ $m1$ $u2$ $u2$	?	0.5	0.5
$\mathbf{y-}$ $\mathbf{y+}$ $\mathbf{m1}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	?	0.5	0.5	$\mathbf{y+}$ $\mathbf{y+}$ $\mathbf{m1}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	$y+$	0.3	0.7
$\mathbf{y-}$ $y+$ $\mathbf{m1}$ $m2$ $\mathbf{u1}$ $u2$	$y-$	0.4	0.6	$\mathbf{y+}$ $y+$ $\mathbf{m1}$ $m2$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7
$y-$ $\mathbf{y+}$ $m1$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7	$y+$ $\mathbf{y+}$ $m1$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $y+$ $m1$ $m2$ $u2$ $u2$	?	0.5	0.5	$y+$ $y+$ $m1$ $m2$ $u2$ $u2$	?	0.5	0.5
$\mathbf{y-}$ $\mathbf{y+}$ $\mathbf{m2}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	?	0.5	0.5	$\mathbf{y+}$ $\mathbf{y+}$ $\mathbf{m2}$ $\mathbf{m1}$ $\mathbf{u1}$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $\mathbf{y+}$ $\mathbf{m2}$ $m1$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7	$y+$ $\mathbf{y+}$ $\mathbf{m2}$ $m1$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7
$\mathbf{y-}$ $y+$ $m2$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y-$	0.4	0.6	$\mathbf{y+}$ $y+$ $m2$ $\mathbf{m1}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $y+$ $m2$ $m1$ $u2$ $u2$	?	0.5	0.5	$y+$ $y+$ $m2$ $m1$ $u2$ $u2$	?	0.5	0.5
$y-$ $\mathbf{y+}$ $\mathbf{m2}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	$y+$	0.3	0.7	$y+$ $\mathbf{y+}$ $\mathbf{m2}$ $\mathbf{m2}$ $\mathbf{u1}$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $\mathbf{y+}$ $\mathbf{m2}$ $m2$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7	$y+$ $\mathbf{y+}$ $\mathbf{m2}$ $m2$ $\mathbf{u1}$ $u2$	$y+$	0.3	0.7
$y-$ $\mathbf{y+}$ $m2$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7	$y+$ $\mathbf{y+}$ $m2$ $\mathbf{m2}$ $u2$ $\mathbf{u1}$	$y+$	0.3	0.7
$y-$ $y+$ $m2$ $m2$ $u2$ $u2$	?	0.5	0.5	$y+$ $y+$ $m2$ $m2$ $u2$ $u2$	?	0.5	0.5

(b)

TABLE 3.2 – (a) Exemple de table de probabilités conditionnelles pour $A_{User,Age}$ dans le MRP-IR de la figure 3.1. (b) Table de probabilité $P(A_{u1,User,Age} = \mathbf{a}|\langle A_{m1,Movie,Year}, A_{m2,Movie,Year}, R_{r1,Rating,Movie}, R_{r2,Rating,Movie}, R_{r1,Rating,User}, R_{r2,Rating,User}\rangle = \mathbf{b})$ dans le RBP de la figure 3.2. La valeur de l'agrégation ψ dépend des valeurs effectives des références des associations $r1$ et $r2$. Une fois les valeurs de référence résolues et la valeur d'agrégation connue, la distribution de probabilités correspond à celle définie en (a) pour cette valeur de ψ. Les valeurs prises en compte pour l'agrégation ainsi que les variables intermédiaires correspondantes sont mises en gras.

3.8.2 Complétion et distributions de probabilité d'un RBP

La complétion d'un MRP-IR mène à l'ajout de dépendances probabilistes de façon automatique et ceci impacte la façon dont les distributions de probabilités originales sont utilisées pour l'inférence dans un RBP. Nous explicitons ici l'impact des nouvelles dépendances sur les distributions de probabilités originalement définies et sur la façon de calculer une probabilité pour réaliser de l'inférence dans un RBP particulier. Pour des raisons de clarté du discours, nous séparons chaque type de dépendance induite dans les sections suivantes.

i Dépendances impliquant des séquences de références non nulles

Étant donné un MRP-IR $\mathcal{M} = (\mathcal{S}, \Phi, \Theta_{\mathcal{S},\Phi})$ et une instance $\mathcal{I} \in inst(\mathbf{R})$, et considérant une distribution de probabilités $P(V_{i,j}|pa(V_{i,j}))$ avec $|pa(V_{i,j})| = p$ et $pa(V_{i,j}) = \{(\psi_k, \sigma_k, V_k)\}_{k\leq p}$. Alors, dans le MRP-IR complété $comp(\mathcal{M})$, nous avons nécessairement une distribution complétée de la forme :

$$P(V_{i,j} \mid \{\ \{(\psi_k, \sigma_k, V_k)\} \cup R_k\ \}_{k\leq p}),$$

où :

$$R_k = \{\ R_{a,b} \mid \exists u : \sigma_k[u;u] \in \{(\mathcal{R}^a, b), (\mathcal{R}^a, b)^{-1}\}\ \}.$$

Alors, pour une instance $\mathcal{I}$, nous avons un ensemble de distributions de probabilités :

$$\bigcup_{\forall t \in \mathcal{I}(\mathcal{R}^i)} \left\{ P\left(v_{t,i,j} \mid \{\ v_k \cup r_k\ \}_{k\leq p}\right)\right\} \tag{3.2}$$

où :

$$v_k = \{v_{s,a,b} \mid V_k = V_{a,b}, s \in \mathcal{I}(\mathcal{R}^a)\} \tag{3.3}$$

et :

$$r_k = \{r_{s,a,b} \mid R_{a,b} \in R_k, s \in \mathcal{I}(\mathcal{R}^a)\} \tag{3.4}$$

Calculer une distribution de probabilités spécifique :

$$P(v_{t,i,j} \mid \{\ v_k \cup r_k\ \}_{k\leq p})$$

peut alors être fait en deux étapes principales :

1. résoudre l'ensemble de parents réel :

   $$pa(v_{t,i,j}) = \{\ pa_k(v_{t,i,j})\ \}_{k\leq p},$$

   chaque $pa_k(v_{t,i,j})$ étant résoluble de façon séparée en utilisant l'Algorithme 2. Le but de ce dernier est de trouver pour chaque variable $v_{t,i,j}$, de tout tuple t de $\mathcal{R}^i$, et considérant des valeurs pour toutes les variables de référence impliquées, l'ensemble des variables atteintes en parcourant le RBP selon la chaîne de références σ_k et utilisant les valeurs de référence des tuples rencontrés à chaque étape ;

2. calculer la distribution de probabilités finale :

   $$P(v_{t,i,j} \mid \{\ \psi_k(pa_k(v_{t,i,j}))\ \}_{k\leq p})$$

   suivant la distribution originale $P(V_{i,j} \mid pa(V_{i,j}))$.

Algorithme 2 Algorithme de résolution de parents dans un RBP

Paramètres : $v_{t,i,j}$ une variable d'un RBP sur une instance $\mathcal{I}$,
σ_k une séquence de références, V_k une variable du MRP-IR
Retourne : pa_k un ensemble de parents de $v_{t,i,j}$ suivant σ_k

$pa_{temp} \leftarrow \{t\}$
$l \leftarrow |\sigma_k|$
$\mathcal{R}^p \leftarrow \mathcal{R}^i$
Pour $u \in [1, \ldots, l]$ **Faire**
  $\mathcal{R}^r \leftarrow ran(\sigma_k[u;u])$
  Si $\sigma_k[u;u] = (\mathcal{R}^p, b)$ **Alors**
    $pa_{temp} \leftarrow \{s \mid s \in \mathcal{I}(\mathcal{R}^r), \exists t \in pa_{temp} : t[b] = s[pk(\mathcal{R}^p)]\}$
  Sinon Si $\sigma_k[u;u] = (\mathcal{R}^r, b)^{-1}$ **Alors**
    $pa_{temp} \leftarrow \{s \mid s \in \mathcal{I}(\mathcal{R}^r), \exists t \in pa_{temp} : s[b] = t[pk(\mathcal{R}^p)]\}$
  Fin Si
  $\mathcal{R}^p \leftarrow \mathcal{R}^r$
Fin Pour
Retourner $pa_k = \{v_{s,a,b} \mid s \in pa_{temp}, V_k = V_{a,b}\}$

Pour se donner une idée plus précise de la sémantique des dépendances induites, la table 3.2 (a) donne un exemple de table de probabilités pour la variable $A_{User,Age}$ dans le MRP-IR et la table 3.2 (b) donne la table induite dans le RBP défini dans la figure 3.2. Nous pouvons voir que pour connaître la distribution de probabilités de la variable $A_{u1,User,Age}$, nous devons d'abord résoudre les valeurs de référence des tuples $r1$ et $r2$. En fonction de ces valeurs, il est alors possible de connaître la valeur d'agrégation $\psi(pa(A_{u1,User,Age})) = \psi$ qui induit enfin la distribution de probabilités de $A_{u1,User,Age}$ en utilisant la distribution définie pour $A_{User,Age}$.

Plus précisément, en considérant la dépendance probabiliste agrégée étudiée dans l'exemple précédent, nous pouvons commencer à exhiber une relation entre les parties d'une séquence de références et la méthode pour retrouver les probabilités correctes. Cette distribution de probabilités s'écrit :

$$P(A_{User,Age}|\{(\psi, \langle (Rating, User)^{-1}, (Rating, Movie)\rangle, A_{Movie,Year})\})$$

La séquence de références est ici $\langle (Rating, User)^{-1}, (Rating, Movie)\rangle$ et la variable parent est $A_{Movie,Year}$. La séquence de références est composée de deux parties. La première, $(Rating, User)^{-1}$ est inversée. Intéressons-nous à la contrainte de référence $(Rating, User)^{-1}$ qui est l'inverse de la contrainte de référence $(Rating, User)$. Nous avons :

- $dom((Rating, User)^{-1}) = ran((Rating, User)) = User$;
- $ran((Rating, User)^{-1}) = dom((Rating, User)) = Rating$.

Dans ce cas où une partie de séquence de références est inversée, nous pouvons conclure que :

- la variable de référence $R_{Rating,User}$ est à ajouter aux parents de la variable $A_{User,Age}$ dans le MRP-IR complété ;
- la première étape de la méthode pour retrouver les probabilités dans un RBP particulier pour un enfant $A_{ui,User,Age}$ particulier sera de trouver toutes les associations instanciant $Rating$ dont la valeur de $User$ est égale à ui et de stocker

ces individus dans une liste intermédiaire préalable au calcul de l'agrégation ψ pour ui.

Focalisons-nous maintenant sur la seconde partie de la séquence de références. Sa valeur est $(Rating, Movie)$. Pour continuer la découverte des parents d'un noeud $A_{ui,User,Age}$ dans un RBP, il faut reprendre à partir de la liste des objets intermédiaires obtenus jusqu'à la partie de séquence de références précédente. Nous avons :

- $dom(ran(\langle (Rating, User)^{-1}, (Rating, Movie)\rangle)) = Rating$;
- $ran(ran(\langle (Rating, User)^{-1}, (Rating, Movie)\rangle)) = Movie.$

Dans ce cas où la partie de séquence de références est directe, nous pouvons conclure que :

- la variable de référence $R_{Rating,Movie}$ est à ajouter aux parents de la variable $A_{User,Age}$ dans le MRP-IR complété ;
- la seconde étape de la méthode pour retrouver les probabilités dans un RBP particulier pour un enfant $A_{ui,User,Age}$ particulier sera de trouver toutes les entités instanciant $Movie$ référencées par au moins une association $Rating$ dans la liste intermédiaire précédemment construite. Cet ensemble constituera alors la nouvelle liste intermédiaire d'objets pour le calcul final de l'agrégation.

Une fois la séquence de références parcourue dans son intégralité, l'agrégation peut être obtenue sur les variables des entités encore présentes dans la liste intermédiaire à cette étape et la distribution de probabilité associée peut être retrouvée à partir du paramètre défini dans le MRP-IR pour cette variable.

ii Ajout de dépendances entre variables clusters et variables références

Considérons enfin une variable de référence $R_{i,j}$ avec $ref(\mathcal{R}^i, j) = \mathcal{R}^u$ dans le MRP-IR. Sans prendre en compte les autres types de parents, nous avons alors une distribution de la forme $P(R_{i,j})$. Nous avons donc dans le MRP-IR complété une distribution de la forme :

$$P(\ R_{i,j} \mid \{(id, \emptyset, S_{i,j})\}\ ).$$

Aussi, le RBP pour une instance donnée doit inclure les distributions de probabilité suivantes :

$$\bigcup_{\forall t \in \mathcal{I}(\mathcal{R}^i)} \{\ P(r_{t,i,j} \mid \{(id, \emptyset, s_{t,i,j})\})\ \}.$$

Ces distributions de probabilité sont très simples et consistent en des lois uniformes parmi les individus du cluster choisi, et une probabilité de 0 pour les individus n'en faisant pas partie.

3.9 Apprentissage

3.9.1 Paramètres

L'apprentissage de paramètres pour un MRP-IR suit les mêmes règles que pour un MRP orienté attributs. Une fois la structure de dépendances et les fonctions de partition connues, il est possible de réaliser une estimation des paramètres à partir des

statistiques calculées sur les différents tuples d'un jeu de données. Pour les variables sélecteur, n'existant pas réellement dans le jeu de données, il suffit de les considérer comme de nouveaux attributs des schémas de relation concernés et d'attribuer à chaque tuple la valeur correspondant à son groupe d'appartenance à l'égard de la fonction de partition. Une fois cette étape intermédiaire résolue, l'estimation de paramètres peut encore une fois se faire par maximum de vraisemblance, ou par une méthode bayésienne permettant de limiter les risques de sur apprentissage.

À titre d'exemple, l'estimation des paramètres pour la variable $S_{Rating,Movie}$ du MRP-IR défini dans la Figure 3.1 selon le maximum de vraisemblance, et utilisant un jeu d'apprentissage $\mathcal{I}$ est obtenue de la façon suivante :

$$P(S_{Rating,Movie} = u | S_{Rating,User} = v : \mathcal{I}) =$$

$$\frac{\mid \{t \in \mathcal{I}(\mathcal{R}^{Rating}) \mid \phi_{Rating,Movie}(t[Movie]) = u \wedge \phi_{Rating,User}(t[User]) = v\} \mid}{\mid \{t \in \mathcal{I}(\mathcal{R}^{Rating}) \mid \phi_{Rating,User}(t[User]) = v\} \mid}.$$

Une exception existe toutefois dans les MRP-IR. Les variables de référence ne sont en effet pas sujettes à l'estimation de paramètres conventionnelle pour des raisons de généralisation des distributions et de performances, menant d'ailleurs à la nécessité des sélecteurs. À la place, les distributions des variables de référence sont définies de façon déterministe de la façon suivante. Pour chaque tuple d'une relation instanciant le schéma $\mathcal{R}^i$ ayant une contrainte de référence f, et considérant une fonction de partition $\phi_{i,f}$ pour $ref(\mathcal{R}^i, f) = \mathcal{R}^j$ nous avons :

$$\forall t \in \mathcal{I}(\mathcal{R}^j) : \phi_{i,f}(t) = c_k \rightarrow P(R_{i,f} = t | S_{i,f} = c_k) = \frac{1}{|\phi_{i,f}^{-1}(c_k)|}$$
$$\forall t \in \mathcal{I}(\mathcal{R}^j) : \phi_{i,f}(t) = c_k \rightarrow \forall c_a \neq c_k P(R_{i,f} = t | S_{i,f} = c_a) = 0$$

où $|\phi_{i,f}^{-1}(c_k)|$ est le nombre d'individus dans le groupe c_k. Le choix d'un tuple particulier dans les MRP-IR peut ainsi être vu comme un processus en deux temps. D'abord, choisir un groupe dans la partition en fonction de la distribution associée. Ensuite, choisir un tuple uniformément dans ce groupe.

3.9.2 Fonctions de partition

La structure d'un MRP-IR est composée de ses fonctions de partition et de sa structure graphique. Dans un MRP-IR, les fonctions de partition sont associées à des attributs du schéma de relation correspondant aux individus à regrouper. Ces attributs définissent alors complètement la partition en ce sens que deux individus possédant les mêmes valeurs pour les attributs d'une fonction de partition font obligatoirement partie du même groupe. Plus formellement, cela signifie que nous avons pour tout $\phi_{i,f}$:

$$\phi_{i,f} : dom(\mathcal{A}(\phi_{i,f})) \rightarrow \{1, \ldots, c_{i,f}\}.$$

Considérant une stratégie de partitionnement à partir d'un ensemble d'attributs, apprendre une fonction de partition revient donc à apprendre son ensemble d'attributs optimal.

L'apprentissage des attributs de fonction de partition se fait de façon gloutonne, en même temps que l'apprentissage des dépendances entre variables tel qu'énoncé

précédemment pour les MRP orientés attributs, par l'utilisation de deux opérateurs spécifiques : *refine* a pour but de préciser les contours d'une fonction de partition, en ajoutant des groupes ; *abstract* a pour but de simplifier une fonction de partition, en supprimant des groupes. En théorie, de nombreuses stratégies pourraient être associées à ces opérateurs, certaines étant suggérées dans [70], comme l'apprentissage d'un arbre de décision associant respectivement les opérateurs de raffinage et d'abstraction à l'ajout ou la suppression de noeuds dans l'arbre. En pratique, la littérature sur le sujet se focalise sur une description simple des opérateurs, ramenant ceux-ci à l'ajout ou la suppression d'un attribut dans l'ensemble des attributs de fonction de partition courants, pour ensuite calculer une nouvelle fonction de partition consistant en la création d'un groupe par élément du produit cartésien des valeurs des attributs.

La figure 3.3 illustre l'usage des opérateurs de manipulation des attributs de fonction de partition.

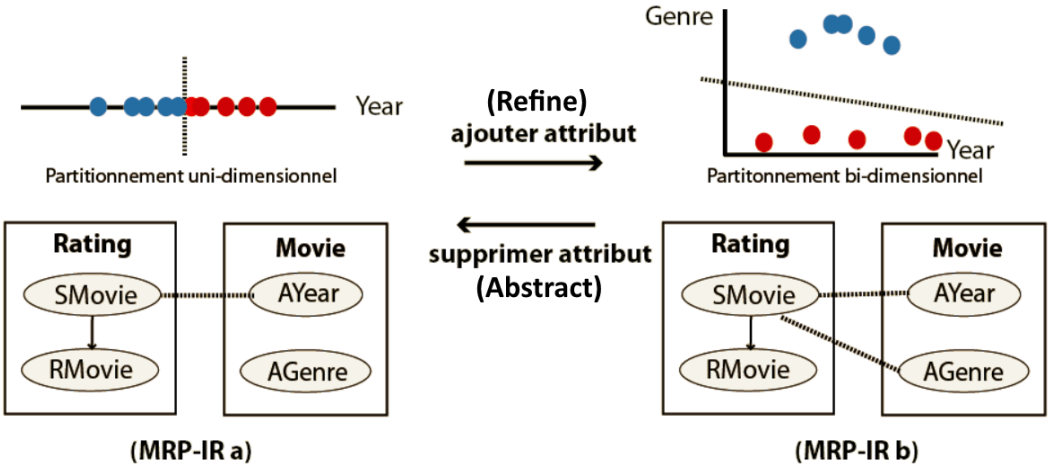


FIGURE 3.3 – Illustration de l'utilisation des opérateurs *refine* et *abstract* pour une unique fonction de partition.

Comme pour les MRP orientés attributs, l'apprentissage de modèle est guidé par la maximisation du score de log-vraisemblance du jeu de données d'apprentissage et ce score est toujours décomposable localement pour un MRP-IR. Toutefois, modifier une fonction de partition a un impact plus important sur l'évaluation d'un modèle que l'usage des opérateurs de voisinage dans un MRP orienté attributs. En effet, alors que précédemment, la mise en cache d'un score était conditionnée par la paire constituée de la variable enfant dans le MRP et l'ensemble des parents, la mise en cache pour un MRP-IR est également conditionnée sur les ensembles d'attributs de fonction de partition des sélecteurs, puisque les modifier pour une contrainte de référence donnée entraîne la modification de la variable aléatoire sélecteur associée. De plus, la modification d'une fonction de partition par le biais de ses attributs entraîne la nécessité de recalculer les scores locaux des enfants de la variable sélecteur associée pour les mêmes raisons.

La figure 3.4 montre un exemple de structure de MRP-IR et l'impact de la modification d'une fonction de partition durant l'apprentissage sur l'évaluation du score. Ce dernier peut toujours être calculé de façon incrémentale, mais le nombre de noeuds à reconsidérer est plus important que pour un apprentissage de MRP orienté attributs.

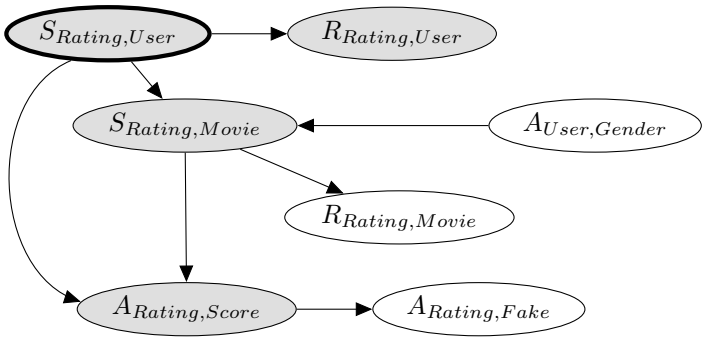


FIGURE 3.4 – Extrait de MRP-IR et visualisation de l'impact sur le calcul du score d'une modification de la fonction de partition $\phi_{Rating,User}$ (sélecteur entouré de noir). Les noeuds au fond coloré correspondent à ceux dont le score local doit être recalculé après mise à jour.

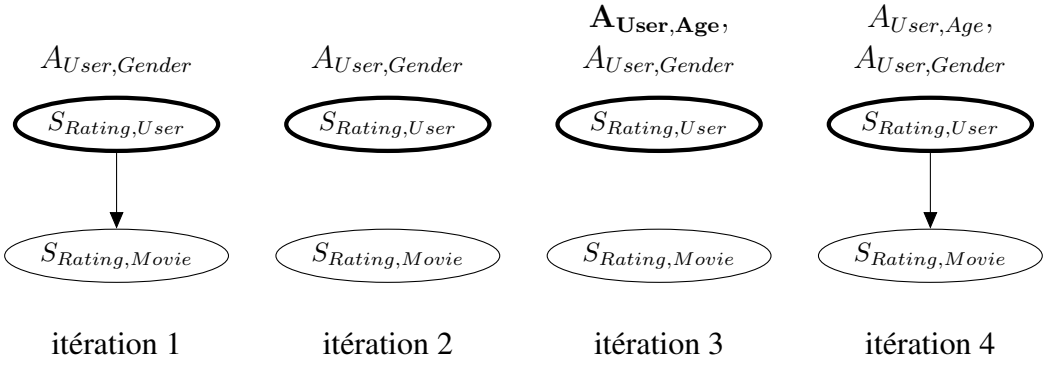


FIGURE 3.5 – Séquence d'opérations sur un MRP-IR. À la dernière itération, le retour à la structure graphique de l'itération 1 n'empêche pas le besoin de recalculer le score local de , $S_{Rating,Movie}$ car l'ensemble des attributs de fonction de partition de son parent $S_{Rating,User}$ a évolué.

3.9.3 Dépendances probabilistes

L'apprentissage des dépendances probabilistes est sensiblement identique à l'apprentissage des dépendances probabilistes pour les MRP orientés attributs. L'approche par recherche gloutonne passe par l'utilisation de trois opérateurs permettant respectivement l'ajout, la suppression ou l'inversion d'une dépendance. Dans le cas des MRP-IR, ces opérateurs existent toujours, mais sont utilisés conjointement aux opérateurs de fonction de partition pour déterminer l'ensemble des voisins d'un modèle courant à une itération donnée. Au-delà de l'impact immédiat de l'usage d'un opérateur de fonction de partition sur la mise à jour du score tel que décrit précédemment, les opérateurs de modification de la structure du graphe subissent également un impact similaire. En effet, puisque la mise en cache des scores est désormais conditionnée par les ensembles d'attributs de fonction de partition, toute modification de l'ensemble des parents d'une variable, lorsque cet ensemble impliquait (resp. impliquera), avant (resp. après) modification, des variables sélecteurs, doit considérer les attributs des fonctions de partition associés à ceux-ci afin de décider de l'utilisation d'une valeur mise en cache ou d'un nouveau calcul de score local.

La figure 3.5 illustre le besoin de recalculer un score local même dans le cas où la structure graphique retrouve un état précédemment rencontré.

3.9.4 Évaluation des modèles

Formellement, nous voulons trouver le modèle $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$ maximisant $P(\mathcal{S}, \Phi|\mathcal{I}, \pi) \propto P(\mathcal{I}|\mathcal{S}, \Phi, \pi)P(\mathcal{S}, \Phi|\pi)$. Supposant que la structure du graphe est indépendante des fonctions de partition, nous pouvons écrire :

$$P(\mathcal{S}, \Phi|\mathcal{I}, \pi) \propto P(\mathcal{I}|\mathcal{S}, \Phi, \pi)P(\mathcal{S}|\pi)P(\Phi|\pi). \tag{3.5}$$

Le premier terme de (3.5), $P(\mathcal{I}|\mathcal{S}, \Phi, \pi)$, est la vraisemblance marginale de la distribution jointe sur tous les paramètres de la structure correspondante :

$$P(\mathcal{I}|\mathcal{S}, \Phi, \pi) = \int_{\Theta} P(\mathcal{I}|\Theta, \mathcal{S}, \Phi, \pi)P(\Theta|\mathcal{S}, \Phi, \pi)d\Theta$$

où $P(\mathcal{I}|\Theta, \mathcal{S}, \Phi, \pi)$ est défini dans (3.1), et $P(\Theta|\mathcal{S}, \Phi, \pi)$ est l'*a priori* sur les paramètres. Dans le but d'obtenir un score décomposable, nous forçons les *a priori* à satisfaire les propriétés d'indépendance globale et de modularité des paramètres. C'est le cas des *a priori* de Dirichlet. Dans ce cas, $P(\mathcal{I}|\mathcal{S}, \Phi, \pi)$ a une forme close :

$$P(\mathcal{I}|\mathcal{S}, \Phi, \pi) = \prod_{\mathcal{R}^a \in \mathbf{R}} \prod_{V_i \in \mathcal{V}(\mathcal{R}^a)} \tag{3.6}$$

$$\prod_{\mathbf{u}_j \in dom(pa(V_{a,i}))} \frac{\Gamma(\alpha_{aij})}{\Gamma(\alpha_{aij} + N_{aij})} \prod_{v_k \in dom(V_{a,i})} \frac{\Gamma(\alpha_{aijk} + N_{aijk})}{\Gamma(\alpha_{aijk})} \tag{3.7}$$

où N_{aijk} représente le nombre d'entités de $\mathcal{I}(\mathcal{R}^a)$ ayant la k^{eme} valeur pour la variable $V_{a,i}$ dans $dom(V_{a,i})$ et ayant une valeur de parents égale à la j^{eme} valeur pour $pa(V_{a,i})$ dans $dom(pa(V_{a,i}))$, α_{aijk} sont les valeurs des hyper paramètres de Dirichlet correspondant, $N_{aij} = \sum_k N_{aijk}$, $\alpha_{aij} = \sum_k \alpha_{aijk}$, et Γ est la *fonction gamma*. Il est important de noter que (3.7) implique pour chaque variable $V_{a,i}$ un nombre de termes égal à $|dom(V_{a,i}) \times dom(pa(V_{a,i}))| + |dom(V_{a,i})|$. Pour les variables de référence, cela signifie en principe que le nombre de termes est potentiellement très grand puisqu'il dépend du produit du nombre de tuples du schéma référencé et du nombre de groupes de la fonction de partition. Toutefois, le calcul se simplifie pour ces variables, grâce à l'uniformité de leur distribution. En effet, pour toute variable de référence $R_{a,i}$, nous avons :

$$P(\mathcal{I}(R_{a,i})|\mathcal{S}, \Phi, \pi) = \prod_{u_j \in dom(S_{a,i})} \frac{\Gamma(\alpha_{aij})}{\Gamma(\alpha_{aij} + N_{aij})} \prod_{v_k \in pk(\mathcal{I}(\mathcal{R}^a))} \frac{\Gamma(\alpha_{aijk} + N_{aijk})}{\Gamma(\alpha_{aijk})}. \tag{3.8}$$

Or, nous savons que la probabilité de choisir un individu d'un groupe déterminé est uniforme sur les individus de ce groupe et nulle pour les autres individus. De ce fait, si $\phi_{a,i}(k) = j \in \{1, \ldots, c_{i,f}\}$, alors $N_{aijk'} = 0$ si $k' \not= k$ et N_{aijk} est en fait égal à $N_{ai.k}$ qui est le nombre de fois où l'individu k est référencé dans $\mathcal{I}(\mathcal{R}^a)[i]$, cette valeur ne dépendant pas du tout de j. Nous pouvons ainsi réécrire (3.7) de la façon suivante pour les MRP-IR :

$$P(\mathcal{I}|\mathcal{S}, \Phi, \pi) =$$

$$\prod_{\mathcal{R}^a \in \mathbf{R}} \prod_{V_{a,i} \in fk(\mathcal{R}^a)} \prod_{\mathbf{u}_j \in dom(pa(V_{a,i}))} \frac{\Gamma(\alpha_{aij})}{\Gamma(\alpha_{aij} + N_{aij})} \prod_{v_k \in dom(V_{a,i})} \frac{\Gamma(\alpha_{aijk} + N_{aijk})}{\Gamma(\alpha_{aijk})}$$

$$\prod_{R_{a,i} \in fk(\mathcal{R}^a)} \prod_{u_j \in dom(S_{a,i})} \frac{\Gamma(\alpha_{aij})}{\Gamma(\alpha_{aij} + N_{aij})} \prod_{v_k \in pk(\mathcal{I}(\mathcal{R}^a))} \frac{\Gamma(\alpha_{ai.k} + N_{ai.k})}{\Gamma(\alpha_{ai.k})}$$

où $\bar{fk}(\mathcal{R}^a)$ est l'ensemble des variables aléatoires du MRP-IR liées à $\mathcal{R}^a$ qui ne sont pas des variables de référence, c.-à-d. l'ensemble des variables d'attributs et de sélecteurs. Cette simplification réduit le nombre de termes pour les variables de référence à $|dom(R_{a,i})| + |dom(S_{a,i})|$.

Le second terme de (3.5), $P(\mathcal{S}|\pi)$, est un *a priori* sur les structures de graphes, étant donné le squelette objet de l'instance. Supposant que la structure de graphe ne dépend pas du squelette, nous pouvons réécrire l'*a priori* $P(\mathcal{S})$. Cet *a priori* est choisi de façon à pénaliser les dépendances probabilistes impliquant de longues séquences de référence. Plus précisément, nous définissons l'*a priori* sur la structure graphique de telle manière que $log(P(\mathcal{S})) = -K_r$, où K_r dénote la somme totale des longueurs des séquences de référence des dépendances du MRP-IR.

Le troisième terme de (3.5), $P(\Phi|\pi)$, est un *a priori* sur les fonctions de partition, étant donné le squelette objet de l'instance. Celui-ci doit être choisi de façon à pénaliser les partitions avec de nombreuses parties, ce qui satisfait également la contrainte de modularité. Intuitivement, nous souhaitons que le nombre de groupes soit petit relativement au nombre d'entités à partitionner, conformément aux objectifs de généralisation à l'origine de l'introduction des sélecteurs.

La littérature sur les MRP-IR ne fait étrangement pas état d'un *a priori* sur les fonctions de partition. Nous proposons néanmoins d'en utiliser pour l'évaluation de nos travaux et considérons diverses solutions éprouvées dans des travaux dédiés, telles que l'utilisation de l'*a priori* d'Ewens-Pitman [129, 29] :

$$P(\Phi|\pi) = \prod_{\phi_{i,n}\in\Phi} \frac{\Gamma(\lambda_{\phi_{i,n}})}{\Gamma(N_i + \lambda_{\phi_{i,n}})} (\lambda_{\phi_{i,n}})^{c_{i,n}} \prod_{j=1}^{c_{i,n}} \Gamma(|\phi_{i,n}[j]|)$$

où N_i est le nombre d'entités de $\mathcal{I}(\mathcal{R}^i)$, $c_{i,n}$ est le nombre de clusters de $\phi_{i,n}$ et $|\phi_{i,n}^{-1}(j)|$ est le nombre d'individus dans le cluster j de $\phi_{i,n}$. Cet *a priori* a été utilisé à de nombreuses reprises dans la littérature et sa limite lorsque le nombre de groupes tend vers l'infini est le processus de restaurant chinois [4, 155].

Une autre alternative est l'utilisation de l'*a priori* de Jensen-Liu [92], proposé à l'origine en réponse aux limites du processus chinois, afin de supprimer l'effet *rich get richer* où un groupe a plus de probabilités d'être choisi à une itération donnée s'il est plus gros que ses concurrents. Il se définit formellement de la façon suivante :

$$P(\Phi|\pi) \propto \prod_{\phi_{i,n}\in\Phi} \lambda_{\phi_{i,n}}^{c_{i,n}-1} (\lambda_{\phi_{i,n}} + c_{i,n}) \prod_{j=1}^{c_{i,n}} (\lambda_{\phi_{i,n}} + j)^{-|\phi_{i,n}^{-1}(j)|}.$$

Considérant l'ensemble des termes énoncés dans cette section, le score pour évaluer un MRP-IR peut être finalement défini de la façon suivante :

$$\gamma_{IR}(\mathcal{S}, \Phi|\mathcal{I}, \pi) = log\, P(\mathcal{I}|\mathcal{S}, \Phi, \pi) + \lambda_{\mathcal{S}}\, log\, P(\mathcal{S}) + \lambda_{\Phi}\, log\, P(\Phi|\pi)$$

prenant ses valeurs dans l'intervalle $]-\infty, 0]$. Les paramètres $\lambda_{\mathcal{S}}$ et λ_{Φ} permettent de contrôler l'importance de chaque terme dans le score final. Nous ajoutons explicitement ici un terme d'a priori pour les fonctions de partition par rapport à la littérature originale sur les MRP-IR [65] et proposons des fonctions concrètes issues de la littérature pour ce terme.

3.10 Limites

Les MRP-IR définis dans la littérature présentent des limites, à la fois en terme des distributions de probabilité qu'il est possible de représenter, mais également à cause des contraintes spécifiées sur les fonctions de partition.

Nous abordons dans cette section les limites des MRP-IR et nous focalisons particulièrement sur les contraintes d'apprentissage. Nous commençons par aborder les limites inhérentes aux contraintes définies sur les fonctions de partition, puis nous continuons par celles induites sur l'apprentissage des dépendances probabilistes.

3.10.1 Fonctions de partition

La qualité d'un MRP-IR dépend significativement de la qualité de ses fonctions de partition. Or, l'apprentissage de celles-ci, tel que défini dans la littérature, présente plusieurs limites.

Tout d'abord, leur apprentissage repose intégralement sur les attributs de fonctions de partition associés. Or, ces derniers doivent être inclus dans l'ensemble des attributs définis pour le schéma de relation des individus à partitionner. Cette contrainte, au cœur de la définition des MRP-IR, est un biais de représentation notable, ne permettant pas de profiter de toute l'information disponible dans un jeu de données relationnel. En effet, même si les MRP-IR représentent des distributions de probabilité dans des cas où les références entre tuples ne sont pas connues, le jeu de données utilisé pour les apprendre contient cette information de structure des associations. Un objectif des MRP-IR étant de prédire des références entre tuples, une information d'intérêt lors de l'apprentissage est la topologie décrite par les associations du jeu de données d'apprentissage et non pas uniquement la description de ces tuples par leurs seuls attributs, qui peut donner une information contradictoire avec un objectif de résolution de références par la suite.

La figure 3.6 illustre cette dualité de point de vue entre la description des tuples par leurs attributs et leur organisation pour un type d'association donné. Considérant deux types d'entités *User* et *Movie* dans le contexte de recommandation de films, nous pouvons voir que le partitionnement des tuples de chaque relation selon leurs seuls attributs descriptifs internes permet d'obtenir une bonne homogénéité des clusters à l'égard de ces relations respectives. Toutefois, si l'on considère ensuite les associations pour le type *Rating*, nous pouvons voir que ces clusters homogènes ne sont pas du tout fidèles à la topologie induite par la matrice d'association des utilisateurs aux films. Nous pouvons ainsi déduire de cet exemple que si l'on décide de partitionner les individus en utilisant les seuls attributs, le modèle appris risque de ne pas représenter fidèlement l'incertitude de références entre les deux types d'entités pour ce type d'association, ce qui peut poser problème pour prédire de nouvelles associations entre les individus.

La détection de densités entre tuples de différents types mène naturellement aux travaux visant à trouver des *co-partitions* ou *co-clusters* entre des ensembles d'individus potentiellement différents, souvent disjoints, mais liés par une information d'affinités entre eux. L'objectif de ces algorithmes dits de *co-partitionnement*, de *co-clustering* ou encore de *clustering relationnel* vise à regrouper des individus d'un même ensemble d'individus de telle sorte qu'ils soient affiliés aux individus des autres ensembles de façon proche. De nombreuses approches existent dans la littérature pour résoudre ce pro-

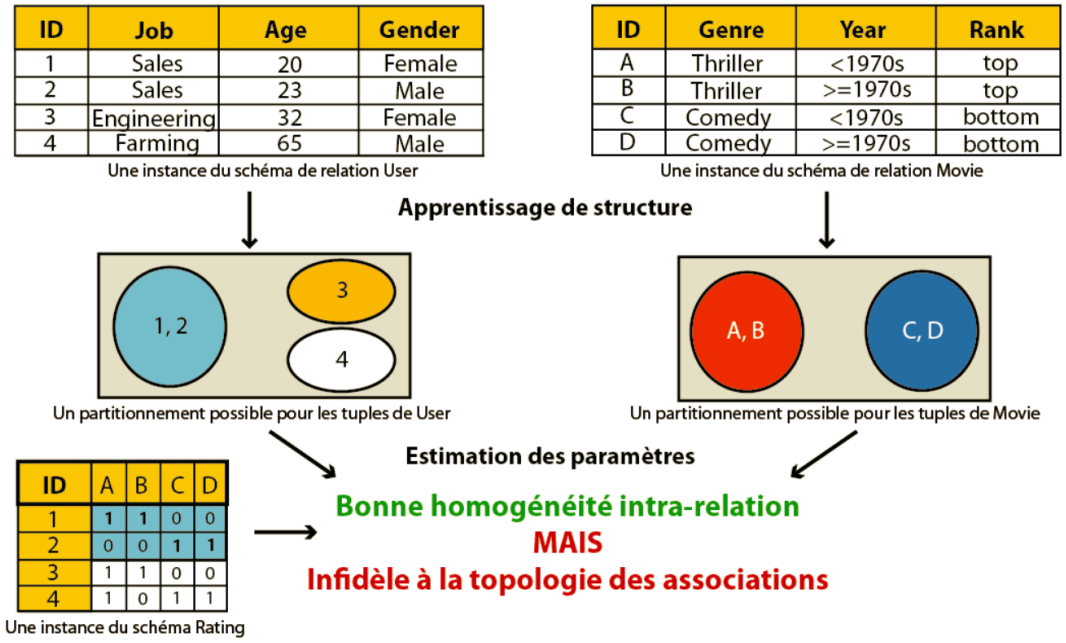


FIGURE 3.6 – Illustration de la dualité de point de vue entre une description des tuples par leurs attributs et leur organisation à l'égard d'un certain type d'association.

blème, que les données soient vues comme un graphe multi parties hétérogène [134], ou comme une matrice (ou tenseur) à factoriser [114, 51]. Ces algorithmes ont montré leur intérêt par rapport à leurs pairs réalisant des partitionnements uni partie sur des matrices de similarité.

En outre, une autre conséquence pragmatique de l'utilisation des seuls attributs pour le partitionnement est d'être sensible au nombre et à la pertinence de ceux-ci, à l'égard des types d'association, pour un type d'entités. Il est alors improbable qu'un faible nombre d'attributs disponible permette de trouver un bon partitionnement des tuples pour une contrainte de référence donnée, à cause de la pauvreté de l'espace de représentation. Dans un cas extrême, où aucun attribut n'est disponible, l'utilisation des MRP-IR est même théoriquement impossible.

Ensuite, l'apprentissage individuel et glouton de fonctions de partitions pour chaque contrainte de référence d'un type d'association rend très difficile la découverte des attributs des différents types d'entités qui, ensemble, offrent de bonnes capacités prédictives pour les associations de ce type en général. La convergence vers un optimum local est ainsi fréquente.

Enfin, même si des solutions plus évoluées à base d'arbres de décision sont rapidement évoquées dans la littérature, la seule stratégie de partitionnement effectivement utilisée à notre connaissance à partir d'un ensemble d'attributs de fonction de partition est la génération d'un groupe distinct par élément du produit cartésien des valeurs des attributs de cet ensemble. La première conséquence de ce choix, déjà évoquée dans la littérature [65], est l'explosion rapide du nombre de groupes au fil de l'ajout d'attributs à la fonction de partition. Ceci est problématique puisqu'il limite l'abstraction des connaissances au fur et à mesure que le nombre de groupes se rapproche du nombre de tuples à partitionner, et entraîne de nouveau un problème de disponibilités de statistiques suffisantes que l'on cherche pourtant à éviter dans ces modèles. Une seconde

conséquence évidente de ce choix est la restriction des attributs de fonction de partition à des ensembles dénombrables. Il est important de noter que cette limite a pour autre conséquence de restreindre le nombre d'attributs de fonctions de partition à des valeurs très faibles en pratique, si l'on cherche à introduire un *a priori* sur les fonctions de partition. En effet, ajouter un attribut à une fonction de partition multiplie par définition le nombre de groupes de celle-ci par le nombre de valeurs du nouvel attribut, suggérant une variation conséquente du résultat de l'évaluation du terme relatif à l'*a priori* de la fonction de partition dans le calcul du score du modèle.

3.10.2 Structure

Au-delà des limites dues à leurs fonctions de partition, les MRP-IR présentent également des limites notables sur leurs structures graphiques et donc sur les connaissances qu'il est possible d'apprendre dans de tels modèles.

Tout d'abord, l'incertitude de groupe est impossible pour de nouvelles entités. En effet, les fonctions de partition dépendant de façon déterministe de leurs attributs, et aucune variable aléatoire n'étant présente dans les types d'entités pour représenter une éventuelle incertitude d'appartenance aux différents groupes, ces attributs sont obligatoires lors de l'instanciation d'un RBP pour un jeu de données compatible avec le schéma de base de données (c'est le sens qu'ont les dépendances induites entre les attributs de fonction de partition et les sélecteurs dans le MRP-IR complété) et sont une condition nécessaire pour la génération des distributions de probabilité des variables de référence $R_{i,f}$.

Ensuite, il est très difficile, voire impossible, de découvrir des dépendances entre diverses associations, à cause des nombreuses dépendances induites ajoutées à un MRP-IR complété, rendant très probable la violation de la contrainte d'acyclicité si des dépendances entre variables de types d'associations sont définies.

Enfin, à cause de l'apprentissage individuel et glouton des ensembles d'attributs de fonction de partition, la convergence de l'algorithme d'apprentissage peut être longue et les dépendances probabilistes découvertes peuvent être sous-optimales, car un bon ensemble d'attributs pour une fonction de partition d'un sélecteur d'un type d'association dépend généralement de l'ensemble d'attributs des fonctions de partition des autres sélecteurs de ce type d'association.

Ce problème est illustré dans la figure 3.7. Les tables décrites en 3.7(a) définissent une instance utilisée pour l'apprentissage d'un MRP-IR sur le domaine UM. La table 3.7(b) est la vue obtenue par jointure entre les trois relations, où chaque tuple correspond à un tuple de la relation $Rating$ auquel sont ajoutés les détails de chaque $User$ et $Movie$ impliqué. La figure 3.7(c) décrit l'optimum global que l'on cherche à découvrir à partir de ce jeu de données, et présente la table de probabilités pour $P(S_{Movie}|S_{User})$ correspondante. Nous pouvons voir que l'ensemble d'attributs optimal pour chaque fonction de partition de $Rating$ est dépendant des ensembles d'attributs des autres fonctions de partition du même type d'association. En effet, nous pouvons voir dans l'exemple que deux attributs de $User$ impliquent de façon déterministe deux attributs de $Movie$ à l'égard du type d'association, sans qu'il existe toutefois une règle déterministe entre paires d'attributs. La figure 3.7(d) montre un optimum local qui pourrait être obtenu à cause de l'apprentissage indépendant (et glouton) de chaque ensemble d'at-

tributs de partition. En effet, pris indépendamment, l'attribut job du schéma $User$ est celui proposant la distribution la moins équilibrée et donc offre la meilleure connaissance discriminatoire pour le sélecteur concerné. Du côté de $Movie$, c'est $rank$ qui propose l'entropie la plus réduite. Pourtant, en regardant la table de probabilités pour S_{Movie}, nous pouvons observer une incertitude plus grande entre les deux sélecteurs que pour l'optimum global. Cet exemple illustre l'importance de considérer mutuellement les fonctions de partition pour obtenir de meilleures capacités prédictives pour la résolution de références.

3.11 Autres extensions des MRP

La littérature sur les MRP propose plusieurs versions de ces modèles, répondant à différentes problématiques, ou abordant un même problème sous différents angles. Outre les MRP orientés attributs, à la base de tous, ce chapitre se focalise sur une extension appelée MRP avec incertitude de référence. Au-delà de ces deux modèles, d'autres extensions existent, comme les MRP avec *incertitude d'existence* (MRP-IE) dont l'objectif est similaire à celui des MRP-IR en ce sens qu'ils cherchent à introduire une incertitude de structure entre les tuples, selon un paradigme de représentation toutefois différent.

Dans les MRP-IE [67, 66, 65], le nombre d'associations existant entre différentes entités n'est plus connu et les variables de référence (ainsi que les sélecteurs) n'y sont plus présentes. À la place, dans chaque type d'association est ajouté un attribut booléen d'existence. La représentation de l'incertitude de structure se fait alors par définition d'une distribution de probabilités d'existence d'un objet d'association, sachant un ensemble de parents dans le MRP-IE, c.-à-d. les attributs des entités associées. Étant donné un MRP-IE et une nouvelle instance, il est possible de générer un RBP dans lequel une association potentielle est créée pour chaque combinaison de tuples des entités dont le type est impliqué dans le type d'association. Par exemple, dans le domaine UM, les deux seules variables d'un MRP-IE du schéma $Rating$ seraient $A_{Rating,Rating}$ et $A_{Rating,Exist}$. Dans le cas d'une instance avec n_u tuples $User$ et n_m tuples $Movie$, le RBP correspondant comporterait $n_u \times n_m$ objets association de type $Rating$ potentiels, chacun ajoutant deux variables au RBP. Les MRP-IE ont été utilisés dans la littérature pour des systèmes de recommandation [88].

D'autres extensions des MRP ont également été proposées pour résoudre des problèmes différents. Les MRP avec *incertitude de hiérarchie* (MRP-IH) [147, 66] visent à découvrir des connaissances impliquant des représentations de granularité différentes. Dans ces modèles, les schémas de relation peuvent être liés de façon hiérarchique. Un MRP-IH définit sur un schéma de base de données hiérarchique $\mathbf{R}$ contient alors pour chaque schéma de relation $\mathcal{R}^i \in \mathbf{R}$ une variable aléatoire pour chacun de ses attributs. Ces dernières sont également dupliquées pour chaque schéma de relation ayant $\mathcal{R}^i$ dans la hiérarchie des schémas. Il est ainsi possible de définir des distributions de probabilités différentes pour deux variables aléatoires correspondant à un même attribut dans un schéma de relation, mais pour deux sous-types différents.

Les MRP avec incertitude d'identité [68] (MRP-II) visent à réaliser de la résolution d'entités, c.-à-d. découvrir les tuples formels qui matérialisent les mêmes concepts réels. Les modèles sont sommairement définis dans [68] et reposent sur l'ajout de types

User			
id	age	gender	job
u1	a1	g1	j0
u2	a0	g0	j0
u3	a1	g0	j0
u4	a0	g1	j0
u5	a0	g0	j1
u6	a1	g0	j1

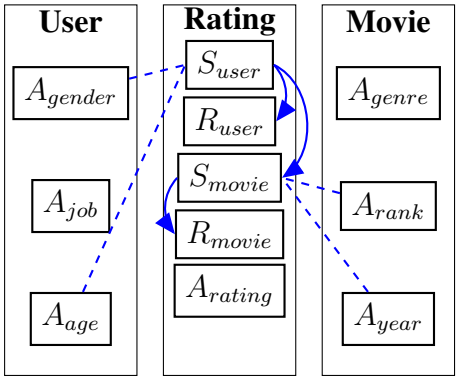
Rating			
id	user	movie	rating
r1	u1	m1	r0
r2	u2	m2	r0
r3	u3	m2	r1
r4	u4	m3	r1
r5	u1	m4	r2
r6	u5	m5	r1
r7	u6	m5	r0
r8	u4	m6	r2

Movie			
id	genre	rank	year
m1	g1	r0	y0
m2	g0	r0	y1
m3	g1	r1	y0
m4	g1	r0	y0
m5	g1	r0	y1
m6	g0	r1	y0

(a) Exemple d'instance d'apprentissage pour le domaine UM.

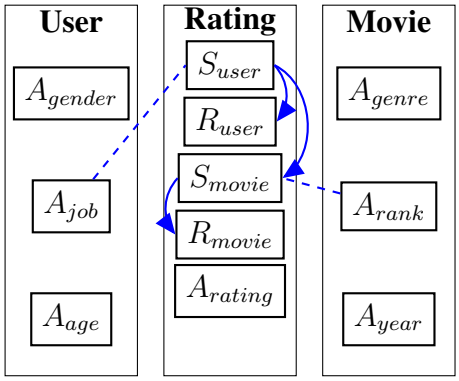
Rating join view									
	user				movie				
id	id	age	gender	job	id	genre	rank	year	rating
r1	u1	a1	g1	j0	m1	g1	r0	y0	r0
r2	u2	a0	g0	j0	m2	g0	r0	y1	r0
r3	u3	a1	g0	j0	m2	g0	r0	y1	r1
r4	u4	a0	g1	j0	m3	g1	r1	y0	r1
r5	u1	a1	g1	j0	m4	g1	r0	y0	r2
r6	u5	a0	g0	j1	m5	g1	r0	y1	r1
r7	u6	a1	g0	j1	m5	g1	r0	y1	r0
r8	u4	a0	g1	j0	m6	g0	r1	y0	r2

(b) Vue construite en réalisant la jointure des 3 schémas sur les données de (a).



			S_{user}			
			0	1	2	3
			a_0g_0	a_0g_1	a_1g_0	a_1g_1
S_{movie}	0	r_0y_0	0	0	0	1
	1	r_0y_1	1	0	1	0
	2	r_1y_0	0	1	0	0
	3	r_1y_1	0	0	0	0

(c) Optimum global que l'on cherche à apprendre et CPD de $S_{Rating,User}$ associée.



			S_{user}	
			0	1
			j_0	j_1
S_{movie}	0	r_0	$\frac{2}{3}$	1
	1	r_1	$\frac{1}{3}$	0

(d) Exemple d'optimum local où l'on peut converger à cause de l'apprentissage indépendant et glouton des attributs de chaque fonction de partition.

FIGURE 3.7 – Illustration du problème de convergence locale due à l'apprentissage indépendant et glouton des attributs de fonctions de partition.

d'entités intermédiaires permettant de distinguer une entité *référencée* d'une entité *réelle*.

À titre d'exemple dans le domaine UM, un MRP-II définit deux nouveaux types d'entités en plus des deux préexistant, qui peuvent être nommés $UserReference$ et $MovieReference$. De plus, deux nouveaux types d'association sont ajoutés :

- $ReferencesUser$ entre $User$ et $UserReference$;
- $ReferencesMovie$ entre $Movie$ et $MovieReference$.

Les références ciblant précédemment $User$ et $Movie$ référencent à la place les types d'entités référencés correspondant. Finalement, le problème de découverte d'une entité réelle sachant une entité référencée peut être réalisé en utilisant les principes définis pour les MRP-IR. Il est important de noter que ces modèles supposent la connaissance exhaustive des entités réelles pour chaque type.

3.12 Conclusion

Dans ce chapitre, nous avons tout d'abord défini le problème d'incertitude de références, où les ensembles de tuples sont connus pour chaque relation sans toutefois connaître les valeurs des attributs ni les valeurs de références entre eux. Nous avons ensuite défini de façon précise les modèles relationnels probabilistes avec incertitude de référence, une extension aux MRP définis dans le chapitre précédent, adaptés à ce contexte, et abordé l'inférence et l'apprentissage dans le cadre de ces modèles. Une de nos contributions dans ce chapitre a été d'expliciter divers points de détail non abordés dans la littérature, tels que la résolution des distributions de probabilités dans les réseaux bayésiens à plat issus de ces modèles. Nous avons en outre proposé une contribution relative à l'évaluation de ces modèles à partir de données en ajoutant explicitement un terme d'a priori pour les fonctions de partition et en proposant des choix possibles pour celui-ci, comme l'a priori d'Ewens-Pitman ou celui de Jensen-Liu. Finalement, nous avons exhibé un ensemble de limites à ces modèles amenant le besoin de les faire évoluer. Ces limites concernent à la fois les contraintes sur les fonctions de partition, empêchant principalement l'usage d'algorithmes de partitionnement relationnels évolués, et la structure graphique des modèles, empêchant principalement la découverte de connaissances entre associations et l'inférence de partitions dans les réseaux bayésiens générés.

Dans le prochain chapitre, nous nous focalisons sur la première catégorie de limites en réalisant l'état de l'art des méthodes de partitionnement relationnelles qui nous intéressent pour la définition de nos extensions aux modèles relationnels probabilistes avec incertitude de référence, proposés dans les chapitres suivants.

4

Méthodes de clustering relationnel

Sommaire

4.1 Introduction

Dans les communautés orientées clustering, les techniques de clustering relationnel, sont devenues de plus en plus populaires et ont montré de meilleurs résultats que leurs équivalents non relationnels lors de nombreuses expériences impliquant des jeux de données non i.i.d. Nous pouvons citer par exemple les résultats en fouille de documents [49, 186], en fouille de données génomiques [21], et en traitement d'images et de vidéos [118, 39]. À partir d'un type d'association défini entre plusieurs types d'entités, dont une instance peut être matérialisée sous la forme d'une matrice (ou tenseur) de co-occurrences ou encore d'un (hyper-)graphe pondéré, l'idée derrière les approches relationnelles est de grouper simultanément les ensembles d'entités potentiellement disjoints, utilisant l'information des associations elle-même, éventuellement enrichie par l'information disponible entre les entités d'un même ensemble.

De nombreuses approches ont été utilisées pour résoudre ce problème. Certaines méthodes utilisent des techniques de factorisation de la matrice de co-occurrences entre les ensembles d'entités, ou du laplacien de cette matrice. Nous pouvons citer par exemple les méthodes basées sur la décomposition en valeurs singulières [116] (SVD) ou les approches spectrales [49]. D'autres techniques de factorisation intéressantes sont basées sur les approches de factorisation non négatives de matrice (NMF). Leur objectif est de factoriser une matrice rectangulaire en deux (ou trois) matrices *facteur* de plus faible rang, de sorte que le produit de ces facteurs offre une reconstruction la plus fidèle possible de la matrice originale. Ces approches sont bien étudiées dans [150, 114, 115] et des exemples pour la fouille de données sont détaillés dans [21, 39].

Nous pouvons également considérer le problème de clustering relationnel sur un jeu de données modélisé par un graphe où les nœuds représentent les individus à partitionner et les arcs définissent les associations observées entre eux. La tâche de clustering relationnel peut alors prendre la forme d'une coupe dans un graphe en un nombre de groupes prédéterminés, essayant de trouver une solution proposant un bon compromis entre équilibre des tailles de ces groupes et minimisation du nombre d'arcs entre eux, ou d'une recherche de structures particulières dans ce graphe, généralement appelées communautés, dont les définitions varient en fonction des algorithmes utilisés.

Suite à nos conclusions concernant les modèles définis dans le chapitre précédent, nous nous intéressons dans ce chapitre aux algorithmes de clustering relationnel des différentes familles énoncées ci-dessus. Nous commençons par décrire les approches orientées factorisation de matrices, avant de nous intéresser aux techniques de partitionnement de graphe et enfin de détection de communautés dans les graphes. Nous nous restreignons ici aux algorithmes dédiés au clustering strict, proposant une partition de l'ensemble des individus. Ce chapitre se termine par une contribution concernant une équivalence entre les problèmes de factorisation de matrice régularisée dans des variétés et de partitionnement dans un graphe augmenté. Une version préliminaire de cette contribution a conduit à la publication d'un article de recherche, présenté lors de la conférence ESANN 2015 [40].

4.2 Factorisation de matrices

4.2.1 Analyse en Composantes Principales et Décomposition en valeurs singulières

Parmi les méthodes de factorisation les plus connues et utilisées, l'analyse en composantes principales (ACP) [93] est une technique de réduction de dimension visant à transformer une matrice de données X de façon à ce que la nouvelle représentation X_{ACP} capture le maximum de variabilité des données originales avec un minimum de ses colonnes. Plus précisément, une ACP non normée est réalisée en calculant la décomposition en vecteurs propres de la matrice de covariance de X. Ces vecteurs propres sont alors triés par ordre décroissant de leurs valeurs propres et chacun de ces vecteurs propres devient la k^{ime} composante principale de X, et donc k^{ime} colonne de X_{ACP}, ssi il a la k^{ime} valeur propre la plus élevée. Les composantes principales ont pour propriétés d'être des combinaisons linéaires des variables originales et d'être indépendantes entre elles et l'approximation à p dimensions la plus respectueuse de la variance des données peut être obtenue en projetant les individus du jeu de données original sur les p premières composantes.

La décomposition en valeurs singulières (SVD), également très populaire, a pour but de trouver une décomposition $U\Sigma V^t$ d'une matrice X de dimensions $n \times m$, où Σ est une matrice diagonale dont les éléments sont les valeurs singulières, non négatives, de X et où la matrice orthogonale U (resp. V) contient les vecteurs singuliers gauche (resp. droites) de X. Les valeurs singulières sont les racines carrées des valeurs propres de la matrice de covariance $\frac{1}{m}XX^t$ et il a été prouvé que la meilleure approximation de X de rang p est $X_p = U_p\Sigma_p{V_p}^t$, où U_p (resp. V_p) contient les p valeurs singulières gauches (resp. droites) correspondant aux p valeurs singulières les plus élevées, constituant également les valeurs de la matrice diagonale Σ_p. La décomposition en valeurs singulières est proche de l'ACP en ce sens qu'une SVD sur la matrice centrée par variable (colonne) est équivalente à une ACP.

Les décompositions offertes par l'ACP et la SVD minimisent la perte d'information du jeu de données lorsque l'on souhaite réduire la dimension de celles-ci, ce qui permet d'obtenir les projections linéaires les plus fiables dans un espace réduit à m dimensions, obtenues en prenant les m premières composantes principales ou vecteurs singuliers. Toutefois, elles présentent deux défauts majeurs. D'une part, les représentations en résultant ne sont généralement pas creuses. D'autre part, celles-ci ne sont également pas nécessairement non-négatives. Ceci rend en pratique difficile l'interprétation de telles projections. Des extensions existent pour ces méthodes, proposant des termes de pénalisation supplémentaires dans les fonctions objectifs pour forcer des propriétés supplémentaires sur les résultats, mais il semble difficile de les concilier toutes ensemble avec la contrainte d'orthogonalité. Pour finir, notons que dans le cas d'un clustering relationnel, où l'on souhaite réduire la dimension des deux ensembles d'individus impliqués, ces méthodes centrées sur un seul d'entre eux à la fois ne semblent pas appropriées.

4.2.2 Besoins de non-négativité et représentations creuses

De nombreux jeux de données dans la vie réelle sont composés de valeurs non négatives et dans la majorité des cas associés, les variables latentes que l'on désire y rechercher doivent conserver cette propriété pour être interprétables. En apprentissage automatique, la propriété de non-négativité est souvent associée de près ou de loin à des notions de distributions de probabilités entre concepts.

De même, la sélection de variables en apprentissage automatique, ainsi que la généralisation est souvent liée à la notion de représentation creuse des données. Cette seconde propriété est également importante pour l'interprétation, mais présente souvent aussi l'avantage de réduire les coûts de calcul impliquant ces représentations.

4.2.3 Factorisation non négative de matrice

Une famille de méthodes dédiée à la factorisation non négative de matrices a récemment gagné en intérêt dans les communautés liées au clustering de données [150, 114]. Considérons une matrice aux éléments non négatifs R_{12} de taille $n_1 \times n_2$, et une valeur de dimension d. Les méthodes de factorisation non négative de matrice ont pour objectif, dans leur version la plus simple, la découverte des matrices facteurs non négatifs G_1 et G_2 de taille $n_1 \times d$ et $n_2 \times d$ de telle sorte que, relativement à une mesure de dissimilarité Δ, nous ayons :

$$\Delta(R_{12}, \tilde{R}_{1,2}) \approx 0. \tag{4.1}$$

avec $\tilde{R}_{1,2} = G_1.{G_2}^t$. Généralement, d est choisi petit par rapport à n_1 et n_2. De nombreuses mesures pour Δ ont été proposées dans la littérature :

- Les l-p normes en général qui vont de la distance de Manhattan à la distance de Minkowski, en passant par la distance Euclidienne :

$$\underset{p}{\Delta}(R_{12}, \tilde{R}_{12}) = \left[\sum_{ij} |R_{12}(ij) - \tilde{R}_{12}(ij)|^p\right]^{1/p}, p \geq 1$$

- Les l-p normes pondérées, pour lesquelles on inclut une matrice de poids W dans le calcul pour privilégier la fiabilité de la reconstruction de certaines parties de R_{12}, généralement en fonction de la confiance que l'on a sur les données de cette matrice :

$$\underset{pW}{\Delta}(R_{12}, \tilde{R}_{12}) = \left[\sum_{ij} W(ij)|R_{12}(ij) - \tilde{R}_{12}(ij)|^p\right]^{1/p}, p \geq 1$$

- la KL divergence dans le cas où les matrices sont normalisées, ou sa généralisation appelée I-divergence pour des matrices aux éléments non négatifs :

$$\underset{KL}{\Delta}(R_{12}, \tilde{R}_{12}) = \sum_{ij} \tilde{R}_{12}(ij) \log \frac{R_{12}(ij)}{\tilde{R}_{12}(ij)}$$

$$\underset{I}{\Delta}(R_{12}, \tilde{R}_{12}) = \sum_{ij} \tilde{R}_{12}(ij) \log \frac{R_{12}(ij)}{\tilde{R}_{12}(ij)} - R_{12}(ij) + \tilde{R}_{12}(ij)$$

– Les f-divergences de Csiszár en général qui se définissent à partir d'une fonction convexe f telle que $f'(1) \ /=0$ sur des matrices comportant uniquement des éléments non négatifs :

$$\underset{f}{\Delta}(R_{12}, \tilde{R}_{12}) = \sum_{ij} \tilde{R}_{12}(ij) f\left(\frac{R_{12}(ij)}{\tilde{R}_{12}(ij)}\right) + f'(1)\left[\sum_{ij} \tilde{R}_{12}(ij) - R_{12}(ij)\right]$$

Dans le cas de matrices normalisées, l'expression se simplifie :

$$\underset{f}{\Delta}(R_{12}, \tilde{R}_{12}) = \sum_{ij} \tilde{R}_{12}(ij) f\left(\frac{R_{12}(ij)}{\tilde{R}_{12}(ij)}\right)$$

Notez que la f-divergence définie à partir de $f(x) = 1 - x + x \log x$ est la I-divergence. De même, la f-divergence définie à partir de $f(x) = |x - 1|$ est la norme l-1 ;

– Les divergences de Bregman, construites à partir de n'importe quelle fonction convexe dérivable $\mathbf{\Phi}$ sur $\mathbb{R}$:

$$\underset{\mathbf{\Phi}}{\Delta}(R_{12}, \tilde{R}_{12}) = \sum_{ij} \Phi(R_{12}(ij)) - \Phi(\tilde{R}_{12}(ij)) - \frac{d\Phi}{d\tilde{R}_{12}(ij)}(R_{12}(ij) - \tilde{R}_{12}(ij))$$

– Les α-divergences, qui sont des f-divergences et des divergences de Bregman, sont une famille paramétrée par $\alpha \in \mathbb{R}$ de divergences permettant de relier de façon continue plusieurs divergences usuelles dont la I-divergence $\underset{I}{\Delta}(R_{12}, \tilde{R}_{12})$ (pour $\alpha \to 1$) et la I-divergence duale $\underset{I}{\Delta}(\tilde{R}_{12}, R_{12})$ (pour $\alpha \to 0$) ;

– Les β-divergences, qui sont également des divergences de Bregman, permettent de relier de la même façon certaines divergences usuelles telles que la I-divergence ($\beta \to 0$) et la distance Euclidienne au carré ($\beta = 1$).

Dans la littérature, la mesure la plus populaire est le carré de la norme de Frobenius, dénotée $\|A\|_F^2$, qui n'est autre que le carré de la norme Euclidienne des éléments de la matrice A considérée comme un vecteur. La I-divergence est également très utilisée pour la factorisation de matrices et il a été prouvé [63] que les solutions de NMF minimisant la KL-divergence étaient équivalentes aux optima locaux obtenus par maximum de vraisemblance avec l'algorithme pLSA [85] (probabilistic Latent Semantic Analysis), un algorithme de clustering de documents basé sur un modèle génératif.

4.2.4 NMF orthogonale et creuse

Dans leurs papiers séminaux, Lee et Seung [114, 115] ont montré l'interprétabilité de la factorisation non négative de matrices pour la découverte de représentation d'objets par parties, p.ex. pour représenter des images comme combinaisons linéaires d'images *de base*, grâce aux propriétés d'additivité des facteurs, la décomposition ne permettant par définition pas la représentation d'éléments à soustraire. Toutefois, l'interprétabilité en terme de clustering à ce stade par la seule propriété de non-négativité n'est pas claire et sous-optimale. Il a été montré plus tard que les contraintes de représentation creuse [87] et d'orthogonalité [51] étaient importantes pour obtenir des interprétations plus

intuitives des facteurs. Dans le cas d'une factorisation orthogonale, il a été prouvé [51] que minimiser la fonction objectif suivante, avec une contrainte d'orthogonalité sur un facteur :

$$\min \|R_{12} - G_1 {G_2}^t\|, \, t.q. \, G_1 \geq 0, \, {G_2}^t G_2 = I \tag{4.2}$$

était équivalent à du clustering par la méthode des k-moyennes [51], ce qui garantit en outre l'unicité de la solution. Dans un contexte de clustering simple (non relationnel), où un jeu de données R_{12} prend la forme d'une matrice décrivant n_1 individus par n_2 caractères, une factorisation de R_{12} en un produit de facteurs $G_1 G_2^T$, avec une contrainte d'orthogonalité pour G_2, G_1 peut être interprétée comme une matrice d'assignation des n_1 individus aux k clusters dont les prototypes moyens sont les vecteurs de G_2 (c.-à-d. les centroïdes des k clusters). La factorisation de (4.2) a également été adaptée pour la réalisation de co clustering en contraignant les deux facteurs à présenter des colonnes orthogonales :

$$\min \|R_{12} - G_1 {G_2}^t\|, \, t.q. \, {G_1}^t G_1 = I, \, {G_2}^t G_2 = I. \tag{4.3}$$

Dans ce cas, les matrices G_1 et G_2 représentent respectivement les affinités de chaque individu de l'ensemble d'entités correspondant à un co cluster particulier.

La formulation du problème telle que faite en (4.3) présente toutefois un défaut majeur relatif à l'espace des solutions admissibles. En effet, dans ces circonstances, la double contrainte d'orthogonalité associée à la mise en commun de l'ensemble de co clusters rend difficile la convergence vers un optimum intéressant [51]. De ce fait, il est intéressant d'ajouter un troisième facteur à la décomposition dans ces circonstances, afin de capturer les effets d'échelle entre les facteurs et éventuellement de définir un nombre de clusters différent pour les deux ensembles d'entités. On parle alors de tri factorisation non négative de matrice (NMTF) :

$$\min \|R_{12} - G_1 S {G_2}^t\|, \, t.q. \, {G_1}^t G_1 = I, S > 0, \, {G_2}^t G_2 = I \tag{4.4}$$

où la matrice S peut être interprétée comme une matrice d'affinité entre les différents clusters de chaque ensemble d'entités.

4.2.5 NMF et régularisation laplacienne

Les méthodes NMF et NMTF abordées jusqu'ici supposent que les entités de chaque ensemble sont contenues de façon uniforme dans les espaces Euclidiens à grande dimension correspondant au rang de la matrice originale R_{12}. Toutefois, de récents travaux [25] ont montré que ces ensembles décrivaient des topologies particulières et que les individus tendaient à être inclus dans des variétés de plus faible dimension. En incluant ces données de variété comme termes de régularisation dans (4.4) et (4.3), soit pour un seul des ensembles d'entités [25], soit, comme plus récemment introduit, pour les deux [73, 165], il a été montré qu'il était possible d'obtenir de meilleurs résultats de factorisation en pratique. Plus précisément, nous pouvons décrire l'information de variété pour l'ensemble d'entités X_k par une matrice carrée d'affinités ou de similarités entre individus du même ensemble W_k. Cette matrice peut contenir pour chaque paire d'entités une similarité réelle, mais peut également être construite par k plus proches voisins, de telle sorte que :

$$W_k(ij) = \begin{cases} 1 & \text{si } i \in X_k \text{ est un } k \text{ plus proche voisin de } j \in X_k \text{ ou inversement,} \\ 0 & \text{sinon.} \end{cases} \tag{4.5}$$

À partir des matrices R_{12}, W_1 et W_2, un algorithme de factorisation régularisé a pour but de trouver les matrices G_1, G_2 et S optimisant la fonction objectif suivante :

$$\begin{aligned} J &= \|R - G_1 S {G_2}^T\|_F^2 + \sum_{k\in\{1,2\}} \lambda_k \mathbf{tr}\left[{G_k}^T L_k G_k\right] \\ t.q.\, &{G_1}^t G_1 = I, {G_2}^t G_2 = I, S \geq 0. \end{aligned} \tag{4.6}$$

où $L_k = D_k - W_k$ est la matrice laplacienne de W_k et D_k est la matrice diagonale des degrés de W_k, c.-à-d. où $D_k(ii) = \sum_j W_k(ij)$.

4.2.6 NMF symétrique

Il est important de noter que dans le cas où les deux ensembles d'entités sont identiques, c.-à-d. $X_1 = X_2$, et que la matrice d'origine R_{12} est symétrique, il est toujours possible d'utiliser des techniques de factorisation non négatives de matrice telles qu'énoncées précédemment. Toutefois, dans ce cas précis, nous pouvons également introduire la contrainte de symétrie dans la factorisation de façon à obtenir une décomposition de telle sorte que $R_{12} \approx G_1 {G_1}^t$. Dans un contexte de clustering, cette situation peut correspondre à un clustering relationnel de graphe uni partie non dirigé et pondéré entre individus où n'importe quel arc peut exister entre les individus.

4.2.7 Multi relations binaires, relations n-aires

Cette section présente essentiellement la factorisation non négative de matrices dans le contexte où le jeu de données est constitué d'une unique matrice R_{12}. Ces travaux peuvent toutefois être étendus à d'autres types de relations comme les multi relations binaires ou les relations n-aires [56, 183, 182]. L'objectif de ces travaux est identique, à savoir la découverte de clusters multi partis pour des jeux de données pouvant impliquer plus de deux ensembles d'entités X_k et pouvant être constitués de plusieurs relations.

Dans le cas de factorisation multi relationnelle binaire, pour m ensembles d'entités $X = \{X_k\}_{k\leq m}$ et étant donné des ensembles de matrices R et W avec :

$$R = \{R_{uv}, 0 < u \leq m, 0 < v \leq m\} \text{ et}$$

$$W = \{W_u, 0 < u \leq m\},$$

il est possible [182] de reformuler le problème pour le cas uni relationnel binaire. L'objectif est alors de trouver des facteurs $G = \{G_k\}_{k\leq m}$ offrant une bonne reconstruction de la matrice symétrique H à $n = \sum |X_u|$ lignes et colonnes et où :

$$H(ij) = \begin{cases} a & \text{ssi } \exists u : i \in X_u, j \in X_u \text{ et } W_u(ij) = a, \\ b & \text{ssi } \exists u : i \in X_u, j \in X_v \not\in X_u \text{ et } R_{uv}(ij) = b. \end{cases} \tag{4.7}$$

Il est alors possible de reformuler (4.6) de la façon suivante :

$$\begin{aligned} J &= \|H - GSG^T\|_F^2 + \lambda \mathbf{tr}\left[G^T L G\right] \\ t.q. &G \geq 0, G^t D G = I. \end{aligned} \tag{4.8}$$

où la matrice G (resp. S) est une matrice construite par la concaténation des G_{uv} (resp. S_{yz}) pour toute paire de types d'entités (u, v) (resp. de clusters (y, z)), et où

$L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ est la matrice laplacienne normalisée, la matrice W est construite par la concaténation des W_u et D est la matrice diagonale des degrés de chaque individu relativement à la matrice W.

Le cas de relations n-aires peut être géré par des techniques de factorisation de tableaux multidimensionnels, appelés communément *tenseurs*, et non plus seulement de matrices. Les techniques de factorisation tensorielles existent depuis longtemps, et de nombreux algorithmes ont été proposés pour ce problème. Nous pouvons citer parmi les plus célèbres l'algorithme PARAFAC [80, 79], les versions non négatives NTF [20, 98, 27], mais aussi les décompositions de Tucker [178, 179].

4.3 Partitionnement de graphes

4.3.1 Coupe totale, ratio de coupe, coupe normalisée

Soit un graphe $\mathcal{G}(V, E)$ avec des poids non négatifs sur les arcs, et k un nombre de groupes définissant une taille de partition souhaitée. Les méthodes de partitionnement ont pour but de trouver une partition de V en k groupes tels que la coupe des arcs est minimale. Dénotant par $P_V = (V_i)_{0 \leq i \leq k}$ une partition de V, une coupe triviale appelée *coupe totale* peut être définie par :

$$cut(\mathcal{G}, P_V) = \sum_{i<j} w(E_{ij}) \tag{4.9}$$

où $E_{ij} \subseteq E$ est l'ensemble des arcs ayant une extrémité dans V_i et l'autre dans V_j, et w est la fonction donnant la somme des poids pour un ensemble d'arcs. En d'autres termes, nous minimisons le poids des arcs reliant des individus de clusters différents.

Des coupes plus évoluées sont toutefois généralement utilisées [166, 117, 31], de façon à introduire des contraintes d'équilibre entre les tailles des différents groupes de la partition, ce qui est souvent souhaité pour des applications comme la répartition de nœuds sur une grappe de serveurs ou des problématiques de parallélisation de processus sur différents cœurs. De plus, cela permet d'éviter les solutions triviales sur certains jeux de données où des nœuds du graphe étant faiblement liés à ses pairs, la couple optimale est obtenue en séparant ce nœud du reste du graphe. Deux coupes entrent principalement dans cette catégorie :

- le ratio de coupe : $rcut(\mathcal{G}, P_V) = \sum_i \frac{\sum_{j \neq i} w(E_{ij})}{w(E_{ii})}$;
- la coupe normalisée : $ncut(\mathcal{G}, P_V) = \sum_i \frac{\sum_{j \neq i} w(E_{ij})}{\sum_j w(E_{ij})}$.

4.3.2 Algorithmes de partitionnement globaux

La première catégorie d'algorithmes pour la résolution du problème de partitionnement de graphe opère selon une approche globale. Les algorithmes de cette catégorie considèrent le graphe entier et les propriétés globales de celui-ci pour réaliser le partitionnement.

Une première solution de cette catégorie est de résoudre le problème de façon exacte, mais ceci n'est généralement pas calculable au-delà de quelques centaines de nœuds.

Une seconde solution est l'utilisation d'algorithmes de partitionnement spectral. Dans cette approche, l'objectif est de calculer le second vecteur propre associé à la seconde valeur propre la moins élevée de la matrice laplacienne du graphe (le premier vecteur propre ne porte pas d'information pertinente et est toujours associé à une valeur propre de 0 [49]). La partition binaire est alors obtenue en calculant la moyenne $\bar{m}$ des valeurs du vecteur propre, puis en assignant tous les nœuds du graphe ayant une valeur $m \leq \bar{m}$ sur ce vecteur propre au premier groupe, et les nœuds restant au second. Trouver plus de 2 groupes peut être effectué par bissection récursive du graphe.

Il est également possible d'utiliser des approches d'exploration de graphe [64, 94]. Commençant par un nœud pris au hasard ou selon certaines heuristiques, ces algorithmes parcourent le graphe en largeur à partir de ce nœud jusqu'à avoir rencontré la moitié de ses nœuds. Ceux-ci sont alors associés au premier groupe, et le reste du graphe au second. Encore une fois, cet algorithme peut être généralisé à plus de deux groupes par utilisation récursive de l'algorithme sur les sous-groupes trouvés aux étapes précédentes.

D'autres méthodes existent dans la littérature. Nous pouvons citer à titre d'exemple les techniques de découverte de flot maximal dans les graphes [72, 59, 53], obtenant de bons résultats en temps polynomial, mais sans prendre en compte les contraintes d'équilibre de tailles entre groupes, ou encore des techniques de partitionnement géographique [162, 161] où les nœuds sont également associés à des coordonnées spatiales offrant des informations supplémentaires pouvant être utilisés de façon efficace par les algorithmes dédiés.

4.3.3 Algorithmes de partitionnement par amélioration locale

Une seconde catégorie d'algorithmes utilise une approche locale pour réaliser le partitionnement. Ces algorithmes fonctionnent de façon itérative et essaient d'améliorer un partitionnement par modifications locales successives.

Nous pouvons citer par exemple la méthode de Kernighan et Lin [96] où, étant donné un partitionnement, le but est de trouver une paire de nœuds appartenant à des groupes différents de telle sorte qu'échanger leurs groupes respectifs maximise la perte sur le score de coupe. Le processus est répété jusqu'à ne plus pouvoir améliorer le score. Cette méthode à la base plutôt coûteuse a été améliorée depuis, p.ex. en considérant des structures de données optimisées pour la recherche locale ou en ne considérant plus que les échanges de nœuds situés à la frontière entre les groupes [58, 149, 95]. Des approches ont également été proposées, p.ex. pour étendre ces travaux au cas d'un partitionnement à $k > 2$ groupes [95] afin d'éviter les contraintes inhérentes à la bissection récursive.

D'autres approches locales existent. Certaines étendent les techniques d'exploration de graphe [181] au cas à $k > 2$ groupes, en choisissant dans un premier temps k nœuds source des groupes de façon à ce qu'ils soient bien répartis dans le graphe, puis

k parcours locaux en largeur du graphe sont effectués, ordonnés de façon à privilégier l'ajout d'un noeud dans le groupe le plus petit à chaque étape. D'autres méthodes sont basées sur le principe de marche aléatoire [124] où le partitionnement résultant dépend de l'affinité entre les nœuds, liée à la probabilité de parcourir les nœuds dans une même marche aléatoire en partant de n'importe quel nœud.

4.3.4 Approches multi niveaux

Des algorithmes très efficaces et capables de passer à l'échelle ont été conçus pour le partitionnement de gros graphes [22], incluant notamment les méthodes multi niveaux (comme METIS [94] et GRACLUS [50]) présentant l'avantage d'être très rapides d'exécution et moins sensibles aux optima locaux que les approches classiques.

Ces méthodes débutent par la simplification du graphe de façon récursive, regroupant les nœuds à chaque itération en fonction de leur proximité telle que définie par les arcs entre eux. Ensuite, une fois que le graphe possède un nombre de nœuds raisonnable (un millier par exemple), une recherche de coupe minimale est effectuée sur ce graphe simplifié. Finalement, la dernière étape inverse le processus de simplification initiale de façon itérative, inférant à chaque étape l'appartenance d'un nœud à une partition par une méthode d'optimisation locale prenant en compte l'appartenance du méta nœud à l'itération précédente. La figure 4.1, empruntée de la littérature dédiée, illustre ce processus.

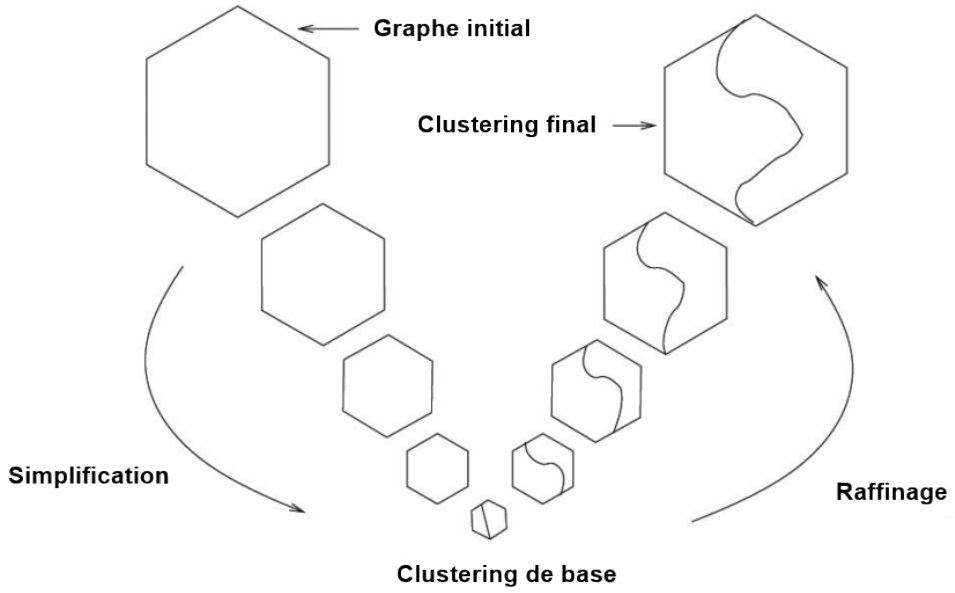


FIGURE 4.1 – Illustration des méthodes de partitionnement de graphe multi niveaux. *Image tirée de [50] et traduite en français.*

4.4 Détection de communautés dans les graphes

4.4.1 Définitions du concept de communauté

De nombreuses définitions du concept de communauté ont été données dans la littérature et les algorithmes de détection de communautés obtiennent généralement des résultats très différents, dépendant de la définition ayant amené à leur développement. L'idée générale est liée à la répartition des arcs dont la majorité devrait se trouver entre individus d'une même communauté, minimisant ainsi les arcs entre individus de communautés différentes. Toutefois, de nombreuses variantes de cette définition existent dans la littérature en fonction de la nature du problème sous-jacent à résoudre.

Une première catégorie de définition repose sur l'évaluation locale des communautés, impliquant uniquement les voisinages directs des noeuds et non pas leur position dans le graphe dans son intégralité. Certaines définitions de cette catégorie sont strictes. Dans un graphe, une composante constituée d'un ensemble de nœuds complètement connectés les uns avec les autres est appelée *clique*, et une définition naïve de communauté peut donc être définie comme une clique maximale du graphe. Toutefois, en pratique, les graphes ne présentent pas généralement de communautés aussi parfaites et de taille significative. Aussi, d'autres définitions plus souples ont été définies comme celle de *n-clique* [3] décrivant un ensemble de nœuds dont la distance entre chaque paire est au plus de n (possiblement en passant par des nœuds en dehors de la communauté), de *n-clan* [133] décrivant une n-clique dont le diamètre est d'au plus n (via le chemin le plus court sans sortir de la communauté), ou encore de *k-plex* [164] décrivant un sous-graphe maximal dans lequel chaque nœud est adjacent à tous les autres du sous-graphe sauf au plus k d'entre eux. Toutes ces définitions présentent le défaut de décrire des communautés par leurs propriétés internes, mais pas leurs propriétés externes, qui permettraient pourtant de vérifier que ces communautés contrastent réellement avec leurs frontières, justifiant d'autant plus leur existence. Des définitions plus récentes prennent en compte ces propriétés, telles que celle de *communauté forte* [158] décrivant un sous-graphe dans lequel le degré interne de chaque nœud est supérieur à son degré externe, pouvant être relâchée en *communauté faible* dans laquelle le degré interne du sous-graphe dans son intégralité doit être supérieur à son degré externe.

Une autre façon de définir une communauté utilise la mesure de ses propriétés globales dans le graphe. Une approche très populaire et récente de cette façon de décrire une communauté est basée sur la comparaison d'un graphe à un modèle de référence, appelé *null model*, dans lequel aucune communauté n'est supposée exister (p.ex. un graphe où chaque arc entre paires de nœuds a la même probabilité d'existence). Newman et Girvan [145] ont proposé un modèle de référence construit à partir du graphe à évaluer en considérant une réorganisation aléatoire des arcs de telle façon à préserver les degrés originaux de chaque nœud en moyenne. Ces travaux ont mené à la définition du critère de *modularité* où l'existence d'une communauté est admise si le degré interne de ses nœuds est supérieur au degré espéré par réorganisation aléatoire dans le modèle de référence, l'écart entre les deux valeurs étant corrélé à la qualité de la communauté. Ces définitions globales ont conduit à toute une classe récente d'algorithmes ayant montré de bons résultats en pratique. Nous nous focaliserons par conséquent sur ces algorithmes dans cette thèse.

Il est important de noter que les approches de partitionnement de nœuds dans un graphe selon le principe de détection de communautés s'opposent conceptuellement aux approches de partitionnement de nœuds selon le principe de la minimisation d'une coupe comme énoncé précédemment. En effet, les partitions n'y sont pas définies par rapport à un nombre de groupes global à découvrir, mais plutôt par la nature de ce qui constitue un groupe. Les concepts de coupe dans un graphe et de communautés ne sont toutefois pas étrangers, des partitions de nœuds en communautés très denses étant souvent liées à des coupes de faible coût. En outre, Newman [144] a récemment proposé une méthode de détection de communautés utilisant les résultats d'un algorithme de partitionnement de graphe.

4.4.2 Modularité et communautés

La recherche d'une bonne partition des nœuds d'un graphe en communautés requiert l'existence d'une fonction d'évaluation permettant de comparer les différentes partitions entre elles. La mesure de qualité la plus populaire est sans doute celle de *modularité* définie par Newman et Girvan [145], basée sur l'idée que la qualité d'une communauté dépend de l'écart entre le degré interne des individus de cette communauté par rapport à ce que l'on obtiendrait dans un graphe de référence n'exhibant aucune communauté. Plus précisément, considérant la matrice d'adjacence A du graphe et la matrice du graphe de référence P contenant pour chaque paire d'individus le nombre espéré d'arcs entre eux dans ce modèle, la modularité est définie de la façon suivante :

$$Q = \frac{1}{2m}\sum_{i,j}(A(ij) - P(ij))\,\delta(C_i, C_j), \tag{4.10}$$

où m est le nombre total d'arcs dans le graphe et $\delta(C_i, C_j)$ est égal à 1 si i et j sont dans la même communauté, 0 sinon.

Un modèle de référence généralement utilisé est le graphe aléatoire dans lequel le même nombre d'arcs que le graphe original est réparti de façon aléatoire, de sorte que l'espérance du degré de chaque nœud soit identique au degré réel dans le graphe original. Dans ce cas, la modularité peut être définie de la façon suivante :

$$Q = \frac{1}{2m}\sum_{i,j}(A(ij) - \frac{k_i k_j}{2m})\delta(C_i, C_j), \tag{4.11}$$

où k_i est le degré du nœud i dans le graphe original.

La mesure de modularité a été inventée et utilisée à l'origine pour évaluer des communautés obtenues par divers algorithmes, sans toutefois intervenir dans les algorithmes eux-mêmes, mis à part pour définir un critère d'arrêt, tel que plusieurs algorithmes divisifs dont le but est de supprimer itérativement les arcs du graphe original en fonction de certaines de leurs propriétés comme leur centralité ou intermédiarité dans le graphe [145]. Au-delà de cet usage initial, de nombreux algorithmes ont rapidement fait de cette mesure un élément central de leur approche.

Une classe d'algorithmes maximisant la modularité propose des approches gloutonnes pour la détection de communauté, afin d'offrir de bonnes performances sur des graphes potentiellement de grande taille. Newman [142] propose un algorithme hiérarchique dans lequel, à partir d'un partitionnement contenant un nœud par groupe, les

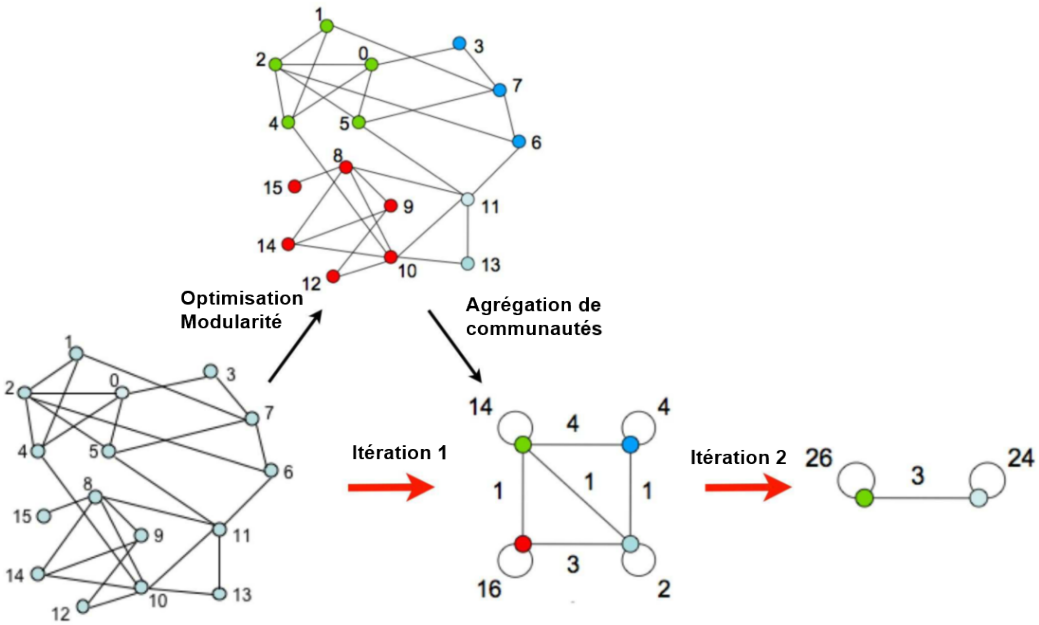


FIGURE 4.2 – Illustration de l'algorithme de Louvain. Image tirée de [16] et traduite en français.

groupes sont itérativement fusionnés les uns aux autres, en fonction de leur connexion par des arcs ou non, ces derniers étant ajoutés au fur et à mesure de telle sorte que l'arc choisi à chaque itération est celui qui mène à la partition maximisant le gain de modularité.

Une approche gloutonne plus récente et montrant des meilleurs résultats est l'algorithme de Louvain [16], dans lequel, toujours en commençant avec chaque nœud dans sa propre communauté, une succession des deux étapes suivantes est effectuée : 1) les nœuds sont tours à tours déplacés dans la communauté menant au meilleur gain de modularité jusqu'à atteindre un optimum local ; 2) tous les nœuds de la même communauté à l'issue de l'étape précédente sont fusionnés pour former un nouveau graphe, dont les arcs entre différents méta nœuds ont pour poids la somme des poids des arcs reliant chaque paire de nœuds constituant respectivement ces méta nœuds à l'étape précédente, et où des arcs réflexifs sont ajoutés pour chaque méta nœud avec un poids représentant la somme des poids des arcs entre nœuds constituant ce méta nœud à l'étape précédente. La figure 4.2 issue de la littérature [16] illustre ces étapes. Une généralisation de l'algorithme de Louvain a récemment été proposée [26] à une classe complète de fonctions d'évaluation séparables.

Des approches plus précises existent pour la détection de communauté basée sur la modularité, mais sont très souvent plus coûteuses. Nous pouvons citer à titre d'exemple les approches basées sur des heuristiques de recuit simulé [77, 76], offrant de bonnes performances, mais souvent non applicables pour des graphes dépassant plusieurs centaines de nœuds, et les méthodes spectrales sur la matrice de modularité B telle que $B(ij) = A(ij) - \frac{k_i k_j}{2m}$ [143], offrant généralement de bons résultats pour réaliser une bissection, mais moins idéales pour un nombre de communautés supérieur à 2.

4.4.3 Cas spécifique des graphes multi partis

La plupart des méthodes énoncées dans cette section tentent de découvrir les meilleures communautés à partir d'un graphe ne disposant d'aucune contrainte concernant les arcs qui y sont admissibles. Pour revenir dans notre contexte relationnel, cette configuration est cohérente dans le cas d'un type d'association non dirigée entre nœuds représentant des entités d'un même type, où n'importe quelle association entre paires d'entités est possible. Toutefois, de nombreux jeux de données présentent en réalité des contraintes sur les arcs, notamment dans le contexte de types d'associations impliquant des entités de types différents. Ces jeux de données sont omniprésents dans la vie réelle et peuvent être modélisés par des hypergraphes multi parties où un hyper arc doit nécessairement relier un tuple de nœuds dont les types correspondent à la définition du type d'association. À titre d'exemple, un type d'association *lit(utilisateur, livre)* est représenté par un graphe biparti où deux types de nœuds coexistent pour chaque type d'entité, et où un arc ne peut exister qu'entre un nœud de chaque type.

Dans le cadre du problème de détection de communautés, les algorithmes utilisés pour des graphes non contraints ne sont pas adaptés aux contraintes multi parties. En effet, considérons d'abord les approches basées sur la recherche de composantes définies strictement, comme les cliques et concepts dérivés décrits ci-dessus. Ces approches ne peuvent fonctionner telles quelles dans un graphe multi parties, car les composantes strictes n'ont pas la même structure dans ces graphes. Pour s'en convaincre, il est intéressant de noter qu'une clique de taille 2 correspond à un triangle dans un graphe non contraint, structure qu'il n'est pas possible de retrouver dans un graphe biparti par exemple. À la place, un équivalent dans un graphe biparti est la recherche de *4-paths clos* [148] correspondant aux groupes de 4 nœuds, dont 2 de chaque type, chacun étant relié aux deux nœuds de l'autre type.

En ce qui concerne les approches basées sur l'évaluation de communautés par une fonction de score visant à comparer une partition par rapport à un modèle de référence, comme les approches utilisant la modularité définies plus haut, celles-ci induisent des coefficients de normalisation qui ne sont plus corrects lorsque les graphes sont contraints. À titre d'exemple, Borgatti et Everett [18] proposent une adaptation de plusieurs mesures d'analyse de graphes pour le cas biparti, p.ex. une adaptation du coefficient de clustering donnant une idée générale sur la tendance des nœuds à se regrouper en communautés. Le score de modularité a lui-même été adapté dans différents articles pour le cas biparti [134, 78, 8].

Il est important de noter que de nombreux articles ont tenté de considérer des projections de graphes multi partis en uni partis afin d'utiliser la grande diversité d'algorithmes existants. Toutefois, cette opération est généralement loin d'être anodine, puisqu'il a été montré que cela ajoutait des structures élémentaires originalement inexistantes (le nombre de triangles obtenus est généralement bien plus grand que celui attendu dans un graphe uni parti de la même taille) amenant des résultats de mesures trop optimistes, même si l'on projette un graphe multi parti généré aléatoirement [146, 74, 75]. De plus, cette opération entraîne une perte d'information notable : une même projection peut avoir pour graphe original de multiples configurations à la sémantique différente [74, 75, 148].

4.4.4 Algorithmes de détection de communautés biparties

Pour pallier les problèmes énoncés ci-dessus, des algorithmes dédiés ont été proposés pour la détection de communautés dans des graphes multi partis et nous nous focalisons ici sur des méthodes fonctionnant sur des graphes bipartis. Barber [8] a proposé une méthode appelée *BRIM* dont le but est d'apprendre de façon alternée les deux matrices d'assignation G_1 et G_2 des individus de chaque type d'entité à un ensemble de k co clusters en maximisant un score de modularité bipartie ayant tour à tour les formes suivantes :

$$Q = \frac{1}{m} \sum_{i=1}^{n_1} \left[\sum_{j=1}^{k} G_1(ij).((B.G_2)(ij)) \right], \tag{4.12}$$

avec G_1 fixée et :

$$Q = \frac{1}{m} \sum_{i=1}^{n_2} \left[\sum_{j=1}^{k} G_2(ij).((B^t.G_1)(ij)) \right], \tag{4.13}$$

avec G_2 fixée, où $B = A - P$ est la matrice rectangulaire de taille $n_1 \times n_2$ contenant pour chaque paire de nœuds la différence entre le poids de l'arc les reliant dans le graphe biparti considéré et le poids de l'arc espéré dans le modèle aléatoire biparti de référence. Une version étendue appelée *adaptative BRIM* a également été proposée dans le même article pour trouver automatiquement le nombre de clusters k. L'approche vise à ré exécuter plusieurs fois BRIM avec différentes valeurs de k pour ensuite trouver le meilleur compromis entre nombre de clusters et modularité bipartie.

Les algorithmes de type BRIM souffrent de problèmes de performances et ne sont donc pas adaptés pour traiter des grands graphes. Une autre approche [159] beaucoup plus rapide, mais sans toutefois garantir la qualité des résultats, appelée *propagation de labels*, a été proposée pour le contexte large échelle. Commençant par un graphe biparti où chaque nœud est étiqueté par sa propre valeur de cluster, cette méthode propage alternativement les labels d'un type d'entité vers les nœuds de l'autre type, par voisinage direct. Le choix du label à conserver pour un nœud correspond alors au mode statistique de l'ensemble des labels reçus. Ce processus est répété jusqu'à convergence.

L'algorithme BRIM et son évolution adaptative sont sensibles à l'initialisation des matrices G_1 et G_2, réalisée aléatoirement, mais l'heuristique utilisée assure un gain de modularité à chaque étape et repose sur une base théorique plus importante. D'un autre côté, l'algorithme de propagation de labels propose un meilleur passage à l'échelle, ne souffre pas de problèmes d'initialisation, et ne nécessite pas de choisir k. Toutefois, il ne garantit pas la qualité des communautés obtenues après convergence. Afin de profiter du meilleur des deux approches, un algorithme appelé *LP-BRIM* [121] a été proposé mêlant propagation de labels et BRIM, utilisant le premier pour initialiser le modèle, et le second pour raffiner cette solution.

Un autre problème posé par la détection de communautés dans les graphes multi partis concerne la validité d'un ensemble de clusters communs pour les différents types d'entités impliqués. En ce sens, Murata [134] a proposé une adaptation de la modularité bipartie permettant de trouver des ensembles de communautés de cardinalité différentes pour chaque type d'entité d'un graphe biparti. La mesure proposée perd toutefois sa propriété de séparabilité, empêchant l'utilisation d'algorithmes gloutons rapides tels que l'algorithme de Louvain généralisé [26].

4.5 Factorisation matricielle avec régularisation laplacienne et coupe minimale dans un graphe biparti augmenté de liens de similarités

Les méthodes de factorisation non négative de matrices (NMF) (cf. 4.2.3) sont des méthodes très populaires dans la littérature, notamment pour le clustering relationnel, et les approches régularisées (cf. 4.2.5) prenant en compte la topologie des données permettent d'enrichir les données initiales pour trouver de meilleurs résultats. Malheureusement, les techniques de type NMF souffrent généralement de problèmes de passage à l'échelle et de sensibilité à l'initialisation. D'un autre côté, plusieurs méthodes de graphe ont été proposées pour gérer de grands jeux de données en un temps très rapide, comme les méthodes multi niveaux (cf. 4.3.4).

Dans cette section, nous proposons de rapprocher les méthodes de factorisation matricielle non négatives régularisées et les méthodes de partitionnement de graphe pour profiter des approches large échelle de cette famille d'algorithmes existant dans la littérature. La première étape de notre contribution vise à montrer une équivalence mathématique entre NMF régularisé par des variétés dans le cas particulier d'un clustering certain à partir d'une matrice binaire, et les méthodes de partitionnement sur un graphe construit comme l'union du graphe relationnel original (décrit par la matrice d'affinité originale) et l'ensemble des graphes de variétés (construits à partir des matrices de similarité). Nous explorons ensuite le partitionnement de graphes régularisés dans le cas de matrices relationnelles moins contraintes (non binaires) et comparons les résultats à ceux obtenus par d'autres méthodes de clustering, incluant une méthode état de l'art de NMF régularisé, et montrons que les méthodes de partitionnement régularisées multi niveaux de graphe peuvent obtenir de meilleurs résultats en un temps considérablement plus court que ces méthodes NMF. Avec l'augmentation croissante du nombre de jeux de données bipartis larges et complexes, notre contribution permet au clustering relationnel de passer plus facilement à l'échelle avec la régularisation par variété.

4.5.1 De l'équivalence de NMF et d'une coupe minimale dans un contexte régularisé

Considérant deux ensembles d'individus, dénotés $\top$ et $\bot$, appelés *modes* de nos données, une matrice A de dimensions $|\top| \times |\bot|$, et des matrices de similarité $W_\top$ et $W_\bot$ entre les individus de chaque mode, la fonction objectif de NMTF doublement régularisée, décrite par [165], est la suivante :

$$\Phi_r = ||A - G_\top S {G_\bot}^t||_F^2 + \lambda_\top \mathbf{tr}({G_\top}^t L_\top G_\top) + \lambda_\bot \mathbf{tr}({G_\bot}^t L_\bot G_\bot) \tag{4.14}$$

où G_i est la matrice facteur du mode i pouvant être interprétée comme une matrice d'assignation de chaque individu du mode à son ensemble de clusters, L_i est le laplacien de W_i et λ_i contrôle l'importance de L_i dans le résultat final.

Le graphe d'affinités correspondant entre les modes est dénoté $\mathcal{G}(\top \cup \bot, E)$ où un arc existe dans $E \subseteq \top \times \bot$ pour chaque paire d'individus correspondant à un élément

non nul de A. La matrice d'adjacence de ce graphe est :

$$\tilde{A} = \begin{bmatrix} 0 & A^t \\ A & 0 \end{bmatrix}.$$

Le calcul de la coupe totale, tel que défini en (4.9), pour un graphe partitionné en k groupes peut être représenté sous une forme matricielle [50]. Considérant en effet la matrice d'adjacence du graphe $\tilde{A}$ de dimensions $|\top \cup \bot| \times |\top \cup \bot|$, et une partition P_V des noeuds, la coupe totale correspond à la trace $\mathbf{tr}(\widetilde{V}^t L \widetilde{V})$ où L est le laplacien de $\tilde{A}$ et $\widetilde{V}$ est la matrice indicatrice de P_V de dimensions $|\top \cup \bot| \times k$ avec :

$$\widetilde{V}(ic) = \begin{cases} 1 \text{ si le } i^{me} \text{ individu appartient au } c^{ime} \text{ cluster,} \\ 0 \text{ sinon.} \end{cases}$$

Le théorème suivant montre que la minimisation de Φ_r sous contrainte de clustering certain pour une matrice binaire est équivalente à la recherche d'un partitionnement à la coupe minimale sur le graphe $\mathcal{G}(\top \cup \bot, E_r \cup E_\top \cup E_\bot)$, où $E_r \subseteq \top \times \bot$ (resp. $E_\top \subseteq \top \times \top$ et $E_\bot \subseteq \bot \times \bot$), sont les ensembles d'arcs inter(resp. intra)-modes, modulo des ajustements de poids.

Théorème 1. *Soit $\top$ et $\bot$, deux ensembles d'entités et A la matrice d'affinité de dimensions $|\top| \times |\bot|$ entre ces deux ensembles, avec $\forall (i,j) : A(ij) \in \{0,1\}$. Soient $W_\top$ et $W_\bot$ les matrices de similarité intra modes, leurs laplaciens $L_\top$ et $L_\bot$ et deux paramètres de régularisation non négatifs $\lambda_\top$ et $\lambda_\bot$. Supposant que nous souhaitons trouver un co clustering certain des deux modes dans le même ensemble de clusters, alors minimiser (4.14) est équivalent à un problème de minimisation de la coupe dans le graphe défini par :*

$$X = \begin{bmatrix} \lambda_\top W_\top & A \\ A^t & \lambda_\bot W_\bot \end{bmatrix}.$$

Démonstration. Considérant la fonction objectif de NMF définie précédemment, nous pouvons la réécrire de la façon suivante :

$$\begin{aligned} \Phi_r = \ & \mathbf{tr}(A^t A + G_\bot S^t G_\top{}^t G_\top S G_\bot{}^t - G_\top S G_\bot{}^t A^t - G_\bot S^t G_\top{}^t A) \\ & + \sum_{i \in \{\top, \bot\}} \lambda_i \mathbf{tr}(G_i{}^t L_i G_i) \end{aligned}$$

Puisque $A^t A$ est une constante, elle n'a pas d'impact sur la minimisation. De plus, en utilisant les propriétés de somme et de transposition d'une trace de matrice, nous pouvons écrire :

$$\begin{aligned} \arg\min \Phi_r = \arg\min \ & \mathbf{tr}(G_\bot S^t G_\top{}^t G_\top S G_\bot{}^t) - 2\mathbf{tr}(G_\top S G_\bot{}^t A^t) \\ & + \sum_{i \in \{\top, \bot\}} \lambda_i \mathbf{tr}(G_i{}^t L_i G_i) \end{aligned}$$

Le premier terme, $tr(G_\bot S^t G_\top{}^t G_\top S G_\bot{}^t) = tr((G_\top S G_\bot{}^t)^t (G_\top S G_\bot{}^t))$ est la norme de la matrice de reconstruction $||G_\top S G_\bot{}^t||_F^2$. De ce fait, la fonction d'optimisation tend à baisser la norme globale de la matrice de reconstruction, privilégiant ainsi des

matrices creuses. Les deux derniers termes de régularisation peuvent aussi être légèrement modifiés pour incorporer les paramètres de régularisation dans les matrices laplaciennes, ce qui nous donne :

$$\arg\min \Phi_r = \arg\min \quad ||G_\top S {G_\perp}^t||_F^2 - 2\mathbf{tr}(G_\top S {G_\perp}^t A^t) + \sum_{i\in\{\top,\perp\}} \mathbf{tr}({G_i}^t \lambda_i L_i G_i)$$

Dans le cas d'un clustering certain, quand les matrices G_i décrivent des assignations certaines pour les deux modes, les ensembles de clusters $C_\top$ et $C_\perp$ ont la même cardinalité et S décrit une assignation des clusters de $C_\top$ aux clusters de $C_\perp$ (S est alors une matrice de permutation), définissant $\widetilde{G}_\perp = G_\perp S^t$, nous pouvons réécrire la fonction objectif d'une façon plus compacte :

$$\arg\min \Phi_r = \arg\min \quad ||G_\top \widetilde{G}_\perp^t||_F^2 + \mathbf{tr}\left(\begin{bmatrix} {G_\top}^t & \widetilde{G}_\perp^t \end{bmatrix} \begin{bmatrix} \lambda_\top L_\top & -A \\ -A^t & \lambda_\perp L_\perp \end{bmatrix} \begin{bmatrix} G_\top \\ \widetilde{G}_\perp \end{bmatrix}\right) \tag{4.15}$$

L'équivalence est valable parce que nous avons :

$$\mathbf{tr}({G_\perp}^t L_\perp G_\perp) = \mathbf{tr}(\widetilde{G}_\perp^t L_\perp \widetilde{G}_\perp) = \mathbf{tr}(S {G_\perp}^t L_\perp G_\perp S^t)$$

quand S est une matrice de permutation. Le second terme de (4.15) est proche d'une matrice laplacienne, la seule différence se trouvant au niveau de la diagonale, puisque les matrices de régularisation laplaciennes impliquent uniquement les degrés intra mode. Toutefois, puisque les degrés inter mode sont constant sachant A, le problème de minimisation ne change pas lorsque l'on ajoute ces degrés inter modes. Finalement, nous pouvons écrire :

$$\begin{aligned} \arg\min \Phi_r &= \arg\min ||G_\top \widetilde{G}_\perp^t||_F^2 + \mathbf{tr}\left(\begin{bmatrix} {G_\top}^t & \widetilde{G}_\perp^t \end{bmatrix} (D - X) \begin{bmatrix} G_\top \\ \widetilde{G}_\perp \end{bmatrix}\right) \\ &= \arg\min ||G_\top S {G_\perp}^t||_F^2 + \mathbf{tr}\left(\begin{bmatrix} {G_\top}^t & \widetilde{G}_\perp^t \end{bmatrix} L \begin{bmatrix} G_\top \\ \widetilde{G}_\perp \end{bmatrix}\right) \end{aligned}$$

où :

$$X = \begin{bmatrix} \lambda_\top W_\top & A \\ A^t & \lambda_\perp W_\perp \end{bmatrix},$$

$D = D^{inter} + D_\top + D_\perp$ la matrice de degrés de X, et L est le laplacien de X. La fonction objectif est donc équivalente à un problème de coupe minimale dans X, modulo un terme privilégiant les solutions creuses, sous hypothèses énoncées précédemment, ce qui conclut notre preuve. □

4.5.2 Discussion

Il est important de noter que, même si les fonctions objectif de NMF régularisé et de la coupe minimale d'un graphe augmenté par les similarités sont équivalentes sous hypothèses de clustering certain pour des matrices binaires, avec un ensemble de clusters partagé, les résultats obtenus en pratique sont également sensibles aux heuristiques des algorithmes qui sont elles différentes. De ce fait, il n'est pas évident, sinon impossible, d'observer une équivalence stricte de façon empirique, même si nous travaillons dans

Jeu de données	$\lvert\top\rvert$	$\lvert\perp\rvert$	$\lvert C_\top\rvert = \lvert C_\perp\rvert$
Glass	214	9	6
Heart	270	13	2
Semeion	1593	256	10
Soybean	47	35	4
SPECTF	267	45	2
Vehicle	846	19	4
Wine	178	13	3
Wpbc	198	33	2

FIGURE 4.3 – Les huit jeux de données UCI.

le cadre contrôlé de nos hypothèses.

En outre, les hypothèses en elles-mêmes sont difficiles à satisfaire en pratique et ne correspondent pas nécessairement au cas commun. Tout d'abord, le cas d'une matrice binaire n'est valable que pour représenter un graphe biparti non pondéré entre deux ensembles d'individus. Ce cas est également rencontré lorsque des individus d'un mode sont décrits par des critères nominaux, auquel cas la matrice doit correspondre à l'affectation de chaque individu à ces valeurs correctes de caractères après transformation de ces derniers par codage disjonctif complet. Ensuite, l'hypothèse visant à faire partager le même ensemble de clusters pour les lignes et colonnes de la matrice est connue comme étant sous-optimale [51], une des raisons qui ont conduit à l'ajout d'un troisième facteur dans les algorithmes de type NMF. Finalement, le clustering certain n'est pas toujours adapté, comme dans le cas de l'extraction de thématiques à partir de documents. Dans ce cas, une approche par aspect [85, 86] semble plus appropriée.

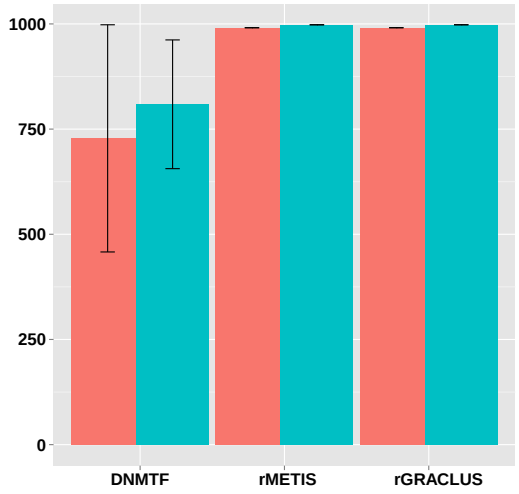
4.5.3 Expériences

Dans les expériences qui suivent, nous commençons par illustrer notre propos concernant les divergences de résultats des approches prouvées identiques à cause des différences d'heuristiques, sur un environnement synthétique reproduisant fidèlement les hypothèses théoriques. Nous décidons ensuite de poursuivre les expériences sur des jeux de données réels, mais qui ne sont pas nécessairement contraints aux hypothèses susmentionnées, pour ainsi voir le comportement de différents algorithmes dans ces cas plus généraux.

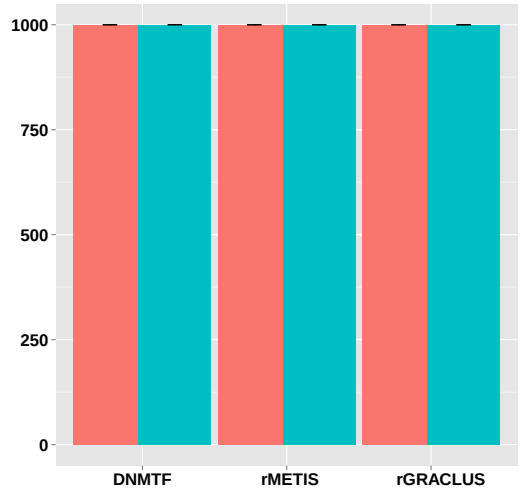
i Données expérimentales

La première étape de nos expériences vise à reproduire fidèlement le contexte de notre démonstration précédente, impliquant un ensemble de co-clusters communs pour les lignes et colonnes à partitionner de la matrice originale, des poids binaires, ainsi que des clusters bien distincts. Pour ce faire, nous synthétisons des matrices dont les règles de remplissage sont gérées par blocs. Pour chacun de ces blocs, une probabilité est définie paramétrant l'existence d'un 1 dans chacune des cases de ce bloc.

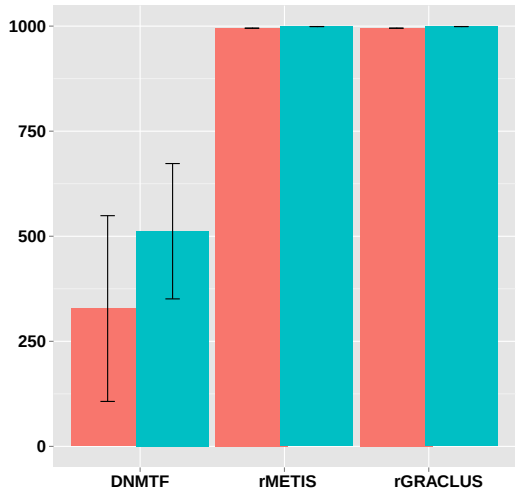
Plus formellement, soient $\top$ et $\perp$ fixés, $C = C_\top = C_\perp$ l'ensemble de clusters commun, les fonctions de partition $P_\top$ et $P_\perp$ prédéterminées, et une matrice à générer X de dimensions $|\top| \times |\perp|$, nous définissons $|C|^2$ blocs de cette matrice X_{IJ} pour



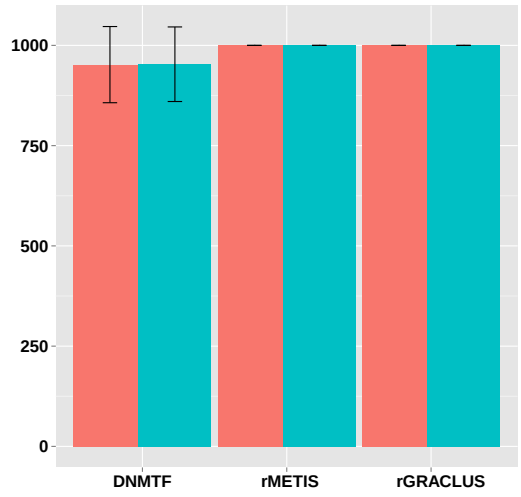
$p_{in} = 0.4, p_{ext} = 0.2$



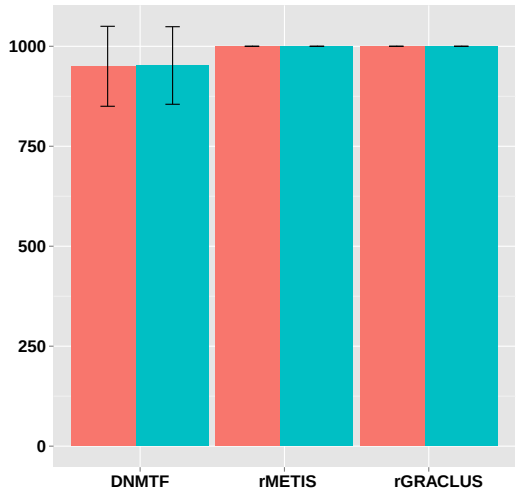
$p_{in} = 0.6, p_{ext} = 0.2$



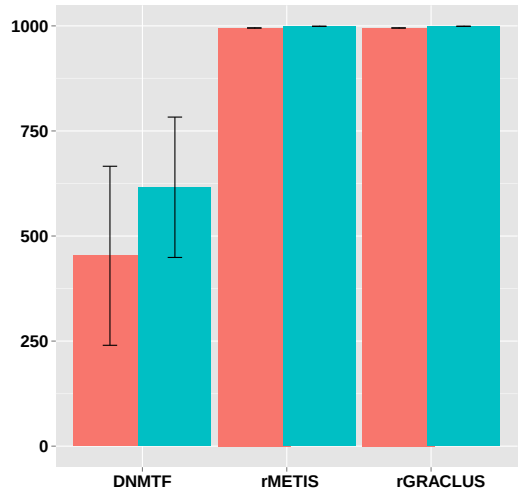
$p_{in} = 0.6, p_{ext} = 0.4$



$p_{in} = 0.8, p_{ext} = 0.2$



$p_{in} = 0.8, p_{ext} = 0.4$



$p_{in} = 0.8, p_{ext} = 0.6$

FIGURE 4.4 – Meilleures moyennes des évaluations NMI (rouge, gauche) et ACC (bleu, droite) + écarts-types ($\times 10^{-3}$) pour les jeux de données synthétiques correspondant aux différentes configurations de p_{in} et p_{ext}.

chaque paire de clusters $(I, J) \in \{1, \ldots, |C|\}^2$. À chacun de ces blocs est associée une probabilité $p_{IJ} \in [0, 1]$ de sorte que la probabilité de tout élément de X vérifie :

$$\forall (i, j) \in \{1, \ldots, |C|\}^2 : \top_i \in C_I, \bot_j \in C_J \rightarrow P(X(ij) = 1) = p_{IJ}.$$

Dans notre expérience, nous considérons que les co-clusters correspondent aux blocs de la forme X_{II} et réduisons le nombre de paramètres à 2 pour n'avoir qu'une probabilité p_{in} que deux individus d'un même co-cluster soient liés par un poids de 1 dans la matrice, et p_{ext} caractérisant la probabilité que deux individus de clusters différents aient un poids de 1 entre eux. Plus précisément, nous étudions diverses matrices correspondant à des valeurs de p_{in} et p_{ext} prises dans la grille $\{0.2, 0.4, 0.6, 0.8\}$, en ne conservant que les cas où $p_{in} > p_{ext}$.

Pour chacune des matrices générées, nous exécutons trois implémentations d'algorithmes différents :

1. graph dual regularization non-negative matrix tri-factorization (DNMTF) [165], un algorithme état de l'art pour la famille des algorithmes NMF **régularisés**, où les matrices de variété sont construites par une approche à base de k plus proches voisins. Nous proposons ici notre propre implémentation de l'algorithme ;

2. METIS[1] [94], un algorithme célèbre de partitionnement de graphe, que nous exécutons sur une version du **graphe augmentée** telle que définie précédemment (dénoté rMETIS) ;

3. GRACLUS[2] [50], un autre algorithme célèbre de partitionnement de graphe, que nous exécutons de la même façon sur une version **augmentée du graphe** (dénoté rGRACLUS).

Tous les algorithmes considérés définissent leurs matrices de similarité $W_\top$ et $W_\bot$ en choisissant un nombre $n_\top$ (resp. $n_\bot$) de plus proches voisins pour chaque individu de $\top$ (resp. $\bot$), assignant 1 à chaque élément des matrices correspondant à des paires d'individus dont chacun contient l'autre parmi ses plus proches voisins, et 0 dans le cas contraire. Pour rappel, les deux méthodes de partitionnement de graphes proviennent de la littérature, mais leurs résultats dans ce chapitre sont obtenus par application de ces algorithmes sur des graphes augmentés d'arcs intra mode pondérés par les similarités susmentionnées. Les paramètres $n_\top$ et $n_\bot$ sont chacun pris dans la grille $\{1, 2, 3, 4, 5, 6, 7, 9, 11\}$ et les paramètres de régularisation $\lambda_\top$ et $\lambda_\bot$ sont chacun pris dans la grille $\{1, 10, 100, 500\}$. Chaque combinaison de paramètres est évaluée à 5 reprises.

Les algorithmes sont évalués par leurs moyennes des indicateurs que sont l'information mutuelle normalisée (NMI) et la précision de clustering (ACC), obtenues sur les 5 itérations, sur chaque jeu de données. Nous conservons la meilleure moyenne obtenue sur l'ensemble des configurations de paramètres, ainsi que l'écart-type des résultats de l'exécution retenue sur les 5 répétitions correspondantes, et les temps de calcul associés. Suivant les conclusions de la section 4.5.1, nous nous attendons à obtenir des résultats de partitionnement sur les graphes régularisés (rMETIS, rGRACLUS) proches de ceux obtenus avec DNMTF, avec des temps d'exécutions moins longs. Les

[1] Implémentation officielle : http://glaros.dtc.umn.edu/gkhome/metis/metis/download
[2] Implémentation officielle : http://www.cs.utexas.edu/users/dml/Software/graclus.html

résultats sont données dans la figure 4.4 pour chaque triplet (jeu de données, algorithme, indicateur).

Nous pouvons voir que dans la moitié des cas, correspondant aux cas où p_{in} et p_{ext} sont suffisamment éloignés pour que les clusters soient aisément identifiables, les trois algorithmes obtiennent de bonnes performances. Toutefois, DNMTF obtient moins généralement une moyenne élevée, montrant les problèmes de convergence vers un optimum local. Dans les trois autres cas plus complexes, DNMTF converge vers des optima locaux de façon systématique, alors que METIS et GRACLUS, grâce aux informations de variété, peuvent obtenir un graphe simplifié respectant la topologie initiale et permettant d'éviter de tomber dans ces sous-optima. Nous pouvons donc ainsi constater l'écart de comportement entre les deux familles d'algorithmes malgré l'équivalence de leurs objectifs dans ce contexte précis.

ii Données réelles

Nous utilisons les résultats théoriques et expérimentaux précédents comme point de départ pour explorer l'efficacité du partitionnement de graphe régularisé pour le clustering relationnel. Comme énoncé précédemment, les hypothèses de matrice binaire, de matrice de permutation S et d'ensembles de clusters communs ne sont pas toujours réalistes. Aussi, nous réalisons nos expériences dans un contexte avec des contraintes plus faibles. Plus précisément, nous comparons les performances et temps d'exécution de six algorithmes, dont deux méthodes de partitionnement de graphe, sur 8 jeux de données UCI [5] dont les caractéristiques sont résumées dans la Table 4.3.

Les algorithmes comparés sont les mêmes que précédemment, ainsi que :

1. k-means selon l'algorithme de Lloyd [122], sur la matrice originale, **sans régularisation** ;
2. projected gradient NMF [119], un algorithme récent pour la résolution de NMF **non régularisé** ;
3. normalized cut (NCut) [166], un algorithme de clustering spectral tentant de trouver la coupe minimale dans le **graphe non augmenté**, tout en conservant un équilibre entre les tailles de cluster. Afin de faciliter la comparaison entre les algorithmes, nous paramétrons l'algorithme afin qu'il utilise une approche à base de k plus proches voisins pour construire la matrice d'affinités. Il est important de noter que seule la variété des individus est considérée.

Ces trois algorithmes sont ajoutés ici pour fournir des résultats de référence pour différentes familles d'algorithmes. Ils sont exécutés sans régularisation ou augmentation du graphe considéré, contrairement aux trois autres. Les implémentations utilisées pour ces trois algorithmes sont celles fournies par la bibliothèque python open-source *scikit-learn*[3] [154], proposant de nombreuses implémentations d'algorithmes d'apprentissage automatique et de fouille de données, et à la communauté active.

Tout algorithme concerné par la régularisation laplacienne dans ces expériences (c.-à-d. NCut, DNMTF, rMETIS, rGRACLUS) définit ses matrices de similarité $W_\top$

[3] http://scikit-learn.org/stable/

Data	Eval.	K-means	NMF	NCut	DNMTF	rMET	rGRA
Glass	NMI	340±4	270±60	339±2	335±21	**407±0**	395±0
	ACC	536±15	514±36	625±2	592±25	**696**±0	**696±0**
	Time	10±0	850±170	70±0	90±20	50±0	30±0
Heart	NMI	217±96	70±18	100±0	217±39	**323±0**	234±0
	ACC	753±80	654±16	652±0	754±34	**819±0**	778±0
	Time	0±0	140±50	160±10	570±0	30±0	30±0
Semeion	NMI	540±10	334±11	604±5	377±6	664±0	**709**±0
	ACC	627±10	409±16	636±0	458±25	747±0	**788**±0
	Time	390±120	3560±1130	1910±100	53560±460	290±10	190±10
Soybean	NMI	879±110	888±33	**1000±0**	954±59	**1000±0**	**1000±0**
	ACC	915±81	940±28	**1000±0**	974±34	**1000±0**	**1000±0**
	Time	0±0	180±80	30±0	120±30	40±0	30±0
SPECTF	NMI	78±7	94±20	117±0	92±14	165±0	**173±0**
	ACC	794±0	794±0	794±0	**795±1**	794±0	794±0
	Time	0±0	140±30	70±10	900±0	40±0	50±0
Vehicle	NMI	101±0	33±10	159±0	136±24	245±0	**250±0**
	ACC	405±1	330±13	447±0	429±34	**515±0**	508±0
	Time	20±10	430±70	170±0	4000±930	50±0	50±0
Wine	NMI	837±27	697±25	907±0	722±31	895±0	**921±0**
	ACC	951±11	892±20	978±0	908±14	966±0	**978±0**
	Time	0±0	100±10	30±0	280±160	40±0	30±10
Wpbc	NMI	19±6	23±4	31±1	47±8	**60±0**	57±0
	ACC	763±0	763±0	763±0	**771±5**	763±0	768±0
	Time	0±0	70±10	40±0	590±0	40±10	30±0

TABLE 4.1 – Meilleures moyennes des évaluations NMI et ACC + écarts-types ($\times 10^{-3}$). Les temps de calcul correspondants sont également donnés (en ms.).

et $W_\perp$ en choisissant un nombre $n_\top$ (resp. $n_\perp$) de plus proches voisins pour chaque individu de $\top$ (resp. $\perp$), assignant 1 à chaque élément des matrices correspondant à des paires d'individus dont chacun contient l'autre parmi ses plus proches voisins, et 0 dans le cas contraire. Pour rappel, les deux méthodes de partitionnement de graphes au cœur de ces expériences proviennent de la littérature, mais leurs résultats dans ce chapitre sont obtenus par application de ces algorithmes sur des graphes augmentés d'arcs intra mode pondérés par les similarités susmentionnées. Les paramètres $n_\top$ et $n_\perp$ sont chacun pris dans la grille $\{1, 2, 3, 4, 5, 6, 7, 9, 11\}$ et les paramètres de régularisation $\lambda_\top$ et $\lambda_\perp$ sont chacun pris dans la grille $\{1, 10, 100, 500\}$. Chaque combinaison de paramètres est évaluée à 5 reprises.

Les algorithmes sont évalués par leurs moyennes des indicateurs que sont l'information mutuelle normalisée (NMI) et la précision de clustering (ACC), obtenues sur les 5 itérations, sur chaque jeu de données normalisé par colonne, étant donné le bon nombre de clusters. Les meilleures moyennes, leurs écarts-types et temps de calcul sont donnés dans la table 4.1 pour chaque triplet (jeu de données, algorithme, indicateur).

Nous pouvons voir que METIS et GRACLUS régularisés par les variétés obtiennent de bonnes performances. De façon surprenante, ils obtiennent la plupart du temps les

meilleures performances, à la fois pour NMI et ACC. Ceci peut s'expliquer par la variance forte des résultats de DNMTF, même en considérant une seule configuration des paramètres, puisque nous évaluons la meilleure moyenne obtenue sur plusieurs itérations pour ces critères. Dans quelques cas toutefois, DNMTF obtient les meilleures performances. Dans ce cas, les algorithmes de partitionnement obtiennent des résultats proches. Nous pouvons également voir que les performances de NCut sont bonnes, mais souvent en dessous des méthodes de partitionnement sur le graphe augmenté. Ce résultat est intéressant, puisque NCut utilise uniquement l'information de voisinage d'un seul des deux modes. Ainsi, il semble que l'usage des deux variétés au lieu d'une seule, en plus des informations de la relation, soit profitable.

En termes de temps d'exécution, DNMTF consomme significativement plus de temps que les algorithmes de partitionnement de graphe, et ces temps augmentent significativement avec la taille des matrices. À titre d'exemple, le temps moyen obtenu sur le jeu de données Semeion est d'environ 50 secondes pour DNMTF, contre un temps inférieur à 0.5 seconde pour rGRACLUS et rMETIS. Ceci confirme notre affirmation de départ et montre l'intérêt de substituer l'usage d'une méthode NMF régularisée par un algorithme de partitionnement de graphe multi niveaux, plus rapide.

4.5.4 Perspectives

Dans cette section, nous avons tout d'abord démontré que la fonction objectif de NMF avec une régularisation laplacienne était mathématiquement équivalente à celle d'un partitionnement de graphe augmenté par ces mêmes informations de similarités, dans le cas d'un clustering certain, avec des ensembles de clusters égaux et sur une matrice originale binaire. Nous avons ensuite montré empiriquement que les résultats obtenus dans le cadre restreint des hypothèses de démonstration sont différents, en dépit de l'équivalence de leurs fonctions objectif. Partant de ce constat, nous avons étendu notre cadre expérimental et montré l'efficacité des méthodes de partitionnement sur des graphes augmentés dans un cas plus général où les matrices originales ne sont plus nécessairement binaires. Cette efficacité est double puisqu'elle permet d'obtenir des résultats au moins proches, mais souvent meilleurs qu'avec un algorithme NMF régularisé, en un temps significativement plus court. Les perspectives pour ce travail incluent l'exploration de plus d'aspects théoriques pour l'équivalence entre les deux approches, notamment dans le cas où nous relâchons les hypothèses effectuées ici. En outre, d'autres régularisations que les approches par k plus proches voisins peuvent être inspectées, par exemple dans le cas où les individus de chaque mode ont des descriptions par attributs et où il est possible de calculer des similarités à partir de ces descriptions.

4.6 Conclusion

Suite à nos conclusions concernant les modèles définis dans le chapitre précédent, nous nous sommes intéressés dans ce chapitre aux algorithmes de clustering relationnel de différentes familles, à savoir des algorithmes de co clustering, incluant notamment les approches par factorisation matricielle, des algorithmes de partitionnement de graphe, avec une attention particulièrement apportée aux approches multi niveaux, et des algorithmes de détection de communautés dans les graphes, nous concentrant plus particulièrement sur les approches basées sur l'optimisation de critères de modularité. Pour les approches orientées graphes, qui ne sont pas toujours naturellement faites pour des

données multi parties, nous nous sommes intéressés aux problématiques induites par la nature multi partie des données sur des algorithmes dans un cas général, et avons décrit plusieurs solutions dédiées à ces types de données dans la littérature. Nous avons enfin terminé ce chapitre par une contribution concernant la recherche d'une équivalence entre les problèmes de factorisation de matrice régularisée dans des variétés, et de partitionnement dans un graphe augmenté. Une version préliminaire de cette contribution a conduit à la publication d'un article de recherche [40].

Dans le chapitre suivant, nous abordons notre première extension aux modèles relationnels probabilistes avec incertitude de référence comblant une partie des limites énoncées dans le chapitre précédent, notamment en terme de partitionnement, à la lumière des approches de clustering relationnel décrites dans ce chapitre.

5

Vers un apprentissage relationnel des fonctions de partition

Sommaire

5.1 Introduction

Dans le chapitre 3, nous évoquons une limite majeure des MRP-IR concernant l'utilisation des seules informations de description des différentes entités selon leurs attributs pour l'apprentissage des fonctions de partition et l'absence totale de prise en compte de l'information des associations pour lesquelles nous cherchons à découvrir des partitions. Pourtant, des résultats en clustering ont affirmé qu'il était souvent préférable d'utiliser des méthodes de clustering relationnel, afin de trouver une information des groupes fidèle au type d'association considéré, prenant en compte les regroupements mutuels des entités durant le processus. Ce constat nous amène à penser qu'il est nécessaire de faire évoluer la définition des MRP-IR pour améliorer la qualité des connaissances découvertes.

Dans ce chapitre, nous cherchons à confirmer l'hypothèse stipulant que l'utilisation d'algorithmes de partitionnement relationnels permet d'apprendre des modèles plus précis que l'utilisation d'algorithmes utilisant les seuls attributs des entités. Nous proposons pour cela une première extension des modèles relationnels probabilistes avec incertitude de références, où nous relâchons les contraintes des MRP-IR visant à restreindre l'utilisation des seuls attributs d'un type d'entité pour partitionner ses instances, et impliquant un déterminisme forçant deux individus ayant les mêmes attributs à se retrouver dans le même cluster. Nous réalisons ensuite des expériences où nous comparons l'apprentissage de paramètres des MRP-IR et des MRP-IR+ avec des méthodes de clustering différentes pour une structure donnée.

Au fil des sections, nous explorons différentes solutions de clustering, parmi celles définies dans le chapitre 4, pour apprendre les fonctions de partition des modèles proposés, et analysons au fur et à mesure les avantages et inconvénients d'utiliser chaque méthode, par le biais des résultats des expériences.

Une version préliminaire de ce travail, impliquant la comparaison de l'apprentissage de paramètres d'un MRP-IR avec d'une part une approche de clustering relationnel par factorisation matricielle et d'autre part une approche de clustering historique par attributs, a fait l'objet d'une publication, présentée lors de la conférence FLAIRS 27, en 2014 [41].

5.2 MRP-IR+

Un MRP-IR+ est proche d'un MRP-IR. Il possède en effet la même structure graphique et la même définition de paramètres que pour les MRP-IR historiques, définis dans le chapitre 3. De ce fait, le MRP-IR+ complété et donc le graphe de dépendance acyclique s'obtiennent de la même façon. Un MRP-IR+ peut toujours être instancié en considérant le squelette objet d'une instance, et par conséquent la création d'un RBP à partir d'un MRP-IR+ et d'une instance suit les mêmes règles que pour les MRP-IR.

Chaque variable sélecteur S_{ij} d'un MRP-IR+ est toujours associée à une fonction de partition $\Phi_{ij} : pk(\mathcal{I}(\mathcal{R}^i)) \rightarrow \{1, \ldots, c_{ij}\}$, possédant des attributs de fonction de partition $\mathcal{A}_{ij}$.

En revanche, celle-ci n'est plus nécessairement exprimée de la façon suivante : $\Phi_{ij} : \mathcal{A}_{ij} \rightarrow \{1, \ldots, c_{ij}\}$. En outre, contrairement à ce qui est fait dans les MRP-IR, nous découplons dans les MRP-IR+ les concepts de fonction de partition, contextuel à une variable sélecteur, de la notion d'algorithme de partitionnement, contextuel à un type d'association dans son intégralité. Ainsi, un algorithme de co-clustering pour un type d'association résulte à lui seul en la définition des fonctions de partition de tous les sélecteurs de ce type. Cet algorithme utilise principalement l'information des associations entre entités pour produire les partitions, au lieu des informations d'attributs des entités seules. Un ensemble d'attributs peut toujours être donné pour chaque type d'entité impliqué, mais cette information est utilisée principalement pour régulariser l'information des associations.

5.3 Propriétés des types d'association

Il est important de caractériser les différents types d'association que l'on peut généralement rencontrer dans les jeux de données relationnels afin de choisir au mieux les algorithmes de partitionnement à utiliser. En effet, tous les types d'association ne possèdent pas les mêmes propriétés et les méthodes de clustering relationnel ne sont pas nécessairement adaptées à l'ensemble des types d'association existant.

Tout d'abord, un type d'association possède une *arité*, déterminant le nombre de références que contient chaque association de ce type. Toutes les arités positives peuvent être rencontrées en théorie. Une arité de 1 correspond à un type d'association fonctionnel, sur lequel nous ne pouvons réaliser de co-clustering. Nous ne nous intéressons pas à ce cas ici et redirigeons le lecteur vers les techniques utilisées pour les MRP-IR historiques, le cas échéant. Une instance de type d'association d'arité 2 peut être représentée par un graphe, où chaque nœud correspond à une entité, contenant un arc (ou arête) par association de l'instance reliant les entités concernées. Une instance de type d'association d'arité $n > 2$ correspond à un hyper graphe où un hyper arc (ou hyper arête) relie les n entités concernées par une association. Notons qu'une instance d'association d'arité 2 peut également être représentée par une matrice et les arités plus élevées par un tenseur d'ordre supérieur.

Ensuite, un type d'association peut contenir plusieurs sous-ensembles distincts d'individus, appelés *modes* des données. Un type d'association d'arité supérieure à 1 est alors appelé *uni-modal* ou *uni partie* ssi toutes les contraintes de référence ciblent le même type d'entité et donc possèdent le même domaine de définition. Dans le cas contraire, le jeu de données est appelé *multimodal* ou *multi parties* (*biparties* pour deux types d'entités référencés, etc.). Notez qu'arité et nombre de modes ne sont pas équivalents. Nous pouvons ainsi avoir des types d'association d'arité 2 uni partie. Dans ce cas, une instance est représentée par un graphe où n'importe quelle paire de nœuds peut être connectée.

Une troisième propriété importante pour un type d'association est d'être *dirigé* ou *non dirigé*. Dans le cas d'un type d'association non dirigé, la nature des contraintes de référence n'a pas d'importance. Ceci se matérialise concrètement pour une instance par un (hyper-)graphe non dirigé. Dans le cas contraire, une instance se matérialise par un (hyper-)graphe dirigé. La différence est importante particulièrement lorsque le type d'association possède un nombre de modes inférieur à son arité. Dans ce cas, un type

d'association non dirigée indique que l'on ne recherche qu'un seul partitionnement pour chaque type d'entités (autant de partitions que le nombre de modes), alors que dans le cas dirigé un partitionnement doit exister par contrainte de référence (autant de partitions que l'arité), correspondant à différents "rôles" dans le type d'association devant être considérés différemment. Autrement dit, un type d'association dirigé correspond à un type d'association dans lequel un mode est créé dans les données pour chaque contrainte de référence, pouvant se matérialiser par un (hyper-)graphe multi parties pouvant contenir potentiellement plusieurs nœuds pour chaque entité dont le type est ciblé par plusieurs contraintes de références.

Ainsi, un type d'association entre personnes indiquant un lien d'amitié entre elles peut être vu comme non dirigé, auquel cas un seul partitionnement de personnes est souhaité, correspondant aux communautés d'amis du jeu de données. Toutefois, une version dirigée de ce type d'association, comme celle indiquant qui *suit* l'activité de qui sur un réseau social, peut induire de trouver deux partitionnements différents : les communautés des personnes en fonction des personnes qu'ils *suivent* sont différentes des communautés de personnes en fonction des personnes suivies.

Les trois propriétés énoncées précédemment concernent essentiellement la nature des contraintes de référence et leur sémantique à l'égard du type d'association, mais il est également intéressant de considérer les différentes propriétés des types d'association à l'égard de leurs attributs descriptifs. Nous pouvons en effet rencontrer les cas suivants :

1. dans les cas les plus simples, il n'existe pas d'attribut dans le type d'association et l'information portée par une instance correspond à un (hyper-)graphe non pondéré ou un tenseur d'affinités uniquement composé de 0 et de 1 ;

2. dans les cas où un attribut descriptif aux valeurs continues et ordonnées existe, l'information d'une instance décrit un graphe pondéré où un tenseur d'affinités pondéré ;

3. dans les cas où un attribut descriptif aux valeurs ordinales existe, tel que pour l'ensemble de valeurs {"fort", "moyen", "faible"}, il est possible de projeter ces valeurs sur la droite des réels, de telle sorte à pouvoir revenir au cas précédent. Cette projection peut se faire conjointement avec un expert, dans le cas où l'écart entre les différentes modalités n'est pas constant ;

4. dans les cas où le type d'association prend des valeurs nominales pour lesquelles il existe un ordre partiel ou aucun ordre, l'information portée par une instance correspond à un (hyper-)graphe étiqueté. Il peut être nécessaire de transformer un type d'association comportant un attribut nominal à p valeurs en p types d'association sans attribut descriptif par codage disjonctif complet, car les différentes valeurs peuvent potentiellement décrire des sémantiques d'associations différentes ;

5. dans les cas où le type d'association possède plusieurs attributs, il peut être utile de modéliser chaque information séparément en fonction des propriétés des attributs par rapport aux cas énumérés ici, entraînant la création de nouveaux types d'association comme pour le cas précédent. Cette approche divisive semble à première vue faire perdre de l'information, mais ceci part de l'hypothèse que

des attributs dans un même type d'association sont corrélés, ce qui n'est pas nécessairement le cas. La division en plusieurs types d'association permet ainsi de mettre à plat les informations correspondantes à partir desquelles des corrélations peuvent être possiblement observées.

5.4 NMF et MRP-IR+

Considérant les différentes contraintes qui peuvent s'exprimer dans les types d'association, notre premier choix s'est tourné vers les méthodes de factorisation non négatives de matrices pour réaliser du clustering relationnel dans le cadre d'un apprentissage de MRP-IR+.

En effet, des variantes de ces méthodes ont été créées pour gérer la majorité des propriétés énoncées précédemment. En se restreignant aux types d'association d'arité 2, les méthodes classiques de NMF pour le co clustering (cf. 4.2.3) permettent par défaut de gérer les types d'associations bipartis en considérant une matrice d'affinités non symétrique rectangle, ou de façon équivalente les types d'associations unipartis dirigés en considérant deux modes composés des mêmes individus et ainsi une matrice d'affinités non symétrique carrée. De plus, il est possible d'utiliser des variantes de ces méthodes pour les types d'association unipartis non dirigés grâce aux méthodes NMF symétriques (cf. 4.2.6) essayant de trouver une factorisation symétrique de la matrice d'affinités carrée originale.

Au-delà des propriétés susmentionnées, les méthodes de type NMF permettent également de gérer nativement des associations valuées, tant que les valeurs sont au moins ordinales, mais peuvent également fonctionner sur des matrices d'affinité binaires pour des types d'association non valués. En considérant les transformations en plusieurs types d'association des cas de types d'association valuées par des valeurs non ordinales, et de types d'associations multi attributs, énoncées ci-dessus, les méthodes NMF peuvent être utilisées en théorie pour toutes les situations que l'on peut rencontrer en terme d'attributs descriptifs.

En outre, certaines extensions de NMF permettent de considérer des régularisations par des matrices d'affinités entre individus de même type (cf. 4.2.5), nous donnant une opportunité pour utiliser à la fois l'information topologique d'associations et l'information des attributs entre individus d'un même mode.

Pour toutes les raisons énoncées dans cette section, nous considérons l'approche NMF pour l'apprentissage des MRP-IR+ pour l'expérimentation à suivre et détaillons maintenant le lien entre NMF et apprentissage de fonctions de partition pour les MRP-IR+, en fonction des variantes utilisées.

Considérons un type d'association binaire $\mathcal{R}_a^i$ impliquant deux types d'entité $\mathcal{R}_e^a$ et $\mathcal{R}_e^b$, ainsi qu'une instance $\mathcal{I}_{app}$ du schéma de base de données correspondant. La restriction $\mathcal{I}_{app}(\mathcal{R}_a^i)$ peut être représentée par une matrice $M_{\mathcal{I}_{app}}(\mathcal{R}_a^i) = M$ de taille $n_a \times n_b$ où n_j est le nombre d'individus de $\mathcal{R}_e^j$ pour l'instance considérée (d'apprentissage). Une méthode NMF permet d'obtenir G_a et G_b tels que $\Delta(M, G_a.G_b^t)$ (ou $\Delta(M, G_a.S.G_b^t)$ pour une tri factorisation) soit faible. En forçant G_a et G_b à être orthogonales [51], nous pouvons interpréter ces facteurs comme des matrices d'assignation

des individus de $\mathcal{R}_e^a$ et $\mathcal{R}_e^b$ vers l'ensemble des clusters correspondant. Nous pouvons alors assigner chaque individu au cluster avec lequel il a le poids le plus fort dans le facteur le concernant. Une fois les clusters de chaque individu déterminés, l'apprentissage de paramètres peut être réalisé en utilisant les valeurs de cluster de chaque individu pour les comptages nécessaires à l'estimation des distributions des sélecteurs et de celles de leurs enfants.

Afin de compléter le lien entre NMF et apprentissage de fonctions de partitions, il reste maintenant à définir une méthode permettant d'inférer le cluster d'un individu absent du jeu de données d'apprentissage.

Il est important de noter qu'historiquement, dans les MRP-IR, les attributs de fonction de partition déterminent complètement la valeur du cluster pour tout individu, et il n'est pas possible d'inférer la valeur du cluster de façon probabiliste, car aucune variable aléatoire ne représente l'incertitude de cluster pour une entité. Par conséquent, les valeurs des attributs de fonction de partition doivent être connues à l'instanciation et sont utilisées pour instancier les distributions de probabilités des variables de référence R_{ij}.

Notons maintenant $\mathcal{I}_{test}$ une instance de test. Les individus de ces instances ne sont pas toujours connus par la matrice d'apprentissage M et les facteurs ne contiennent donc pas d'information sur les probabilités d'appartenance de chacun de ces individus aux clusters associés. Considérons maintenant la matrice de relations augmentée $M_{app+test}(\mathcal{R}_a^i) = M+$ contenant l'information sur les relations entre tous les individus de $\mathcal{I}_{app}(\mathcal{R}_e^a)$ et $\mathcal{I}_{test}(\mathcal{R}_e^a)$ d'une part et tous les individus de $\mathcal{I}_{app}(\mathcal{R}_e^b)$ et $\mathcal{I}_{test}(\mathcal{R}_e^b)$ d'autre part. Plusieurs approches peuvent être considérées pour découvrir les matrices facteurs étendues à tous les individus G_a+ et G_b+ :

- réapprendre complètement $G_a + .G_b+^t$ (ou $G_a + .S.G_b+^t$) : cette solution coûteuse n'est pas de l'instanciation, mais un nouvel apprentissage, ce qui peut entraîner une remise en cause lourde du MRP-IR+ associé (au moins un nouvel apprentissage de paramètres semble indispensable a posteriori). De plus, rien n'indique que les clusters auront la même sémantique pour M et $M+$, la méthode de factorisation étant sensible aux permutations. Cela impliquerait de comparer les différents ensembles de clusters, ce qui est également coûteux ;

- dans le cas tri factorisé, réapprendre $G_a+.S.G_b+^t$ en fixant S : solution coûteuse qui n'est pas réellement une instanciation de modèle appris. De plus, rien ne prouve que S sera toujours le bon biais pour G_a+ et G_b+ car S est juste une matrice indiquant une dépendance entre les ensembles de clusters C_a et C_b ;

- trouver G_a+ ou G_b+ en effectuant les mêmes calculs que lors de l'apprentissage, une seule fois, et en fixant G_b^t ou G_a (ou $S.G_b^t$ ou $G_a.S$ pour le cas tri factorisé). Ceci ne fonctionne que si seul l'un des deux ensembles d'individus ne change ;

- intégrer un ensemble d'individus de test à la fois : d'abord, trouver G_a+ tel que $\Delta(M_a+, G_a + .G_b^t)$ (ou $\Delta(M_a+, G_a + .S.G_b^t)$) est minimisé et où S et G_b^t sont fixés et M_a+ est la restriction de $M+$ aux lignes des individus de $\mathcal{I}_{app}(\mathcal{R}_e^a)$; ensuite, trouver G_b+ tel que $\Delta(M+, G_a+.S.G_b+^t)$ (ou $\Delta(M+, G_a+.S.G_b+^t)$) est minimisé et où G_a+ et S sont fixés.

La dernière approche semble la moins contraignante et la moins coûteuse, et nous la choisissons donc pour la suite. Notons toutefois que cette approche n'utilise pas l'intégralité de la matrice $M+$ pour en déduire les facteurs G_i+, mais il semble difficile d'aller outre cette limite (comment en effet lier deux jeux de données relationnels sans chevauchement ?).

5.5 Expériences, partie 1

Dans cette section, nous testons notre approche d'apprentissage des fonctions de partition par des méthodes relationnelles, et comparons les performances obtenues avec celles des fonctions de partition orientées attributs des MRP-IR historiques.

5.5.1 Génération des jeux de données

Dans nos expériences, deux processus de génération de jeux de données ont été créés : l'un a pour but de privilégier les résultats de l'apprentissage des MRP-IR+ par clustering relationnel, tandis que l'autre a pour but de favoriser l'approche orientée attributs.

Le schéma relationnel des jeux de données générés consiste en trois schémas de relation : deux types d'entités appelés E_1 et E_2, et un type d'association appelé A. A contient uniquement une référence vers E_1 et une autre vers E_2. Les types d'entités contiennent le même nombre paramétré d'attributs descriptifs binaires.

Nous générons le même nombre d'entités de type E_1 et E_2 dont les valeurs d'attributs sont obtenues aléatoirement selon des distributions $\phi_{Ei,j}$ pour chaque attribut $\mathcal{A}_j \in E_i$. Nous assignons ensuite les entités de E_1 et E_2 aux ensembles de clusters correspondants C_1 et C_2, de façon différente selon la méthode de génération de données : la méthode favorable au clustering relationnel assigne aléatoirement une valeur de cluster à chaque entité, selon des distributions $P(S_1)$ et $P(S_2)$ données en paramètres ; la méthode favorable au clustering par attributs assigne des valeurs de clusters de façon déterministe par rapport aux valeurs d'attributs des entités, respectant la contrainte des MRP-IR. Après cette étape, les associations de A sont générées en choisissant aléatoirement un groupe c_1 dans C_1, selon la distribution $P(S_1)$, puis en choisissant un groupe c_2 dans C_2 utilisant une distribution $P(S_2|S_1)$ générée aléatoirement, avant de finalement tirer uniformément une entité de E_1 appartenant au groupe c_1 et une entité de E_2 appartenant au groupe c_2.

Pour simplifier la comparaison entre les deux approches, le nombre de clusters est toujours égal à 2^m, où m est le nombre d'attributs de chaque type d'entité.

5.5.2 Protocole d'apprentissage

Dans une expérience, nous générons tout d'abord un jeu de données favorable ou défavorable à l'approche relationnelle, que nous divisons ensuite en 10 parties de façon à réaliser un apprentissage en validation croisée à 10 itérations.

Durant une expérience, trois modèles sont appris : un MRP-IR+ utilisant une méthode de co-clustering relationnel, un MRP-IR utilisant une approche par produit cartésien des attributs de fonction de partition, et un MRP-IR+ optimiste pour lequel nous

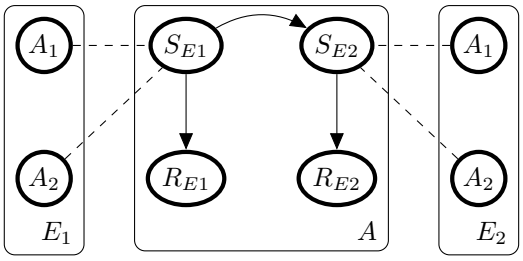


FIGURE 5.1 – Structure fixée pour l'apprentissage de paramètres de MRP-IR+ dans le cas où le nombre d'attributs pour chaque entité est égal à 2. Les liens en pointillés indiquent les attributs de fonctions de partition définis pour les MRP-IR avec fonctions de partition orientées attributs.

connaissons l'assignation réelle des entités aux clusters. La structure, dont nous apprenons les paramètres et les fonctions de partition, est la même pour chaque modèle et est illustrée dans la Figure 5.1.

Le choix de la méthode de clustering relationnel se porte ici sur l'implémentation disponible de NMF minimisant la KL-divergence de [151] pour l'interprétabilité aisée des facteurs obtenus, comme distributions de probabilité des individus d'un type d'entité sur des prototypes moyens de clusters de l'autre type.

Une expérience diffère d'une autre par différents paramètres : le nombre d'attributs binaires m de chaque type d'entités (induisant le nombre de clusters), et le nombre d'entités de chaque type. Les distributions de probabilité sont générées aléatoirement avant échantillonnage du jeu de données.

5.5.3 Méthode d'évaluation

Pour chaque itération de validation croisée et pour chacun des trois modèles appris, nous conservons les log-vraisemblances obtenues sur les 10 itérations de validation croisée sur les jeux de données de test successifs. Nous évaluons chaque expérience en calculant la moyenne et l'écart-type de ces vraisemblances et calculons la significativité statistique de l'hypothèse affirmant que les modèles MRP-IR+ appris par clustering relationnel obtiennent de meilleurs résultats que les MRP-IR appris par clustering par produit cartésien des attributs. Nous effectuons pour cela un z-test [172]. Nous calculons ainsi pour chaque expérience un z-score z défini comme suit :

$$z = \frac{\mu_A - \mu_R}{\sigma_R/\sqrt{10}}, \tag{5.1}$$

où μ_R, μ_A et σ_R représentent respectivement la moyenne des résultats de la validation croisée pour les MRP-IR+, la moyenne des résultats pour les MRP-IR, et l'écart-type des résultats obtenus pour les MRP-IR+. Nous calculons ensuite la p-valeur asymétrique, à partir de la loi normale standardisée cumulative :

$$P(Z \leq z) = \int_{-\infty}^{z} \frac{1}{2\pi} e^{\frac{-u^2}{2}} du \tag{5.2}$$

Cette mesure asymétrique indique la probabilité que les résultats relationnels soient significativement plus faibles que ceux obtenus par produit cartésien. Plus la p-valeur est faible, plus les MRP-IR+ sont significativement meilleurs.

5.5.4 Résultats

Nous lançons plusieurs séries d'expériences, dans le but d'étudier l'impact des différents paramètres sur la log-vraisemblance obtenue. Dans la première série, nous faisons varier le nombre d'individus de 25 à 200, conduisant à un nombre d'associations potentiel compris entre 625 et 40 000. Nous fixons dans cette première série d'expériences la densité d'associations réelle à 0.5, c.-à-d. 50% des associations sont effectivement générées. Nous faisons également varier le nombre d'attributs de chaque type d'entité de 1 à 4, conduisant à un nombre de clusters compris entre 2 et 16. La moyenne et l'écart-type des log-vraisemblances obtenus par validation croisée sont donnés dans les Tableaux 5.1 et 5.2 pour chacun des trois types de modèles appris, de même que la significativité statistique correspondant aux p-valeurs définies précédemment.

k	méthode	nEi 25		50		100		200	
		μ	σ	μ	σ	μ	σ	μ	σ
2	IR+	-637	0	-3 275	7.2	-14 306	25.5	-70 394	78.7
	IR	-665	0	-3 330	0	-15 219	0	-71 831	0
	Optimal	-638	0	-3 251	0	-14 417	0	-70 583	0
	p	**<0.001**		**<0.001**		**<0.001**		**<0.001**	
4	IR+	-1 471	0	-6 720	12.08	-26 088	22.5	-109 909	121.5
	IR	-1 433	0	-6 709	0	-26 301	0	-111 050	0
	Optimal	-1 475	0	-6 752	0	-26 274	0	-110 531	0
	p	>0.999		0.998		**<0.001**		**<0.001**	
16	IR+	-2 025	20.9	-7 750	37.78	-37 805	77.3	-136 246	224.7
	IR	-1 972	0	-7 613	0	-37 546	0	-135 974	0
	Optimal	-2 031	0	-7 763	0	-37 669	0	-135 734	0
	p	>0.999		>0.999		>0.999		>0.999	

TABLE 5.1 – Résultats d'expériences pour les jeux de données favorables aux MRP-IR+. μ représente la moyenne des log-vraisemblances obtenues alors que σ représente l'écart-type correspondant et p indique les p-valeurs des z-tests. Une p-valeur est mise en gras si elle correspond à un test réussi.

Selon la Table 5.1, nous pouvons voir que notre modèle relationnel obtient de meilleurs résultats que les MRP-IR avec produit cartésien dans le cas favorable où les clusters ne sont pas expliqués par les attributs. Toutefois, cela n'est plus vrai lorsque le nombre de clusters est élevé relativement au nombre d'entités de chaque type. Dans ce cas, il semble que les MRP-IR peuvent apprendre des paramètres suffisamment détaillés correspondant aux données. Ces résultats peuvent néanmoins être expliqués par le protocole d'expérimentation. En effet, plus nous augmentons le nombre de clusters pour un nombre fixe d'entités, plus nous prenons le risque d'avoir certains clusters qui ne soient pas représentés dans un des jeux de test lors de la division en 10 sous-jeux de données pour validation croisée. Par conséquent, les individus d'un jeu de test associés à un cluster inconnu ont plus de chances de conduire à des vraisemblances faibles, ce qui peut expliquer la différence.

Il est intéressant d'observer la présence de plusieurs valeurs nulles pour les écarts-types des log-vraisemblances obtenues lors des expériences. Ceci semble intuitif pour les modèles optimaux puisque, ces modèles ne réalisant aucun apprentissage de fonction de partition, les modèles pour chaque itération font alors toujours appel aux mêmes

k	méthode	nEi 25		50		100		200	
		μ	σ	μ	σ	μ	σ	μ	σ
2	IR+	-1 138	0	-6 361	0	-33 820	0	-151 058	0
	IR	-1 169	0	-6 377	0	-33 658	0	-150 746	0
	Optimal	-1 149	0	-6 404	0	-33 853	0	-151 323	0
	p	**<0.001**		**<0.001**		>0.999		>0.999	
4	IR+	-1 897	6.1	-7 917	18.2	-35 272	45.8	-151 699	98.4
	IR	-1 888	0	-7 887	0	-35 538	0	-152 073	0
	Optimal	-1 894	0	-7 994	0	-35 538	0	-152 176	0
	p	>0.999		>0.999		**<0.001**		**<0.001**	
16	IR+	-1 302	2.8	-6 754	108.1	-40 568	60.7	-104 183	59.8
	IR	-1 275	0	-6 631	0	-40 664	0	-104 957	0
	Optimal	-1 294	0	-6 947	0	-40 859	0	-104 835	0
	p	>0.999		>0.999		**<0.001**		**<0.001**	

TABLE 5.2 – Résultats d'expériences pour les jeux de données favorables aux MRP-IR avec clustering sur produit cartésien. μ représente la moyenne des log-vraisemblances obtenues alors que σ représente l'écart-type correspondant et p indique les p-valeurs des z-tests. Une p-valeur est mise en gras si elle correspond à un test réussi.

paramètres pour les mêmes individus. En ce qui concerne l'apprentissage des MRP, la raison est proche : l'ensemble des attributs de fonction de partition fourni étant toujours l'ensemble total des attributs disponibles, et l'appartenance à un groupe pour un individu étant complètement déterminée par cet ensemble, les probabilités de chaque association sont toujours identiques pour une paire d'individus donnés. La situation est en revanche différente pour le cas du clustering relationnel, dont les résultats changent en fonction des individus du jeu d'apprentissage et de leurs associations précises. De plus, la sensibilité de NMF à son initialisation peut également conduire à des résultats différents à partir des mêmes données, renforçant un peu plus la variabilité obtenue dans les résultats. Ces remarques sont également valables pour les résultats d'expériences suivants.

Nous pouvons également voir, selon la Table 5.2 que les MRP-IR+ ne sont pas nécessairement meilleurs que les MRP-IR dans le cas défavorable où les clusters sont totalement expliqués par les valeurs des attributs. Toutefois, ils ne sont pas significativement pires non plus. Ce résultat est assez intuitif puisque si les entités ayant les mêmes valeurs d'attributs se retrouvent dans les mêmes clusters, ces derniers déterminant leur comportement associatif, alors la méthode relationnelle peut retrouver les clusters originaux de façon inverse à partir des associations. Notre modèle apparaît ainsi plus généralisable que les MRP-IR avec produit cartésien sur les attributs. Nous pouvons toutefois noter une exception dans le cas des données non favorables pour quelques cas où les méthodes relationnelles sont meilleures que les résultats des MRP-IR. Toutefois, puisque les p-valeurs semblent distribuées aléatoirement dans la table, rien ne semble l'expliquer mieux que le hasard.

Dans la seconde série d'expériences, nous faisons varier la densité d'associations entre 0.1 et 0.9 par pas de 0.1, pour un nombre d'entités de chaque type constant fixé à 100 et un nombre d'attributs fixé à 2, amenant un nombre de clusters fixé à 4. La

ratio densité	méthode Rel		Att		Opt		
	μ	σ	μ	σ	μ	σ	p
0.1	-6 397	41.23	-6 601	0	-6 571	0	**<0.001**
0.2	-12 222	34.57	-12 508	0	-12 470	0	**<0.001**
0.3	-18 934	58.10	-19 081	0	-19 104	0	**<0.001**
0.4	-22 377	19.72	-22 864	0	-22 619	0	**<0.001**
0.5	-26 642	22.52	-26 951	0	-26 820	0	**<0.001**
0.6	-29 537	42.05	-29 647	0	-29 657	0	**<0.001**
0.7	-32 697	17.09	-32 906	0	-32 773	0	**<0.001**
0.8	-35 217	39.67	-35 580	0	-35 384	0	**<0.001**
0.9	-37 320	73.36	-37 294	0	-37 402	0	>0.999

TABLE 5.3 – Résultats d'expériences pour les jeux de données favorables aux MRP-IR+. μ représente la moyenne des log-vraisemblances obtenues alors que σ représente l'écart-type correspondant et p indique les p-valeurs des z-tests. Une p-valeur est mise en gras si elle correspond à un test réussi.

moyenne et l'écart-type des log-vraisemblances obtenus par validation croisée sont donnés dans les Tables 5.3 et 5.4 pour chacun des trois types de modèles appris, de même que la significativité statistique correspondant aux p-valeurs définies précédemment.

Les résultats de la Table 5.3 montrent que dans un cas favorable au clustering relationnel, les paramètres appris sont généralement plus vraisemblables que pour le cas d'un clustering par attributs, et ce pour la plupart des densités d'associations.

Les résultats de la Table 5.4 pour les données favorables au clustering par attributs montrent également des résultats en faveur des MRP-IR+.

5.5.5 Discussion

Les premières expérimentations faites pour comparer MRP et MRP-IR+ semblent vérifier les propositions faites au sujet du clustering relationnel. Les conclusions de ces travaux sont doubles :

- les méthodes de clustering relationnel permettent d'obtenir de meilleurs résultats d'apprentissage de paramètres que pour les MRP-IR avec un clustering par attributs dans le cas favorables au premier, comme attendu ;

- le clustering relationnel permet d'obtenir des résultats au moins semblables au clustering par attributs, même dans le cas favorable au dernier.

Il est important de noter que ces premières expérimentations sont limitées et présentent quelques défauts. En effet, les distributions de probabilité des différentes variables sont générées de façon aléatoire, amenant des cas où une distribution peut être involontairement proche de la loi uniforme et ainsi non discriminante, ce qui peut conduire à des résultats également aléatoires sur l'apprentissage et enfin l'évaluation des modèles. Il serait intéressant d'approfondir les expériences dans un environnement davantage contrôlé afin de tirer des conclusions péremptoires sur la comparaison entre clustering relationnel et clustering par attributs pour l'apprentissage de MRP.

ratio densité	méthode Rel		Att		Opt		
	μ	σ	μ	σ	μ	σ	p
0.1	-28 979	41.23	-29 358	0	-29 370	0	**<0.001**
0.2	-29 341	34.57	-29 735	0	-29 724	0	**<0.001**
0.3	-30 404	24.21	-30 437	0	-30 746	0	**0.0039**
0.4	-30 572	67.35	-30 676	0	-30 943	0	**<0.001**
0.5	-31 183	3.61	-31 390	0	-31 503	0	**<0.001**
0.6	-31 229	7.28	-31 412	0	-31 518	0	**<0.001**
0.7	-30 734	23.02	-31 412	0	-31 041	0	**<0.001**
0.8	-31 158	15.56	-31 465	0	-31 526	0	**<0.001**
0.9	-29 935	24.026	-30 212	0	-30 287	0	**<0.001**

TABLE 5.4 – Résultats d'expériences pour les jeux de données favorables aux MRP-IR avec clustering sur produit cartésien. μ représente la moyenne des log-vraisemblances obtenues alors que σ représente l'écart-type correspondant et p indique les p-valeurs des z-tests. Une p-valeur est mise en gras si elle correspond à un test réussi.

Au-delà des performances obtenues pour les log-vraisemblances des différents modèles sur les différentes paramétrisations, nous avons pu observer des différences très importantes en terme de temps d'exécution entre les deux modèles comparés. En effet, même si NMF semble en théorie permettre de gérer de nombreux cas, la version utilisée ici met beaucoup de temps pour converger, pouvant atteindre 30 minutes par itération de validation croisée pour un jeu de données de petite taille impliquant juste 200 entités de chaque type. De plus, nous avons observé qu'il était possible pour l'algorithme de ne pas converger, notamment si une matrice facteur devenait singulière, c.-à-d. non inversible. Ces constats posent des problèmes significatifs pour des objectifs d'intégration de ces méthodes à un algorithme d'apprentissage de structure de modèle à terme, où plusieurs exécutions peuvent être requises en un temps possiblement rapide et n'entraînant pas l'arrêt du programme.

5.5.6 Vers un NMF large échelle

Suite aux problèmes de performances énoncées plus haut, il est intéressant de considérer des méthodes de factorisation non négatives avec de meilleures capacités de passage à l'échelle. Dans ce contexte, un article de Wang [183] propose une approche large échelle de NMTF dans laquelle les règles de convergence sont simplifiées. L'idée principale de ses travaux est de factoriser $M_{\mathcal{I}}(\mathcal{R}^i_a) = M$ en 3 matrices G_1, S et G_2 de telle sorte que :

- $\Delta(M, G_1.S.G_2^t)$ soit le plus faible possible ;
- G_1 et G_2 soient des *matrices indicatrices de clusters* contenant un seul 1 par ligne, le reste des valeurs étant fixé à 0 ;
- S contient les affinités entre les ensembles de clusters.

Les avantages de ces matrices forçant le clustering dur sont :

- d'éviter une étape de post-traitements pour affecter les clusters aux individus ;

– de permettre de diviser le problème de clustering en une somme de problèmes plus simples, puisque la nature des matrices G_1 et G_2 permet de calculer indépendamment les valeurs des lignes de G_1 et G_2 à chaque itération. Cela évite une partie des calculs matriciels complexes et coûteux.

Cette méthode semble adaptée dans le cadre des MRP-IR+ et il est intéressant de considérer cette méthode pour pallier les problèmes d'échelle évoqués précédemment. Cette méthode est d'autant plus compatible avec les MRP-IR+ qu'une extension est proposée pour inclure une régularisation laplacienne et ainsi autoriser l'utilisation d'attributs de fonction de partition en plus de l'information d'associations.

Une seconde approche large échelle est évoquée par Cichocki dans [35], en utilisant à chaque itération un sous-ensemble de la matrice originale au lieu de la matrice complète. L'algorithme pour NMF (2 facteurs) consiste à optimiser alternativement les deux fonctions suivantes :

$$G_1[r,.] = \arg\min G_1[r,.] | \Delta(M[r,.], G_1[r,.].G_2^t)$$

$$G_2[.,c] = \arg\min G_2[.,c] | \Delta(M[.,c], G_1.G_2^t[.,c])$$

où $G_1[r,.]$ (resp. $G_2[.,c]$) décrit une restriction de G_1 (resp. G_2) à un ensemble r (resp. c) de ses lignes (resp. colonnes). Le choix des restrictions se fait à chaque itération et on prend généralement les mêmes lignes ou colonnes pour M.

Cette approche est justifiée par le constat que lorsque l'on souhaite approximer une matrice M de dimensions $n_1 \times n_2$, par des facteurs de dimension $n_1 \times c_1$ et $n_2 \times c_2$, avec $c_i << n_i$, alors M contient beaucoup plus d'information que ce que la reconstruction à partir des facteurs peut exprimer. Aussi, il y a une forme de redondance dans M au regard de l'approximation recherchée. C'est pourquoi l'on pourrait en théorie apprendre les mêmes facteurs à partir d'une matrice originale moins grande.

5.6 Expériences, partie 2

Nous continuons les expériences en proposant un cadre plus contrôlé pour la génération des données, palliant le problème de protocole énoncé précédemment. Nous en profitons également pour étudier le comportement de méthodes de clustering plus rapides, cherchant à pallier ainsi le problème de temps d'exécution amené par les méthodes de factorisation.

5.6.1 Protocole de génération des données

Dans ce protocole expérimental, les données sont générées à partir d'une matrice plus ou moins bruitée et définie par blocs. Soit une matrice M de taille $n_1 \times n_2$, avec k clusters pour chaque type d'entité E_1 et E_2. Soit C_1, la variable aléatoire "cluster auquel appartient l'entité de type 1" et C_2, la variable aléatoire "cluster auquel appartient l'entité de type 2". La matrice peut alors être divisée en k blocs dont on souhaiterait maîtriser la densité. Pour chacun des blocs, il est possible de définir une densité d_{ab} déterminant la probabilité que $M(ij)$ soit égal à 1 pour tout (i,j) tel que $C_1(i) = a$ et $C_2(j) = b$. Soit une matrice de densités D de dimensions k, l'algorithme 3 est utilisé pour générer les données.

Algorithme 3 Algorithme de génération de données via l'approche matricielle par blocs

Paramètres : D : une matrice de probabilités de taille k
Retourne : M : une matrice de dimensions $n_1 \times n_2$
Pour $(i, j) \in I(E1) \times I(E2)$ **Faire**
$a = C_1(i)$
$b = C_2(j)$
$M(ij) = 1$ avec une probabilité de $D_{a,b}$, 0 sinon
Fin Pour

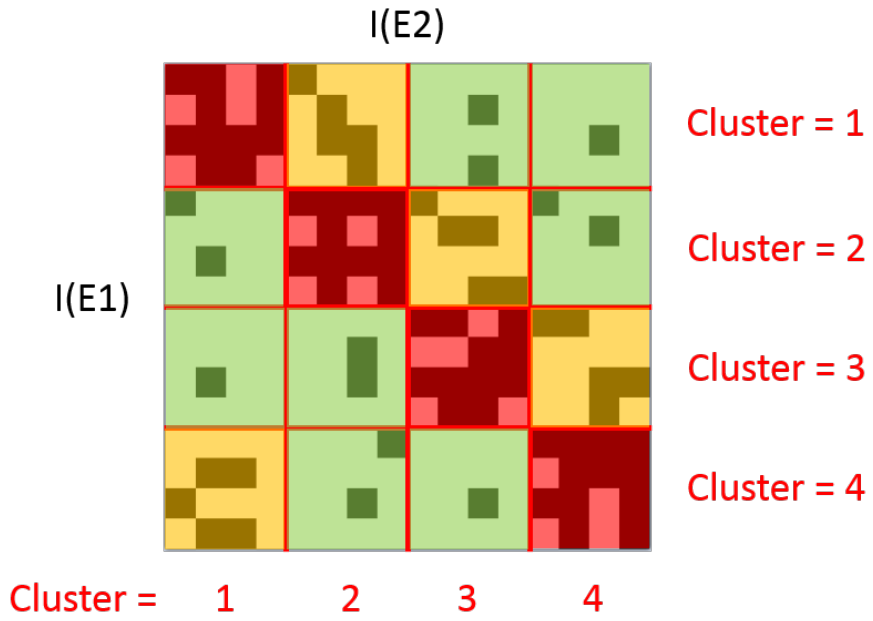


FIGURE 5.2 – Exemple de matrice générée par le protocole de génération de données par blocs. La zone de couleur rouge (resp. orange, verte) correspond à une densité d'associations entre les paires d'individus de 0.7 (resp. 0.3, 0.1). Les associations effectivement générées sont représentées par des zones sombres.

Les densités sont choisies de façon à obtenir des co-clusters qui se démarquent dans la matrice, tout en permettant de générer du bruit pour les blocs ne correspondant pas à des co-clusters. En pratique, trois densités différentes sont définies pour chaque expérience, $d_{max} > d_{moy} > d_{min}$, définissant respectivement une densité forte entre les individus d'un même co-cluster, une densité moyenne pour chaque cluster d'un type d'entité avec un cluster d'un autre type d'entité autre que son co-cluster (bruit significatif localisé), une densité faible pour les autres blocs (bruit faible ambiant). Nous considérons deux contextes de jeux de données à l'égard de ces paramètres : un cas dense, avec des valeurs de densités de 0.7, 0.3 et 0.1 (illustré par la figure 5.2), et un cas creux (sparse) avec des valeurs de densité de 0.1, 0.03 et 0.01. Nous construisons des jeux de données de façon à favoriser à chaque fois une méthode de clustering relationnelle.

5.6.2 Algorithmes considérés

Dans cette série d'expériences, nous considérons plusieurs types d'algorithmes de clustering pour les MRP-IR et MRP-IR+ : 1) MRP avec clustering par produit cartésien (CP) ; 2) Orthogonal Non-Negative Matrix Tri-Factorization [51] (ONMTF) ; 3) Fast Non-Negative Matrix Tri-Factorization [183] (FNMTF) ; 4) Louvain [16] sur un graphe biparti correspondant à la matrice d'associations seule (Louvain-R) ; 5) Louvain sur un graphe construit à partir des similarités entre individus calculées à partir de leurs descriptions par attributs (Louvain-A) ; 6) Louvain sur un graphe biparti augmenté de liens entre individus d'un même mode de la même façon que Louvain-A (Louvain-AR).

L'objectif est encore une fois d'apprendre les paramètres d'une structure fixée maximale d'un MRP-IR, identique que précédemment. Les modèles appris sont évalués par validation croisée à 5 itérations. Les différentes évaluations réalisées sont :

- le nombre de clusters trouvés par la méthode (Louvain ne prend en effet pas cette valeur en paramètre) ;
- la pureté du clustering pour chaque sélecteur ;
- l'information mutuelle normalisée (NMI) du clustering pour chaque sélecteur ;
- la log-vraisemblance obtenue pour les associations du jeu de test.

5.6.3 Résultats

La Figure 5.3 montre les résultats obtenus dans le cas de matrices denses pour les différents indicateurs proposés en faisant varier le nombre d'entités de chaque type entre 0 et 500 et avec un nombre de clusters réels fixé à 4 pour chaque type. Nous avons enlevé les résultats obtenus par les approches de factorisation matricielle après avoir observé qu'ils obtenaient des résultats moins bons que CP la plupart du temps, afin de ne pas surcharger les diagrammes. Nous pouvons voir que les résultats de log-vraisemblance pour Louvain-R sont meilleurs que pour Louvain-A et CP, méthodes orientées attributs. Cette différence est significative comme le prouve un test de Wilcockson sur les 5 itérations ($z < -1.96$). Nous pouvons également voir que Louvain-A et CP donnent des résultats d'ailleurs très proches. De son côté, Louvain-AR ne semble pas fonctionner du tout, comme l'indique la faible quantité de clusters découverts : 2 au lieu de 4. À titre indicatif, les temps d'exécution obtenus pour 400 entités sont de 3 189s pour Louvain-R, 442s pour Louvain-A, et 21 758s pour Louvain-AR pour l'ensemble des 5 itérations. La topologie d'un type d'association dense semble ainsi pouvoir être apprise avec Louvain-R.

La Figure 5.4 montre les résultats obtenus dans le cas de matrices creuses pour les mêmes indicateurs. Nous pouvons voir que dans ce cas, même si Louvain-R possède de bons résultats pour les évaluations de clustering de pureté, indiquant une bonne densité des clusters découverts, le score de NMI est en revanche très mauvais, indiquant des différences très nettes avec le clustering réel des individus. Le nombre de clusters obtenus pour cette méthode montre également à quel point l'algorithme est loin de la réalité, trouvant une centaine de groupes en moyenne contre les 4 réels. De ce fait, Louvain-R et l'approche hybride Louvain-AR offrent des performances en vraisemblance très mauvaises, bien loin de ce qui est obtenu par les approches par attributs,

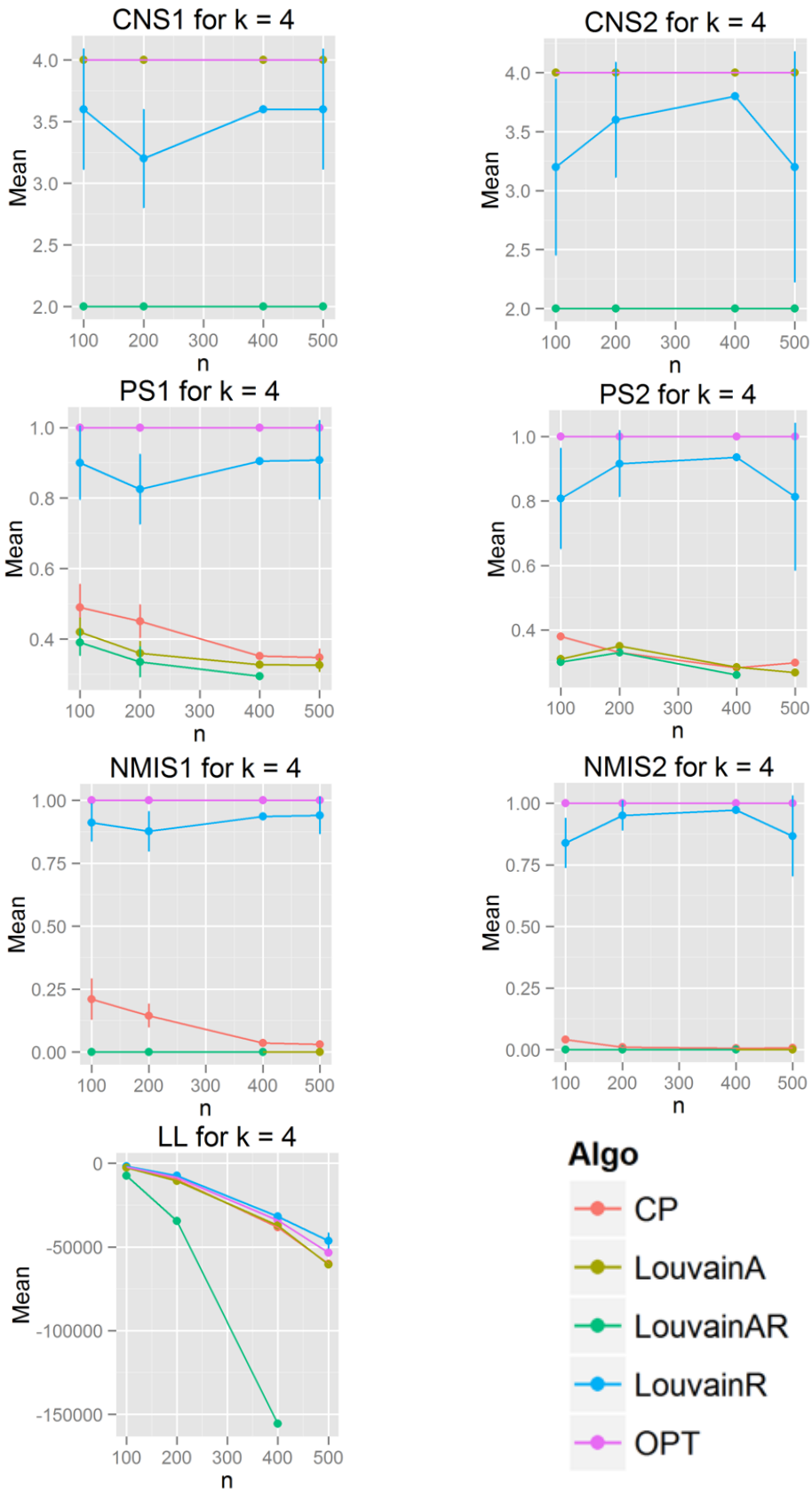


FIGURE 5.3 – Résultats d'expériences pour les données denses. CNS1 et CNS2 représentent le nombre de clusters obtenus pour chaque sélecteur ; PS1 et PS2 représentent leurs scores de pureté ; NMIS1 et NMIS2 représentent leurs scores d'information mutuelle normalisée ; LL est la log-vraisemblance des individus de test du type d'association considéré ; k est le nombre de blocs réels.

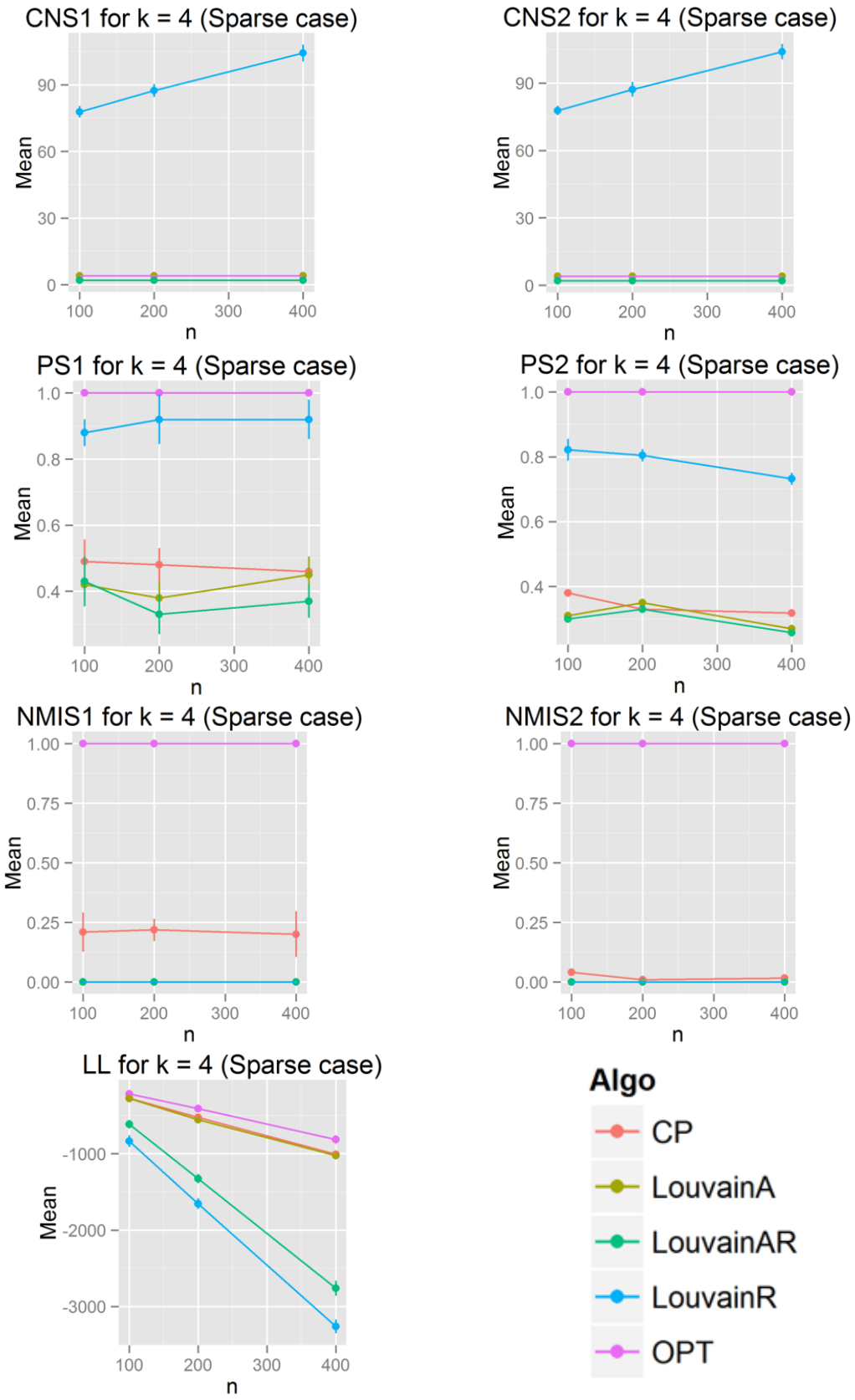


FIGURE 5.4 – Résultats d'expériences dans le cadre de matrices de données creuses. CNS1 et CNS2 représentent le nombre de clusters obtenus pour chaque sélecteur ; PS1 et PS2 représentent leurs scores de pureté ; NMIS1 et NMIS2 représentent leurs scores d'information mutuelle normalisée ; LL est la log-vraisemblance des individus de test du type d'association considéré ; k est le nombre de blocs réels.

pourtant loin d'être elles-mêmes optimales. Ceci montre les limites d'une approche de détection de communautés où le nombre de clusters n'est pas donné à l'avance, et expose également les problèmes de limite de résolution évoqués dans la littérature pour les algorithmes utilisant le critère de modularité [61].

Il semble donc à première vue intéressant d'utiliser une approche de détection de communautés telle que Louvain sur le graphe reliant les différentes entités en fonction de leurs associations effectives pour des données d'associations non creuses et sans mélanger les informations d'associations et d'attributs. Cette méthode semble offrir de meilleurs résultats en un temps moindre que les approches par factorisation non négative de matrices dans ces cas. D'autres approches doivent ainsi être considérées pour réussir à introduire de façon efficace l'information d'attributs, et pour gérer les données creuses. Nous explorons d'autres solutions en ce sens dans le chapitre suivant.

5.7 Conclusion

Les expériences réalisées dans ce chapitre montrent que l'utilisation du co-clustering pour l'apprentissage des MRP-IR+ a des avantages par rapport à la méthode historique. Tout d'abord, l'utilisation d'algorithmes de co-clustering permet justement de trouver un partitionnement des lignes et des colonnes conjointement construit, contrairement à ce qui est fait avec l'approche par produit cartésien. Ces partitionnements sont alors très représentatifs des associations et y sont locaux, ce qui semble conférer aux variables aléatoires dédiées une information réellement représentative du type d'association.

Ensuite, l'apprentissage par une méthode de co-clustering permet d'utiliser l'information des associations pour le partitionnement, ce qui est un plus par rapport à la méthode obtenue par le produit cartésien. En effet, le choix des attributs de partition pour un sélecteur biaise l'apprentissage des paramètres qui en est fait, et peut changer complètement l'information qui pourra en être extraite. Certains partitionnements seront alors plus discriminants et d'autres ne permettront d'obtenir que des distributions uniformes. L'intérêt d'utiliser une méthode de co-clustering est alors de trouver les partitionnements qui maximiseront une discrimination entre clusters de chaque entité.

En outre, l'apprentissage par une méthode de co clustering présente l'avantage de fournir des résultats même si l'une ou les deux entités impliquées ne présentent aucun attribut descriptif. Il est évident que ceci est impossible avec le partitionnement par produit cartésien. Il est intéressant de noter que cet argument est loin d'être anodin puisqu'il peut à titre d'exemple permettre l'inclusion de données textuelles dans les MRP-IR, sous la forme d'associations (Document, Mot) où la classe de mots ne présente aucun attribut.

De façon inverse, la possibilité pour la méthode de co-clustering de s'abstraire des attributs descriptifs pourrait permettre de gagner du temps lors de l'apprentissage du graphe du MRP-IR+ si les entités possèdent de nombreux attributs descriptifs, puisque cela évite une combinatoire potentiellement grande pour le choix d'attributs de fonctions de partition.

La prochaine étape pour l'amélioration des modèles est de considérer un apprentissage de structure utilisant des méthodes de clustering relationnel. Toutefois, les MRP-IR+ sont limités pour cet objectif. En effet, comme le déterminisme entre attributs et partitions n'y est plus vérifié, d'autres contraintes des MRP-IR héritées par les MRP-IR+ posent problème. Ainsi, l'instanciation d'un MRP-IR+ nécessite toujours d'instancier les distributions $P(R_{ij}|S_{ij})$, nécessitant de savoir dans quelle partition se trouve un individu, dépendant de la topologie des associations de l'instance d'apprentissage et donc impliquant de savoir où se trouvent les entités dans cette topologie. Si nous souhaitons réellement fonctionner dans un contexte d'incertitude d'associations pour une nouvelle association et utiliser des méthodes de co-clustering, il faut donc ajouter l'appartenance d'une entité à un groupe pour un type d'association comme variable du modèle à apprendre.

Dans le chapitre suivant, nous proposons un nouveau modèle de MRP ayant les contraintes pour les fonctions de partition des MRP-IR+, mais permettant également de considérer l'incertitude de groupes pour les entités du domaine considéré. Nous proposons un algorithme d'apprentissage de structure pour ces nouveaux modèles et comparons les résultats d'apprentissage obtenus sur des jeux de données synthétiques et réels par rapport aux MRP-IR historiques.

6

MRP avec incertitude de Clusters

Sommaire

6.1 Introduction

Les résultats du chapitre 5 relatifs à l'apprentissage de paramètres des MRP-IR, et d'une extension définie appelée MRP-IR+, nous laissent penser que l'utilisation d'algorithmes de clustering relationnel tend à améliorer les résultats d'apprentissage des MRP dans un contexte d'incertitude de références. Néanmoins, l'impact de l'utilisation de clustering relationnel sur l'apprentissage va au-delà du simple apprentissage de paramètres et nous pensons qu'elle peut également améliorer l'apprentissage de structure. De plus, les limites relatives aux fonctions de partition ne sont pas les seules identifiées au chapitre 3 où nous évoquons également une catégorie de limites liées à la structure graphique des MRP-IR, à savoir : l'impossibilité de réaliser de l'inférence de groupe pour un nouvel individu après instanciation d'un modèle ; le cloisonnement important des types d'association empêchant la découverte de dépendances probabilistes entre eux ; une difficulté de convergence de structure à cause de l'exploration purement empirique des ensembles d'attributs de fonction de partition.

Notre objectif dans ce chapitre est de réaliser de l'apprentissage de structure de MRP dans un contexte d'incertitude de références et en utilisant des algorithmes de clustering relationnel, tout en corrigeant les limites des MRP-IR énoncés plus hauts et détaillés dans le chapitre 3. Nous proposons dans ce sens un second modèle, appelé *MRP avec incertitude de clusters* (MRP-IC), une extension plus importante des MRP-IC que les MRP-IR+ du chapitre 5. Il contient tout d'abord les mêmes contraintes relâchées que les MRP-IR+, mais propose également l'ajout de nouvelles variables permettant de modéliser l'appartenance d'une entité à un cluster, dans le but de pouvoir inférer ces valeurs et non plus les déterminer de façon certaine à l'instanciation. Cet ajout permet de pallier la perte de la contrainte de déterminisme des MRP-IR, entraînant la possibilité de réaliser de l'apprentissage de structure de façon plus généralisable. Les MRP-IC sont détaillés en terme d'inférence et d'apprentissage conjoint de leurs fonctions de partition et de leurs structures graphiques, et nous réalisons des expériences où nous comparons les structures apprises par les MRP-IR et les MRP-IC, d'abord sur des données synthétiques, puis sur des données réelles.

Une version préliminaire de ce travail a conduit à la publication d'un article [42], présenté lors de la conférence IJCNN 2015.

6.2 MRP avec incertitude de Clusters

Un modèle relationnel probabiliste avec incertitude de clusters (MRP-IC) est un modèle relationnel probabiliste opérant dans un contexte d'incertitude de référence comme pour les MRP-IR définis au chapitre 3 et utilise des variables aléatoires liées à des fonctions de partition permettant d'effectuer de l'inférence de valeurs de référence pour des associations. Toutefois, ceux-ci introduisent des variables aléatoires supplémentaires permettant de déterminer l'appartenance des entités aux différents groupes au moment de l'inférence, et relâchent les contraintes de fonctions de partitions des MRP-IR pour autoriser l'utilisation de méthodes de clustering relationnel comme pour les MRP-IR+ du chapitre 5.

Définition 6.1. *Étant donné un schéma de base de données $\mathbf{R} = \langle \mathcal{R}^i \rangle_{i \leq m}$, un MRP-IC $\mathcal{M} = (\mathcal{S}, \Phi, \Theta_{\mathcal{S},\Phi})$ est composé d'une structure graphique $\mathcal{S}$, d'une structure de partitions Φ et de paramètres $\Theta_{\mathcal{S},\Phi}$, simplifié par Θ quand aucune confusion n'est possible. La structure graphique est un graphe acyclique dirigé (DAG) composé de variables aléatoires $\mathcal{V}$ et d'une fonction de parents $pa : \mathcal{V} \rightarrow \mathcal{P}(\Psi \times \Sigma(\mathbf{R}) \times \mathcal{V})$, où Ψ est l'espace des fonctions d'agrégation, et $\Sigma(\mathbf{R})$ est l'ensemble infini de toutes les séquences de références possibles dans $\mathbf{R}$. La structure de partitions est un ensemble de fonctions de partitions qui sont associées aux variables aléatoires cluster dans $\mathcal{S}$.*

Pour tout schéma de relation $\mathcal{R}^i \in \mathcal{R}$, nous définissons une variable aléatoire $A_{i,j}$ pour chaque attribut $\mathcal{A}_j \in att(\mathcal{R}^i)$ avec $dom(A_{i,j}) = dom(\mathcal{A}_j)$. De plus, pour chaque type d'association $\mathcal{R}^i_a \in \mathbf{R}_a \subseteq \mathbf{R}$, nous ajoutons une variable de référence $R_{i,n}$ ainsi que deux variables aléatoires de cluster $C^S_{i,n}$ (appelée variable de cluster source) et $C^T_{k,i,n}$ (appelée variable de cluster cible) pour chaque type d'entité référencé $r_n \in fk(\mathcal{R}^i_a)$ où $ref(\mathcal{R}^i_a, r_n) = \mathcal{R}^k$.

Pour chaque paire de variables aléatoires cluster $(C^S_{i,n}, C^T_{k,i,n})$, obtenues pour la référence $r_n \in fk(\mathcal{R}^i_a)$ avec $ref(\mathcal{R}^i_a, r_n) = \mathcal{R}^k$, nous définissons une fonction de partition $\phi_{i,n} : \mathcal{I}(\mathcal{R}^k) \rightarrow \{1, \ldots, c_{i,n}\}$, où $c_{i,n}$ dénote le nombre correspondant de parties, sur les entités de $\mathcal{I}(\mathcal{R}^k)$ pour une instance $\mathcal{I}$. L'identité suivante doit être vérifiée :

$$dom(C^S_{i,n}) = dom(C^T_{k,i,n}) = ran(\phi_{i,n}) = \{1, \ldots, c_{i,n}\}.$$

La fonction de partition peut également avoir un ensemble d'attributs $\mathcal{A}(\phi_{i,n}) \subseteq att(\mathcal{R}^k)$ signifiant qu'elle repose en partie ou complètement sur ces attributs pour partitionner les données.

Par souci de concision, nous notons dans la suite par $V_{i,j}$ une variable qui peut être soit de la forme $A_{i,j}$, $R_{i,j}$, $C^S_{i,j}$ ou $C^T_{i,a,j}$.

Comme pour les MRP-IR, la fonction de parents doit obéir à certaines contraintes pour définir un MRP-IC valide. D'abord, la fonction doit être telle que chaque RBP obtenu à partir de ce modèle est garanti acyclique. De plus, l'image d'une variable $V_{i,j}$ par la fonction pa nécessite que chaque triplet $(\psi, \sigma, V_{a,b}) \in pa(V_{i,j})$ satisfasse les règles suivantes : si σ est une séquence de références non vide, alors ψ est une fonction d'agrégation et nous avons $dom(V_{a,b}) = ran(\sigma)$; dans le cas contraire, ψ doit être la fonction identité id, et nous avons $dom(V_{a,b}) = dom(V_{i,j})$.

Il est important de noter que nous avons toujours :

$$\left[C^S_{i,n} \wedge ref(\mathcal{R}^i, n) = \mathcal{R}^k \right] \rightarrow \left[\exists C^T_{k,i,n} \wedge dom(C^S_{i,n}) = dom(C^T_{k,i,n}) = ran(\phi_{i,n}) \right],$$

c.-à-d. le schéma de relation référencé et le schéma de relation référençant doivent tous deux avoir une variable cluster associée à la même fonction de partition. Ces variables peuvent toutefois avoir des distributions différentes et ne représentent pas la même information. En effet, si nous considérons une association comme un (hyper-)arc étiqueté entre les entités, la variable cluster source définit la probabilité qu'une extrémité d'un arc soit une entité d'un cluster particulier, alors que la variable cluster cible définit la probabilité qu'une entité soit dans un cluster particulier pour le type d'association considéré. Notez que la probabilité de choisir une entité spécifique dans une association pour une valeur de variable de référence R est toujours conditionnée par les deux variables cluster associées. Cela peut être interprété comme la contribution séparée des informations relationnelles et internes d'un individu sur la probabilité

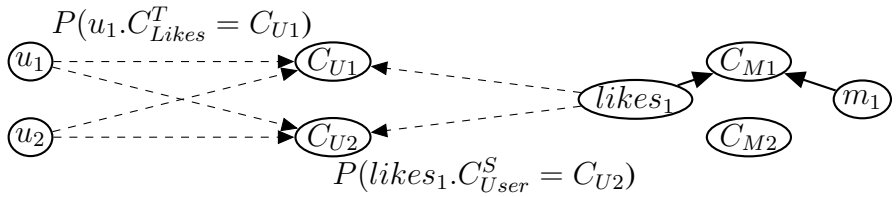


FIGURE 6.1 – Illustration de la différence entre variables cluster source et cible pour un schéma de relation $likes$ déterminant le goût d'un ensemble d'utilisateurs pour un ensemble de films (un utilisateur est ainsi associé à un film s'il l'apprécie). Pour un tuple spécifique de cette relation, nous connaissons ici le film référencé (arc continu), mais pas l'utilisateur correspondant (incertain, arcs en pointillé). La variable cluster source $likes_1.C^S_{User}$ est liée à la distribution du choix d'un cluster d'utilisateur, qui peut être influencée par le film m_1 référencé. La variable cluster cible $u_i.C^T_{Likes}$ est définie pour chaque utilisateur u_i et représente la distribution d'appartenance d'un individu à un cluster, qui peut être influencée par les autres variables de l'individu ou des variables distantes par des séquences de référence non nulles dans le schéma relationnel. La probabilité $P(likes_1.User = u_i)$ est donc dépendante de la contribution de deux variables cluster différentes, une orientée variables relationnelles de $likes_1$, l'autre orientée information de description interne des u_i.

de ce dernier d'être concerné par la référence. La Figure 6.1 illustre la différence de sémantique entre les variables de cluster source et cible.

Finalement, les paramètres $\Theta_{\mathcal{S},\Phi}$ sont composés de distributions de probabilités conditionnelles pour chaque variable $V \in \mathcal{V}$ étant donnés ses parents $pa(V_{i,j})$, c.-à-d. $P(V_{i,j}|pa(V_{i,j}))$.

Un exemple de MRP-IC pour notre domaine UMA est montré dans la Figure 6.2. Pour des raisons de lisibilité, nous remplaçons les noms des variables $V_{i,j}$ par des variables V_j dans le rectangle du schéma de relation i. Le MRP-IC définit une variable aléatoire pour chaque attribut et contrainte d'intégrité référentielle dans le schéma relationnel. De plus, pour chaque référence, deux variables cluster sont ajoutées, une pour le type d'association (la variable source), l'autre pour le type d'entité référencé (la variable cible). Par exemple, la référence *user* dans le type d'association *Rating* mène à la création des variables C^S_{user} dans $Rating$ et C^T_{Rating} dans $User$. Enfin, chaque variable possède un ensemble de parents potentiellement vide et suit une distribution de probabilités conditionnelles sachant ses parents. Il est important de noter que le parent A_{year} dans le type d'entité *Movie* de A_{age} dans *User* implique une séquence de références de longueur 2 et une fonction d'agrégation ψ. Aussi, la CPD pour A_{age} est de la forme : $P(A_{age}|(\psi, \sigma, A_{year}), C^T_{Rating})$, avec $\sigma = \langle\ (Rating, user)^{-1}, (Rating, movie)\ \rangle$.

6.3 MRP-IC complété et graphe de dépendance

Comme pour les MRP-IR, nous définissons un opérateur $comp$ sur les MRP-IC, ajoutant les triplets nécessairement induits dans la fonction pa, dans le but de garantir l'inférence sur un MRP-IC pour n'importe quelle instance. Cette étape est également nécessaire pour vérifier l'acyclicité du modèle, et donc sa validité.

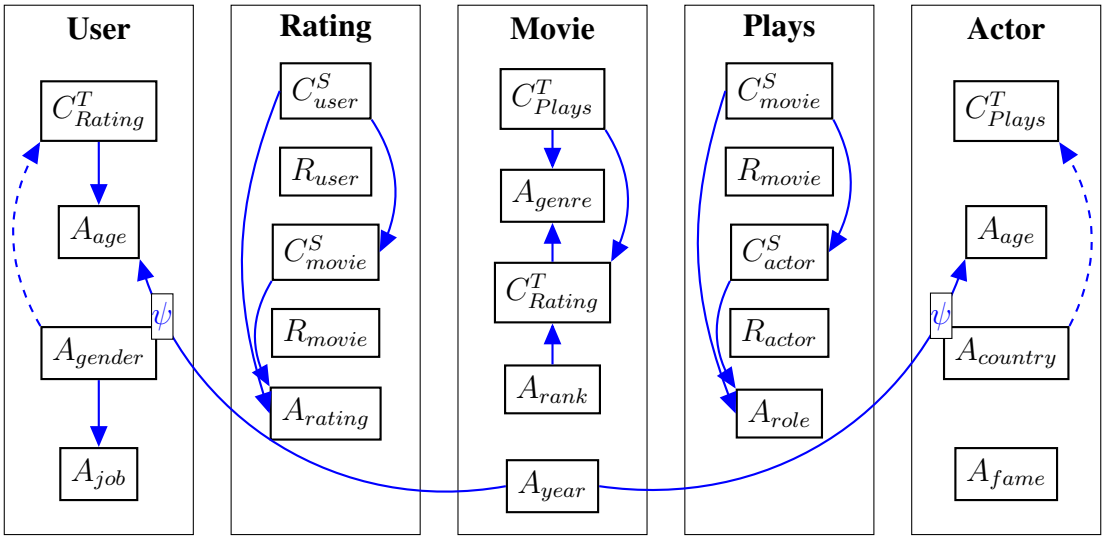


FIGURE 6.2 – Un MRP-IC possible pour le schéma relationnel UMA.

Définition 6.2. *Soit un MRP-IC* $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$. *Nous appelons modèle complété de* $\mathcal{M}$, *le modèle* $comp(\mathcal{M}) = (comp(\mathcal{S}), \Phi, \Theta)$, *avec* $comp(\mathcal{S}) = (\mathcal{V}, comp(pa))$, *tel que :*

- $pa \subseteq comp(pa)$;
- *si* $(\psi, \sigma, V_{a,b}) \in pa(V_{u,v})$ *et* $\sigma = \langle \sigma_i \rangle_{i \leq l}$, *alors nous avons :*

$$\forall i \leq l : (id, s_i, R_{a,n}) \in comp(pa)(V_c) \mid \sigma_i = (\mathcal{R}^a, f_n) \vee \sigma_i = (\mathcal{R}^a, f_n)^{-1}$$

  où id *est l'agrégateur identité (qui n'agrège rien) et où nous avons :*

  - $s_i = (\sigma_1, \ldots, \sigma_i)$ *si* σ_i *est inversée,*
  - $s_i = (\sigma_1, \ldots, \sigma_{i-1})$ *si* σ_i *est directe ;*

- *si* $\phi_{i,f} \in \Phi$, *pour* $r_f \in fk(\mathcal{R}^i)$ *et* $ref(\mathcal{R}^i, r_f) = \mathcal{R}^k$, *alors nous avons pour ses attributs* $A_{k,j} \in \mathcal{A}(\phi_{i,f})$ *:*

$$(id, \emptyset, A_{k,j}) \in comp(pa)(C^T_{k,i,f})$$

  où $\emptyset$ *est la séquence de références vide ;*

- $R_{i,f} \in \mathcal{V} \rightarrow \{(id, \emptyset, C^S_{i,f}), (id, \langle (\mathcal{R}^i, f) \rangle, C^T_{k,i,f})\} \subseteq comp(pa)(R_{i,f})$.

Un MRP-IC $\mathcal{M}$ *est complété si* $comp(\mathcal{M}) = \mathcal{M}$.

Les modèles graphiques dirigés doivent garantir l'acyclicité de leurs instanciations. Les MRP introduisent un processus de vérification au niveau des modèles factorisés dans le but de garantir que toute instanciation d'un tel modèle résulte en un RBP acyclique. Ce processus de vérification consiste à vérifier l'acyclicité d'un graphe de dépendance, obtenu à partir du MRP original. Dans le cas d'un MRP-IC, le graphe de dépendance est trivial une fois que le MRP-IC complété a été obtenu.

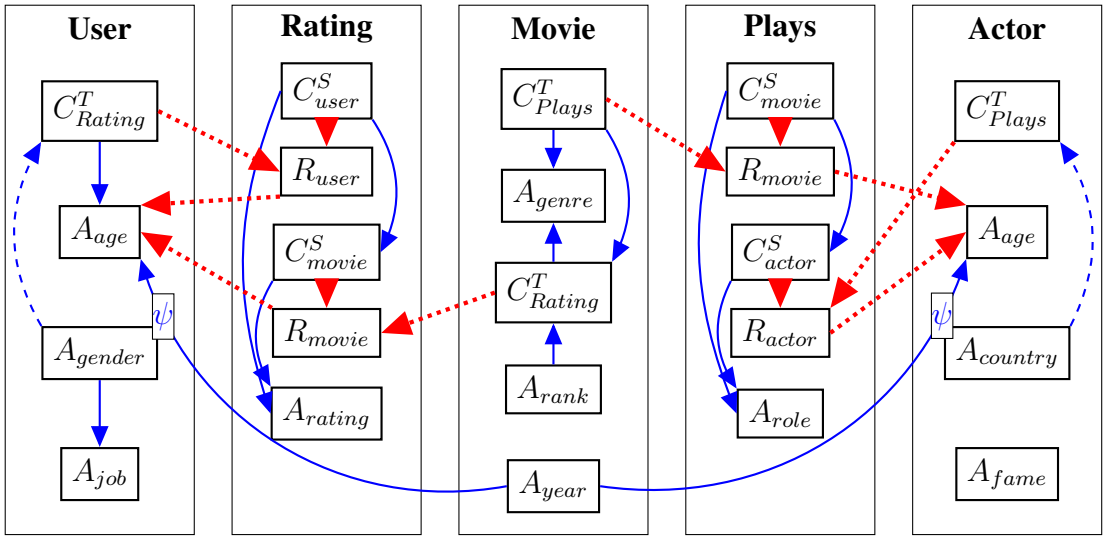


FIGURE 6.3 – MRP-IC complété pour le MRP-IC définit précédemment sur UMA.

Définition 6.3. *Soit pa_V la projection de la fonction pa définie de la façon suivante :*

$$pa_V = \{\ (C, V_p)\ |\ \exists\sigma\exists\psi(\ (\psi, \sigma, V_p) \in pa(C)\ )\ \}\,.$$

Un MRP-IC est dit valide, si la projection pa_V de la fonction de parents du MRP-IC complété décrit un graphe acyclique.

Comme pour les MRP-IR, un MRP-IC valide garantit que chaque instance $\mathcal{I}$ mène à la génération d'un RBP valide.

La forme complétée du MRP-IC précédent est présentée dans la Figure 6.3. Les nouvelles dépendances sont en gras. Nous pouvons voir deux sortes de nouvelles dépendances par rapport au MRP-IC original : la première entre les variables cluster source et cible et la variable de référence correspondante (p.ex. C^S_{user} et C^T_{Rating} vers R_{user}) ; la seconde entre les variables de référence impliquées dans une séquence de référence, et la variable enfant pour laquelle cette séquence a été définie. Par exemple :

$$A_{year} \in pa(A_{age}) \rightarrow \{R_{movie}, R_{user}\} \in comp(pa)(A_{age})).$$

6.4 Inférence

6.4.1 Instanciation d'un MRP-IC

Comme pour tous les modèles relationnels probabilistes, un MRP-IC peut être instancié pour produire un réseau bayésien à plat (RBP), ensemble avec une instance $\mathcal{I}$ du même schéma relationnel. Considérant une instance $\mathcal{I} \in inst(\mathbf{R})$ et un MRP-IC complété $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$ pour le même schéma de base de données, nous pouvons construire un RBP unique $\mathcal{G}(\mathcal{M}, \mathcal{I}) = (V_\mathcal{I}, pa_\mathcal{I}, \Theta_\mathcal{I})$ pour le squelette objet $\pi(\mathcal{I}) = \pi$.

Définition 6.4. *[62] Soit $\mathcal{M} = (\mathcal{S}, \Phi, \Theta)$, un MRP-IC complété valide et une instance $\mathcal{I} \in inst(\mathbf{R})$ compatible, le réseau bayésien à plat $\mathcal{G}(\mathcal{M}, \mathcal{I}) = (V_\mathcal{I}, pa_\mathcal{I}, \Theta_\mathcal{I})$ est défini de la façon suivante :*

- *une variable aléatoire $a_{e,i,j} \in V_{\mathcal{I}}$ existe pour tout $A_{i,j}$ de $\mathcal{M}$, avec $dom(a_{e,i,j}) = dom(A_{i,j})$ pour tout $e \in \pi(\mathcal{R}^i)$;*

- *une variable de référence $r_{e,i,f}$ existe pour chaque $R_{i,f}$ de $\mathcal{M}$, avec $dom(r_{e,i,f}) = \pi(\mathcal{R}^i)$ pour tout $e \in \pi(\mathcal{R}^i)$;*

- *une variable cluster source $C^S_{e,i,f} \in V_{\mathcal{I}}$ existe pour tout $C^S_{i,f}$ de $\mathcal{M}$ et tout $e \in \pi(\mathcal{R}^i)$;*

- *de façon similaire, une variable cluster cible $C^T_{e,k,i,f} \in V_{\mathcal{I}}$ existe pour tout $C^T_{k,i,f}$ de $\mathcal{M}$ et tout $e \in \pi(\mathcal{R}^k)$;*

- *pour tout $V_{i,j}$ et chaque triplet parent $(\psi, \sigma, V_{a,b}) \in pa(V_{i,j})$, nous avons :*

$$\forall s \in \mathcal{I}(\mathcal{R}^i)\, \forall t \in \pi(\mathcal{R}_a)\ :\ (\psi, \sigma, v_{t,a,b}) \in pa_{\mathcal{I}}(v_{s,i,j});$$

- *une distribution de probabilités conditionnelle $P(v_{e,i,j}|pa(v_{e,i,j})) \in \Theta_{\mathcal{I}}$ existe pour tout $v_{e,i,j} \in V_{\mathcal{I}}$ avec $P(v_{e,i,j}|pa(v_{e,i,j})) = P(V_{i,j}|pa(V_{i,j}))$ pour la variable aléatoire correspondante dans $\mathcal{M}$. Toutes les variables instanciant la même variable aléatoire dans le MRP-IC sont dites attachées au même paramètre.*

Un RBP décrit une distribution jointe unique sur les individus de $\mathcal{I}$, utilisant les règles probabilistes définies dans $\mathcal{M}$. Nous pouvons ainsi calculer la vraisemblance d'une instance sachant son squelette objet et le MRP-IC :

$$\begin{aligned} P(\mathcal{I}|\mathcal{M}, \pi) = & \prod_{\mathcal{R}^i \in \mathbf{R}} \prod_{\mathcal{A}_j \in att(\mathcal{R}^i)} \prod_{e \in \mathcal{I}(\mathcal{R}^i)} P(a_{e,i,j}|pa(a_{e,i,j})) \\ & \prod_{\substack{f \in fk(\mathcal{R}^i), \\ ran(f) = \mathcal{R}^k}} P(c^S_{e,i,f}|pa(c^S_{e,i,f})) \cdot P(r_{e,i,f}|pa(r_{e,i,f})) \\ & \prod_{g \in \mathcal{I}(\mathcal{R}^k)} P(c^T_{g,k,f}|pa(c^T_{g,k,f})) \end{aligned} \tag{6.1}$$

6.4.2 Complétion et distributions de probabilité d'un RBP

Tout comme les MRP-IR, la complétion d'un MRP-IC mène à l'ajout de dépendances probabilistes de façon automatique et ceci impacte la façon dont les distributions de probabilités originales sont utilisées pour l'inférence dans un RBP. Nous explicitons ici l'impact des nouvelles dépendances sur les distributions de probabilités originalement définies et sur la façon de calculer une probabilité pour réaliser de l'inférence dans un RBP particulier. Pour des raisons de clarté du discours, nous séparons chaque type de dépendance induite dans les sections suivantes.

i Dépendances impliquant des séquences de références non nulles

Ce type de complétion est le même que pour les MRP-IR pour le calcul d'une distribution de probabilités conditionnelles dans le RBP pour une variable ayant des parents impliquant des séquences de références de taille non nulle, et s'appuie sur la résolution de parents à partir des valeurs des variables de référence impliquées (cf. 3.8.2).

ii Ajout de parents aux variables cluster cible

Considérons maintenant une variable cluster cible $C^T_{i,u,j}$ dont la variable cluster source associée est $C^S_{u,j}$ dans le MRP-IC. Sans prendre en compte les autres types de parents, nous avons une distribution de la forme $P(C^T_{i,u,j})$. Nous avons donc dans le MRP-IC complété une distribution de la forme $P(C^T_{i,u,j}|A_{pf})$ où $A_{pf} = \{(id, \emptyset, A_{i,a})|A_{i,a} \in \mathcal{A}(\phi_{u,j})\}$. Aussi, le RBP pour une instance inclut l'ensemble de distributions de probabilités suivant :

$$\bigcup_{\forall t \in \mathcal{I}(\mathcal{R}^i)} \{\ P(c^T_{t,i,u,j} \mid \{(id, \emptyset, a_{t,i,a})|A_{i,a} \in \mathcal{A}(\phi_{u,j})\})\ \}.$$

Dans le but de calculer une distribution spécifique, nous avons juste à calculer la distribution de probabilité des appartenances aux clusters de $\phi_{u,j}$ pour un t donné, sachant ses valeurs d'attributs $\{a_{t,i,a}|A_{i,a} \in \mathcal{A}(\phi_{u,j})\}$. Ceci est équivalent à l'ajout d'un attribut spécifique $C^T_{i,u,j}$ dans $\mathcal{R}^i$ et à donner une valeur de $ran(\phi_{u,j})$ pour chaque tuple t de $\mathcal{I}(\mathcal{R}^i)$ pour ensuite simplement calculer des comptages de distributions de probabilités conditionnelles dans la relation étendue, et enfin estimer les paramètres de façon régulière à partir des comptages réalisés de cette façon.

Dans les MRP-IR, les clusters sont totalement déterminés par les attributs de fonction de partition et ces valeurs doivent être connues à l'instanciation pour pouvoir assigner chaque tuple à ses groupes. Dans les MRP-IC, nous autorisons plus de flexibilité en permettant la découverte de l'appartenance d'individus aux différents groupes comme étape intégrante de l'inférence. Ceci permet d'utiliser un plus grand nombre de méthodes pour l'apprentissage des fonctions de partition, où le déterminisme énoncé n'est pas vérifié, mais aussi de découvrir des associations d'un type à partir d'informations provenant d'autres types d'associations, ce qui peut être pratique si un individu n'est pas présent pour tous les types d'associations le concernant potentiellement.

iii Ajout de dépendances entre variables clusters et variables références

Considérons enfin une variable de référence $R_{i,j}$ avec $ref(\mathcal{R}^i, j) = \mathcal{R}^u$ dans le MRP-IC. Sans prendre en compte les autres types de parents, nous avons alors une distribution de la forme $P(R_{i,j})$. Nous avons donc dans le MRP-IC complété une distribution de la forme :

$$P(\ R_{i,j} \mid \{(id, \emptyset, C^S_{i,j}), (id, \langle(\mathcal{R}^i, j)\rangle, C^T_{u,i,j})\}\ ).$$

Aussi, le RBP pour une instance donnée doit inclure les distributions de probabilité suivantes :

$$\bigcup_{\forall t \in \mathcal{I}(\mathcal{R}^i)} \{\ P(r_{t,i,j} \mid \{(id, \emptyset, c^S_{t,i,j})\} \cup c^T)\ \}.$$

avec :

$$c^T = \bigcup_{\forall s \in \mathcal{I}(\mathcal{R}_u)} \{(id, \langle(\mathcal{R}^i, j)\rangle, c^T_{s,u,i,j})\}$$

Dans le but de calculer une distribution spécifique d'un tuple $t \in \mathcal{I}(\mathcal{R}^i)$, nous devons considérer chaque tuple s tel que $C^T_{s,u,i,j} = C^S_{t,i,j}$ et ensuite définir une loi uniforme parmi ces entités, une probabilité nulle étant assignée aux autres individus.

Dans les MRP-IR, seule la variable cluster source existe. De ce fait, les appartenances des individus à des groupes spécifiques doivent être encodées à l'instanciation via la distribution de la variable de référence elle-même, empêchant de faire de l'inférence sur les clusters des individus. Dans les MRP-IC, puisque deux variables clusters

existent, dont une présente dans le type d'entité référencé, nous pouvons définir une distribution de probabilités pour une variable de référence dépendant de l'assignation courante des individus aux clusters (changer l'évidence pour ces assignations peut ainsi modifier les croyances calculées pour les associations).

6.4.3 Inférence des variables de référence

Dans un MRP-IC, les variables de référence sont conditionnées par deux variables clusters. Lors de la génération d'un RBP pour une instance donnée, la dépendance avec la variable cluster cible C^T mène à la création de multiples parents pour chaque instance de la variable de référence considérée puisque les valeurs de cluster de chaque entité potentiellement référencée doivent être connues pour déterminer quel est le sous-ensemble d'entités d'intérêt pour une association particulière. Or, un problème de performance important dans les modèles dirigés survient dans le cas où une variable possède de multiples parents, augmentant considérablement la dimension du modèle, c.-à-d. le nombre de paramètres à estimer. Pour limiter ce problème dans les MRP-IC, nous proposons de réaliser l'inférence de références en deux temps : d'abord, trouver les valeurs les plus probables de toutes les variables cluster c^T du RBP, incluant alors ces estimations comme évidences du modèle ; puis, déterminer les valeurs des variables r en considérant les évidences sur les c^T. Les parents étant connus pour chaque variable r, la seconde étape est ici très rapide.

6.5 Apprentissage

Nous nous intéressons maintenant à l'apprentissage des modèles, tant sur le plan des paramètres que de la structure (fonctions de partitions et dépendances probabilistes).

6.5.1 Estimation de paramètres

L'apprentissage de paramètres d'un MRP-IC est similaire à ce qui est fait pour les autres versions des MRP. Étant donnés $\mathcal{S}$ et Φ, les statistiques suffisantes peuvent être calculées pour une variable aléatoire V au niveau du MRP, en générant d'abord une vue tabulaire de V et ses parents, pour finalement compter les lignes dans la vue comme pour les réseaux bayésiens. L'estimation de paramètres peut ensuite être effectuée en suivant une approche par maximisation de la vraisemblance, ou par des approches bayésiennes utilisant des a priori sur les distributions (p.ex. uniforme ou Laplace dans un cas par défaut).

Il y a une simple exception toutefois : les distributions des variables de référence ne sont pas estimées puisqu'elles sont fonction des variables aléatoires clusters. De ce fait, elles sont automatiquement calculées à partir des valeurs de leurs variables clusters parentes.

6.5.2 Apprentissage de fonctions de partition

i Hypothèses sur les données

Les MRP-IC reposent sur des hypothèses faites sur les données. Nous supposons notamment dans les MRP-IC que l'essentiel de l'information portée par un type d'association provient de la topologie décrite par ses instances, et de l'information portée par

les entités concernées. Nous restreignons ici l'information topologique à la notion de communauté, mais nous pourrions également inclure d'autres variables permettant de préciser l'information telles que le concept de *rôle* dans une communauté par exemple, décrivant la position d'un individu à l'intérieur d'une communauté identifiée pour différencier par exemple les individus au cœur et ceux à la frontière de celle-ci. Les travaux effectués dans [110] montrent un exemple récent d'utilisation de la notion de rôle.

Différentes notions de communauté ont été données dans la littérature. Dans notre contexte, une communauté décrit une forme de densités d'associations entre les individus d'un sous-ensemble d'entités de chaque type concerné. De nombreux algorithmes tentent plus ou moins directement de découvrir ces structures. Les techniques de détection de communauté ou de partitionnement dans les graphes, ainsi que les approches de co-clustering sont des exemples adaptés pour cette tâche (cf. Chapitre 4).

Le lien entre détection de communautés et incertitude d'associations repose sur l'hypothèse que les parties plus denses d'un ensemble d'associations tendent à se densifier plus encore. De plus, nous supposons que nos types d'association décrivent des structures creuses, dans le but de détecter des parties denses à l'intérieur. Cette hypothèse est raisonnable, p.ex. dans les réseaux dits *sans échelle* [46], très communément rencontrés en pratique.

ii Hyper graphe support d'un type d'association

Nous définissons maintenant le concept d'*hyper graphe support* d'un type d'association instancié. Celui-ci définit la structure de données sur laquelle les algorithmes de partitionnement seront exécutés.

Définition 6.5. *Étant donné un schéma relationnel et un de ses types d'association $\mathcal{R}^j \in \mathbf{R}$ avec n références. Nous notons $\mathcal{R}^{j\rightarrow}_* = \{ref(\mathcal{R}^j, r_k) | r_k \in fk(\mathcal{R}^j)\}$ le multi ensemble des types d'entités référencés par $\mathcal{R}^j$. Nous notons également $\mathcal{R}^{j\rightarrow}$ l'ensemble obtenu à partir de $\mathcal{R}^{j\rightarrow}_*$, sans dupliquas. L'hyper graphe support $\mathcal{G}(\mathcal{R}^j, \mathcal{I}) = (\mathcal{V}^j(\mathcal{I}), \mathcal{E}^j(\mathcal{I}))$, pour $\mathcal{R}^j$ et l'instance $\mathcal{I} \in inst(\mathbf{R})$, est un hyper graphe à p modes où $p = |\mathcal{R}^{j\rightarrow}| \leq n$. C'est l'hyper graphe où chaque nœud correspond à une entité impliquée dans le type d'association, et où un hyper arc dénote une association référençant un ensemble d'entités respectant les contraintes du type d'association.*

Il est important de noter que l'ensemble des noeuds (resp. hyper arcs) est lié à l'ensemble des entités sans (resp. avec) dupliquas $\mathcal{R}^{j\rightarrow}$ (resp. $\mathcal{R}^{j\rightarrow}_*$). Cela est dû à l'existence de types d'association qui peuvent impliquer plusieurs fois le même type d'entité, l'exemple le plus simple étant un type d'association binaire entre des entités du même type, tel que celui exprimant la notion d'amitié entre personnes.

Intuitivement, l'hyper graphe support représente la structure graphique d'un type d'association instancié.

iii Matrices de similarité des individus d'un type d'entités

Utiliser le seul hyper graphe support pour réaliser du clustering, c.-à-d. faire du clustering topologique, peut mener à des problèmes de sur apprentissage et a été montré comme limité en pratique [52]. Souvent, nous avons besoin d'utiliser à la fois l'information des associations et celle des entités impliquées dans le but de trouver des groupes qui font plus de sens et sont plus robustes. L'information provenant des entités

seules peut être matérialisée par des matrices de similarité utilisant l'information des attributs des types concernés. Plus formellement, pour chaque schéma de relation $\mathcal{R}^j$, nous définissons une fonction de similarité $s_j : dom(att(\mathcal{R}^j))^2 \rightarrow \mathbb{R}+$ pour toute paire d'entités dans $\mathcal{R}^j$. Par extension, étant donnée une instance $\mathcal{I}$, $s_j(\mathcal{I})$ définit une matrice définie semi-positive $W_j(\mathcal{I})$ où chaque élément est une mesure de similarité pour deux individus dans $\mathcal{I}(\mathcal{R}^j)$. Notez que de nombreuses mesures de similarité peuvent être utilisées à cette étape pour la construction des matrices. Toutefois, pour des raisons évoquées plus loin, nous sommes davantage intéressés par les fonctions de similarité linéaires, c.-à-d. dont les valeurs peuvent être calculées comme une somme de contributions de fonctions de similarités ne prenant en compte qu'un seul attribut. La distance Euclidienne est ainsi non linéaire, mais son carré l'est puisque c'est la somme des carrés des différences pour chaque valeur d'attribut. Un autre exemple simple pour le cas continu est la norme $l1$. Dans le cas de variables discrètes non nécessairement ordonnées, une distance de Hamming peut être utilisée.

iv Apprentissage à partir d'hyper graphes support et de matrices de similarité

Dans les MRP-IC, nous sommes intéressés par les mécanismes d'apprentissage de fonctions de partition utilisant à la fois l'hyper graphe support d'un type d'association et les matrices de similarité des schémas de relation des types d'entités impliqués, dans le but d'apprendre simultanément toutes les fonctions de partition du type d'association. Formellement, pour tout type d'association $\mathcal{R}^j$, notre objectif est de trouver l'ensemble de fonctions de partitions $\Phi_{\mathcal{R}^{j\rightarrow}} = \{\phi_{j,f} | r_f \in fk(\mathcal{R}^j)\}$ d'une instance d'apprentissage $\mathcal{I}$ utilisant l'hyper graphe support $\mathcal{G}(\mathcal{R}^j, \mathcal{I})$ de $\mathcal{R}^j$ et l'ensemble des matrices de similarité $W_{\mathcal{R}^{j\rightarrow}}(\mathcal{I}) = \{W_{\mathcal{R}^l}(\mathcal{I}) | \mathcal{R}^l \in \mathcal{R}^{j\rightarrow}\}$.

La qualité d'un ensemble de fonctions de partitions $\Phi_{\mathcal{R}^{j\rightarrow}}$ relativement aux matrices de similarité concernées et à l'hyper graphe support est donnée par une fonction de coût $\gamma(\mathcal{G}_j, W_{\mathcal{R}^{j\rightarrow}}, \Phi_{\mathcal{R}^{j\rightarrow}})$.

v Propriétés souhaitées des algorithmes de fonction de partition

De nombreux algorithmes de partitionnement existent dans la littérature et nombre d'entre eux peuvent être utilisés pour retrouver des fonctions de partition dans le contexte des MRP-IC. Toutefois, les algorithmes choisis doivent de préférence avoir certaines propriétés. D'abord, pour un schéma de relation avec n ($n > 2$) contraintes de références, l'algorithme choisi doit être en mesure de fonctionner sur des hyper graphes ou des tenseurs d'ordre n (ou des graphes et matrices pour le cas où $n = 2$). De plus, il est important que l'algorithme soit adapté pour les données à n-modes. La différence est importante puisque la nature n-modale des hyper graphes support redéfinit le concept de communauté dans la plupart des cas [148, 78]. En effet, dans ces hyper graphes, les triangles n'existent pas et une clique est donc définie de façon plus relâchée. De plus, la densité maximale qui peut être atteinte pour une sous-structure est plus limitée que dans le cas d'un hyper graphe quelconque, ce qui impacte significativement les normalisations effectuées dans les scores. Cette propriété est d'autant plus importante que des travaux récents ont montré que la projection d'un hyper graphe à $n > 1$ modes était infidèle à la version originale, conduisant à la découverte de trop de sous-structures (car génère trop d'arcs) [146, 74, 75]. Aussi, recourir aux méthodes

générales ne supposant aucune contrainte n-modale après projection des données n'est pas suffisant.

vi Partitionner par référence et non par type d'entité

Dans les MRP-IC, nous ne définissons pas un unique partitionnement dans chaque type d'entité. À la place, nous définissons un partitionnement pour chaque paire (type d'association, contrainte de référence). C'est différent de ce qui peut être trouvé dans les approches de détection de communautés multi relationnelles, où les différents schémas de relation sont supposés faire partie d'un tout.

La première raison pour ce choix est que la détection de communautés basée sur un seul schéma de relation est représentative de son hyper graphe support. Cela est cohérent avec le concept d'apprentissage génératif où nous souhaitons trouver des corrélations entre des informations locales. Chaque variable aléatoire doit représenter quelque chose d'identifiable, ce qui n'est pas le cas d'une variable cluster de consensus qui représente à elle seule plusieurs types d'associations en une seule fois, sans aucune possibilité de récupérer l'information réelle à l'intérieur.

La seconde raison est que trouver un consensus entre plusieurs hyper graphes peut être insensé puisque plusieurs types d'associations peuvent exhiber des comportements très différents. En fait, trouver un consensus entre des partitionnements repose sur l'hypothèse que ceux-ci sont fortement corrélés, ce qui n'est pas nécessairement vrai. Forts de ce constat, de nombreux travaux visent ainsi à rechercher différents partitionnements des données comme autant de points de vue différents sur les individus [14, 184, 30]. Notons que dans un MRP-IC, même si nous ne construisons pas de variables consensus, nous pouvons toujours mesurer l'accord entre des paires de variables de clustering, et beaucoup de mesures existent dans ce but. Pour résumer :

- conserver l'information locale sur les communautés basée sur un seul type d'association à la fois permet de conserver l'information spécifique de sa topologie ;
- il est toujours possible de découvrir des corrélations entre les communautés de différents types d'association dans le but de trouver de possibles correspondances entre elles ;
- conserver les dépendances complètes entre les communautés de différents types d'association permet de supprimer l'hypothèse de point de vue unique sur les associations considérées et sur leurs corrélations.

Notez qu'une méthode récente de détection de communautés pour les réseaux hétérogènes multi relations est basée sur l'idée qu'un tel réseau peut être vu comme l'union de sous-réseaux, classiquement un pour chaque relation considérée, et donc la détection globale de communautés peut être réalisée par l'adaptation de l'algorithme Louvain en calculant une modularité composite : une somme pondérée des modularités de chaque sous-réseau [120]. Les résultats semblent prometteurs et confirment l'intérêt de conserver une information de communautés locale à une relation, au moins en premier lieu.

6.5.3 Couplage faible entre modèle et fonctions de partition

Comme pour les MRP-IR, nous établissons dans les MRP-IC un couplage faible entre les techniques de découverte de communautés et l'apprentissage de modèles probabi-

listes. Ceci permet de multiples possibilités, notamment :

- introduire différents algorithmes de clustering pour différents types d'association dédiés aux entités considérées (exemple pour les textes, images ou attributs géographiques). Cela est confirmé par Fortunato [60] qui indique qu'une perspective importante en détection de communautés est la création de techniques dédiées à des graphes aux propriétés spécifiques, alors que ces dernières sont perdues dans les méthodes générales ;

- introduire différentes méthodes de partitionnement pour différentes propriétés des hyper graphes (pondéré, orienté, dynamique, ...) ;

- introduire différents algorithmes de clustering pour le même hyper graphe, permettant de mesurer une éventuelle proximité entre leurs résultats et un consensus **pour ces mêmes données**.

Ce découplage va dans le sens d'une adaptabilité à plusieurs situations que l'on peut rencontrer. Le but est alors de trouver des sous-structures topologiques et d'apprendre un modèle probabiliste conjointement, sans toutefois contraindre complètement l'optimisation de l'une de ces tâches par l'autre, contrairement à une approche tout modèle par exemple [175] où une structure de modèle est fixée pour apprendre les variables latentes correspondant aux partitionnements avec l'algorithme EM.

6.5.4 Sélection de modèles

Dans cette section, nous nous intéressons à l'algorithme d'apprentissage d'un MRP-IC dans son ensemble dont le but est de trouver un ensemble de dépendances probabilistes, de fonctions de partition et de paramètres associés les plus fidèles possible à un jeu de données d'apprentissage, tout en permettant la généralisation du modèle sur d'autres jeux de données censés obéir aux mêmes lois probabilistes. Nous commençons par aborder le cadre général de l'algorithme, consistant en une heuristique gloutonne, puis nous détaillons plus précisément les diverses opérations utilisées pour balayer l'espace des hypothèses et abordons finalement l'évaluation des modèles permettant de comparer différents candidats entre eux.

i Recherche gloutonne

L'apprentissage de structure dans un MRP-IC peut être réalisé en utilisant une stratégie de recherche gloutonne. Débutant avec une structure vide, sans arc entre les variables aléatoires, ou à partir d'une structure initialisée, l'algorithme de recherche gloutonne alterne entre : 1) une phase où des modèles candidats sont générés par modifications légères du meilleur MRP-IC obtenu jusqu'ici, utilisant différents opérateurs de voisinage ; 2) une phase où les modèles candidats sont évalués et où les scores obtenus sont comparés pour ne conserver que le meilleur voisin qui devient alors le MRP-IC courant. Ce processus est contraint de façon à privilégier les modèles les plus simples en pénalisant les dépendances probabilistes faisant intervenir des séquences de référence plus longues, et en pénalisant les partitions, à la fois en fonction de la quantité d'information nécessaire pour calculer les résultats (nombre d'attributs) et en fonction du nombre de clusters. Les détails de l'algorithme principal sont donnés dans l'Algorithme 4 où s_{max} décrit la longueur maximale des séquences de références autorisées

pour la génération des voisins, et $\overset{*}{\mathcal{M}}_{\leq s}$ décrit l'ensemble des modèles candidats relativement à une contrainte de séquence de références de taille maximale s. La fonction $initPartitionFunctions(\mathcal{M})$ construit les fonctions de partition en utilisant uniquement l'information des associations (aucun attribut) et retourne le modèle $\mathcal{M}$ avec celles-ci. La fonction $neighbors(\mathcal{M}, s)$ retourne l'ensemble des voisins d'un modèle $\mathcal{M}$ utilisant divers opérateurs de voisinage, avec la contrainte de n'autoriser que la mise à jour de dépendances probabilistes impliquant une séquence de références de taille strictement égale à s. Finalement, la fonction $best(\mathcal{M}^{\mathcal{C}})$ retourne le modèle ayant le meilleur score parmi un ensemble de candidats, utilisant une fonction de score quelconque.

L'utilisation d'une heuristique gloutonne garantit uniquement la convergence vers un optimum local, et c'est également le cas de la méthode d'apprentissage proposée ici. Toutefois, l'utilisation de fonctions de co-clustering est en soi un avantage pour la qualité de la convergence puisqu'elle prend en compte l'intégralité des informations d'association pour calculer un ensemble de fonctions de partition pour un type donné, et non plus séparément chaque fonction de partition, avec le risque que leur union soit peu informative. De surcroît, cette approche permet de trouver les attributs corrélés à un type d'association sans avoir à parcourir l'ensemble des combinaisons de façon heuristique, ce qui peut mener plus facilement l'apprentissage des MRP-IR vers un mauvais optimum local.

Malgré tout, l'utilisation d'une heuristique gloutonne pour les MRP-IC rend probable la convergence vers un sous-optimum. Les problèmes rencontrés sont principalement les mêmes que pour l'apprentissage d'autres types de MRP. Au-delà de ces écueils habituels, il est important de noter que le choix précis des algorithmes de partitionnement a également un impact sur la convergence. En effet, la plupart des algorithmes de partitionnement étant eux-mêmes basés sur des heuristiques, ils convergent également vers des optima locaux de partitionnement. Aussi, si l'optimum local de partition trouvé par l'algorithme est mauvais, alors la structure qui pourra être apprise à partir de cela s'en trouvera très probablement impactée négativement. Une solution pour améliorer la qualité d'un résultat de partitionnement peut être de l'exécuter plusieurs fois, mais cela a un coût de calcul non négligeable. Une autre solution est l'utilisation de méthodes moins sensibles à la structure exacte des données, comme les approches généralisant les données au fur et à mesure du partitionnement, p.ex. Louvain [16] ou les partitionnements METIS [94] et GRACLUS [50].

D'autres techniques usuelles en optimisation, non explorées dans cette thèse, pourraient être envisagées pour améliorer la convergence de l'algorithme, telles que les solutions généralement utilisées dans le cadre d'une optimisation heuristique. Nous pouvons citer à titre d'exemple les techniques à base de recuit simulé [101] et d'initialisations aléatoires [170].

ii Opérateurs d'exploration de l'espace

L'algorithme de recherche gloutonne tente d'améliorer un modèle $\mathcal{M}$ qui a été déterminé comme le meilleur candidat jusqu'ici par rapport à une mesure d'évaluation donnée, en choisissant la modification locale de ce modèle qui maximise le même score, parmi un ensemble de candidats qui sont tous des variantes de $\mathcal{M}$. Les variantes sont obtenues en appliquant différents opérateurs de voisinage. La quantité et nature de ceux-ci peut varier dans le contexte des MRP-IC, surtout pour ceux menant à des mises à jour de fonctions de partition dépendant de la nature des algorithmes de parti-

Algorithme 4 Algorithme glouton pour l'apprentissage de MRP-IC

Paramètres : $\mathcal{M}_0$: le modèle de départ, s_{max} : la taille maximale de séquences de références

Retourne : $\overset{*}{\mathcal{M}}$: un optimum local obtenu à partir de $\mathcal{M}_0$

$s \leftarrow -1$
$\overset{*}{\mathcal{M}}_{\leq s} \leftarrow \mathcal{M}_0$
Si les fonctions de partition de $\overset{*}{\mathcal{M}}_{\leq s}$ n'existent pas **Alors**
$\overset{*}{\mathcal{M}}_{\leq s} \leftarrow initPartitionFunctions(\overset{*}{\mathcal{M}}_{\leq s})$
Fin Si
Répéter
$s \leftarrow s + 1$
$\overset{*}{\mathcal{M}}_{\leq s} \leftarrow \overset{*}{\mathcal{M}}_{\leq s-1}$
Répéter
$\mathcal{M}^{\mathcal{C}}_{\leq s} \leftarrow neighbors(\overset{*}{\mathcal{M}}_{\leq s}, s)$
$\overset{*}{\mathcal{M}}_{\leq s} \leftarrow best(\mathcal{M}^{\mathcal{C}}_{\leq s})$
Jusqu'à $\overset{*}{\mathcal{M}}_{\leq s}$ converge
Jusqu'à $s >= s_{max}$
Retourner $\overset{*}{\mathcal{M}}_{\leq s}$

tionnement utilisés.

Les opérateurs les plus basiques pour la structure graphique du modèle probabiliste sont également applicables ici : add_edge, $remove\_edge$ et $reverse\_edge$, menant respectivement à l'ajout, la suppression ou l'inversion d'un arc dans la structure graphique du MRP-IC. Ces opérateurs fonctionnent de la même façon que pour les MRP-IR.

Nous nous focalisons maintenant sur des opérations moins triviales liées à la mise à jour des fonctions de partition. Nous présentons dans cette thèse un ensemble d'opérations agissant sur les ensembles d'attributs de fonction de partition, mais d'autres opérations peuvent être envisagées en fonction des algorithmes de clustering utilisés.

iii Mettre à jour les attributs de fonction de partition

L'information d'associations seule ne permet pas toujours d'obtenir les communautés les plus robustes pour un jeu de données relationnel et de nombreux travaux en détection de communauté enrichissent les données d'associations entre individus avec la description interne de ceux-ci dans le but de construire des clusters plus robustes et significatifs. Toutefois, tous les attributs d'un individu ne sont pas nécessairement corrélés avec son comportement à l'égard d'un type d'association. De plus, il paraît raisonnable de vouloir limiter le nombre d'attributs à ajouter pour le partitionnement dans le but de minimiser la quantité d'information nécessaire pour effectuer des prédictions à partir des seules fonctions de partition à propos d'individus non observés. Ce problème est un problème de sélection de variables pour le partitionnement. Dans l'approche que nous décrivons ensuite, nous considérons que l'initialisation des fonctions de partition a été réalisée, consistant en l'exécution des divers algorithmes de partitionnement sans utiliser d'attributs. L'approche est ensuite basée sur des opérations de

voisinage, utilisant l'information des dépendances du modèle courant pour en déduire les meilleurs attributs candidats à ajouter dans l'algorithme de partitionnement. Nous définissons plus précisément deux opérateurs : $add\_attribute$ et $remove\_attribute$.

add_attribute Considérant un type d'association $\mathcal{R}^j$ avec $n \geq 2$ références et $\mathcal{R}^{j\rightarrow}$, l'ensemble des entités référencées sans dupliqua. Il y a alors n fonctions de partition après l'étape d'initialisation. L'ensemble complet des attributs candidats que nous pouvons ajouter pour améliorer la fonction de partition est composé de l'union de tous les attributs de tous les types d'entités dans $\mathcal{R}^{j\rightarrow}$. Trouver le meilleur sous-ensemble d'attributs peut être réalisé en testant toutes les possibilités, avec à chaque fois un réapprentissage des fonctions de partition, mais ceci n'est pas performant.

Nous choisissons à la place une autre stratégie et décidons de choisir des attributs en fonction de leurs corrélations avec le partitionnement obtenu jusqu'ici pour ensuite recalculer ce dernier en utilisant les meilleurs attributs selon ce critère. Une idée simple en ce sens serait d'utiliser l'information mutuelle normalisée du partitionnement induit par un attribut relativement au partitionnement obtenu grâce à l'information des associations. Ainsi, pour chaque schéma de relation $\mathcal{R}^i \in \mathcal{R}^{j\rightarrow}$, son affinité avec la variable de cluster correspondante est calculée, formellement $nmi(A_{i,k}, C^T_{i,j,f})$ pour toute variable aléatoire attribut $A_{i,k}$ correspondant à l'attribut $\mathcal{A}_k \in att(\mathcal{R}^i)$. Toutefois, cette approche naïve ne permet pas de faire la différence entre une dépendance probabiliste directe et indirecte, contrairement à ce que l'on peut identifier avec un modèle graphique probabiliste. Aussi, nous choisissons une autre méthode exploitant les données graphiques du modèle en cours d'apprentissage afin de limiter le nombre d'attributs ajoutés aux fonctions de partition. Restreindre l'ensemble des attributs de fonction de partition possibles à ceux directement dépendants de la variable cluster cible C^T, soit en tant que parent ou enfant, peut ainsi être une meilleure intégration de la mise à jour des attributs de fonction de partition à l'algorithme de recherche gloutonne, tout en limitant le nombre de candidats et de ré exécution des algorithmes de partitionnement, ainsi que l'impact de cet opérateur sur les mises à jour consécutives de la structure graphique et le nombre de calculs de similarités requis.

L'Algorithme 5 présente l'impact de l'ajout d'un attribut à une fonction de partition sur la structure d'un MRP-IC et sur la variation de score du modèle résultant de cette opération. La Figure 6.4 montre un exemple de séquence d'itérations de l'algorithme glouton avec l'opérateur d'ajout d'attributs, sur une version simplifiée du domaine UMA.

Mettre à jour les similarités entre les individus d'un type d'association, après avoir ajouté un attribut de fonction de partition, peut être réalisé de façon efficace dans le cas d'une fonction de similarité linéaire. Dans ce cas, la nouvelle similarité entre deux individus est seulement mise à jour par l'ajout d'une contribution à la similarité précédente. Nous proposons plus précisément la méthode de mise à jour suivante. Considérant s_A, la fonction de similarité entre deux entités ne prenant en compte que l'ensemble de leurs attributs A. Si cette fonction est linéaire, alors il existe une fonction de similarité s_{A_i} pour chaque attribut $A_i \in A$ donnant la mesure de similarité entre deux entités à l'égard de ce seul attribut. Aussi, nous pouvons écrire : $s_A(e1, e2) = \sum_{A_i \in A} s_{A_i}(e1, e2)$. Par extension, considérant un paramètre de régularisation λ_A, permettant d'équilibrer l'importance des mesures de similarité par rapport à l'information de l'hyper graphe

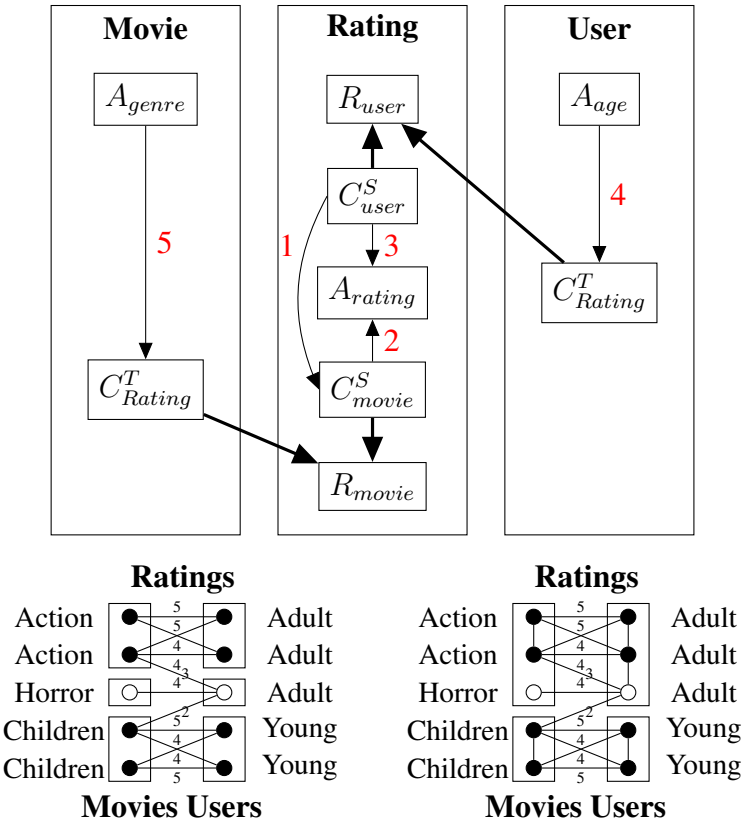


FIGURE 6.4 – Exemple simple d'apprentissage de MRP-IC. Commençant avec une structure vide, avec les seuls parents des variables de référence définis (en gras), nous commençons par apprendre les fonctions de partition des données du type d'association *Rating* et obtenons les résultats en bas à gauche, où les deux nœuds blancs forment à eux seuls leur propre partie (l'utilisateur blanc est connecté de façon égale à chaque cluster de films, tandis que le film blanc est connecté à un seul utilisateur lui même dans son propre cluster). Nous mettons ensuite à jour les CPD des variables cluster de façon adéquate (ils n'ont pas de parent à cette étape). Ensuite, nous ajoutons itérativement des arcs dans l'ordre donné par les flèches sur le MRP-IC. Le graphe connecté encode le savoir que les associations sont plus denses entre certaines paires de clusters (1), que la valeur des notes dépend des paires de clusters impliqués dans l'association (2,3) et que les attributs de chaque type d'entité sont très corrélés aux valeurs de partitions (4,5). À ce moment, nous pouvons ajouter les attributs directement dépendants aux fonctions de partitions pour améliorer les résultats d'apprentissage. Le graphe support (en bas à droite) du type d'association est mis à jour en ajoutant des arcs de similarité entre les individus d'une même valeur d'attributs et de nouvelles fonctions de partition sont calculées. L'utilisateur blanc n'est alors plus à la frontière entre les deux clusters et peut joindre le cluster des adultes, alors que le film blanc peut le suivre, puisqu'il est maintenant associé à l'utilisateur blanc dont le cluster a été mis à jour.

Algorithme 5 Algorithme de l'opérateur *add_attribute* pour les MRP-IC.

Paramètres :
$(\mathcal{S}, \Phi)$: la structure d'un MRP-IC avec $\mathcal{S} = (\mathcal{V}, pa)$,
$\phi_{i,n}$: une fonction de partition de Φ,
$A_{k,j}$: un attribut de $\mathcal{R}^k = ref(\mathcal{R}^i_a, r_n)$ pour $r_n \in fk(\mathcal{R}^i)$,
$\mathcal{I}$: une instance d'apprentissage
Retourne : $(\mathcal{S}', \Phi')$: la structure mise à jour, Δ_s : la variation de score.

$\mathcal{A}(\phi_{i,n}) \leftarrow \mathcal{A}(\phi_{i,n}) \cup A_{k,j}$ #Mettre à jour les attributs de $\phi_{i,n}$
$\mathcal{A}_i \leftarrow \{\mathcal{A}(\phi_{i,l}) | r_l \in fk(\mathcal{R}^i)\}$ #Récupérer les ensembles d'attributs des $\phi_{i,l} \in \Phi_i$
$\Phi'_{\mathcal{R}^{i\rightarrow}} \leftarrow part(\mathcal{C}_i, \mathcal{A}_i, \mathcal{R}^{i\rightarrow}, \mathcal{I})$ #Recalculer les fonctions de partition de $\mathcal{R}^i$
$\Phi' \leftarrow (\Phi \backslash \Phi_{\mathcal{R}^{i\rightarrow}}) \cup \Phi'_{\mathcal{R}^{i\rightarrow}}$ #Mettre à jour l'ensemble des fonctions de partition
$C_i \leftarrow \bigcup_{\phi_{i,l} \in \Phi_i} \{C^T_{ref(\mathcal{R}^i, r_l), i, l}, C^S_{i,l}\}$ #Récupérer les variables cluster impactées
#Ajouter les enfants et l'attribut ajouté pour le calcul du score
$V_i \leftarrow \{A_{k,j}\} \cup C_i \cup enfants(C_i, \mathcal{S})$
#La variation de score est celle des V_i + celle des a priori de fonctions de partition
$\Delta s \leftarrow score\_delta(V_i, (\mathcal{S}, \Phi), (\mathcal{S}', \Phi'))$
#Si la variable $C^T_{k,i,n}$ est un parent de $A_{k,j}$, inverser l'arc.
#Si l'inversion rend la structure cyclique, l'ajout d'attribut est impossible.
Si $(id, \emptyset, C^T_{k,i,n}) \in pa(A_{k,j})$ **Alors**
  $pa' \leftarrow pa$
  $pa'(A_{k,j}) \leftarrow pa'(A_{k,j}) \backslash (id, \emptyset, C^T_{k,i,n})$
  $pa'(C^T_{k,i,n}) \leftarrow pa'(C^T_{k,i,n}) \cup (id, \emptyset, A_{k,j})$
  $\mathcal{S}' = (\mathcal{V}, pa')$
  Si $acyclique(comp(\mathcal{S}'))$ **Alors**
    Retourner $(\mathcal{S}', \Phi'), \Delta_s$
  Sinon
    Retourner $(\mathcal{S}, \Phi), 0$
  Fin Si
Fin Si
Retourner $(\mathcal{S}', \Phi'), \Delta_s$

support, nous pouvons alors écrire :

$$s_A(e1, e2) = \lambda_A(\sum_{A_i \in A} s_{A_i}(e1, e2)). \tag{6.2}$$

Si nous souhaitons maintenant mettre à jour les similarités après l'ajout de $A_j \notin A$, et considérant sa fonction de similarité s_{A_j}, nous pouvons calculer la nouvelle similarité de façon incrémentale :

$$s_{A \cup A_j}(e1, e2) = \lambda_{A \cup A_j}(\lambda_A^{-1} s_A(e1, e2) + s_{A_j}(e1, e2)) \tag{6.3}$$

où $\lambda_{A \cup A_j}$ est le nouveau paramètre de régularisation pour les similarités mises à jour, c.-à-d. celui qui préserve l'égalité suivante :

$$\sum_{\forall (e1,e2)} s_{A \cup A_j}(e1, e2) = \sum_{\forall (e1,e2)} s_A(e1, e2).$$

Notez que les paramètres λ_A, et les valeurs de similarité pour tous les attributs obtenus avec les fonctions s_{A_i} peuvent être mis en cache après avoir été calculés une

Algorithme 6 Algorithme de l'opérateur *remove_attribute* pour les MRP-IC.

Paramètres :
$(\mathcal{S}, \Phi)$: la structure d'un MRP-IC avec $\mathcal{S} = (\mathcal{V}, pa)$,
$\phi_{i,n}$: une fonction de partition de Φ,
$A_{k,j}$: un attribut de $\mathcal{A}(\phi_{i,n})$,
$\mathcal{I}$: une instance d'apprentissage
Retourne : $(\mathcal{S}', \Phi')$: la structure mise à jour, Δ_s : la variation de score.

$\mathcal{A}(\phi_{i,n}) \leftarrow \mathcal{A}(\phi_{i,n}) \backslash A_{k,j}$ #Mettre à jour les attributs de $\phi_{i,n}$
$\mathcal{A}_i \leftarrow \{\mathcal{A}(\phi_{i,l}) | r_l \in fk(\mathcal{R}^i)\}$ #Récupérer les ensembles d'attributs des $\phi_{i,l} \in \Phi_i$
$\Phi'_{\mathcal{R}^{i\rightarrow}} \leftarrow part(\mathcal{C}_i, \mathcal{A}_i, \mathcal{R}^{i\rightarrow}, \mathcal{I})$ #Recalculer les fonctions de partition de $\mathcal{R}^i$
$\Phi' \leftarrow (\Phi \backslash \Phi_{\mathcal{R}^{i\rightarrow}}) \cup \Phi'_{\mathcal{R}^{i\rightarrow}}$ #Mettre à jour l'ensemble des fonctions de partition
$C_i \leftarrow \bigcup_{\phi_{i,l} \in \Phi_i} \{C^T_{ref(\mathcal{R}^i, r_l), i, l}, C^S_{i,l}\}$ #Récupérer les variables cluster impactées
#Ajouter les enfants pour le calcul du score
$V_i \leftarrow C_i \cup enfants(C_i, \mathcal{S})$
#La variation de score est celle des V_i + celle des a priori de fonctions de partition
$\Delta s \leftarrow score\_delta(V_i, (\mathcal{S}, \Phi), (\mathcal{S}', \Phi'))$
#La suppression d'un attribut ne provoque pas de problème d'acyclicité : fin.
Retourner $(\mathcal{S}', \Phi'), \Delta_s$

première fois. Ceci simplifie les calculs requis après chaque ajout d'attribut, si celui-ci a déjà été ajouté.

Pour finir, il est important de noter que l'opérateur *add_attribute* pourrait être étendu à n'importe quelle variable (hors référence) du MRP-IC, apprise par exploration de l'espace des fonctions d'agrégation et des séquences de référence. Il pourrait également être utilisé pour régulariser les fonctions de partition avec les variables cluster d'une autre fonction de partition, puisque les variables de cluster cible peuvent être considérées comme des attributs (topologiques) une fois leurs fonctions de partition fixées.

remove_attribute Supprimer un attribut d'une fonction de partition peut également être nécessaire dans le cas où, après plusieurs itérations, supprimer un arc d'une variable attribut vers une variable cluster, alors que cet attribut est également dans les attributs de fonction de partition de la variable cluster, mène à un meilleur score. Ceci peut en effet être la preuve que l'attribut de fonction de partition n'est plus nécessaire pour la tâche de partitionnement. Dans ce cas, supprimer un attribut de fonction de partition doit être une option à considérer. L'impact d'une telle opération est proche de celui de $add\_attribute$ sauf qu'aucune vérification d'acyclicité n'est nécessaire. L'Algorithme 6 donne le détail de l'impact d'une telle opération. Encore une fois, les similarités peuvent être mises à jour de façon incrémentale, utilisant des valeurs de similarité mises précédemment en cache pour les attributs supprimés et les paramètres de régularisation.

Comme énoncé plus haut, l'ensemble des opérations concernant la mise à jour des attributs de fonction de partition peut utiliser l'information de la structure graphique pour discriminer l'ensemble des attributs d'intérêt pour améliorer les fonctions de partition, afin de distinguer les dépendances directes des dépendances indirectes dans le

Algorithme 7 Algorithme glouton pour l'apprentissage de MRP-IC avec opérateurs sur les attributs de fonctions de partition

Paramètres : $\mathcal{M}_0$: le modèle de départ, s_{max} : la taille maximale de séquences de références
Retourne : $\overset{*}{\mathcal{M}}$: un optimum local obtenu à partir de $\mathcal{M}_0$

$s \leftarrow -1$
$\overset{*}{\mathcal{M}}_{\leq s} \leftarrow \mathcal{M}_0$
Si les fonctions de partition de $\overset{*}{\mathcal{M}}_{\leq s}$ n'existent pas **Alors**
  $\overset{*}{\mathcal{M}}_{\leq s} \leftarrow initPartitionFunctions(\overset{*}{\mathcal{M}}_{\leq s})$
Fin Si
Répéter
  $s \leftarrow s + 1$
  $\overset{*}{\mathcal{M}}_{\leq s} \leftarrow \overset{*}{\mathcal{M}}_{\leq s-1}$
  Répéter
    $\mathcal{M}^{\mathcal{C}}_{\leq s} \leftarrow neighborsDAG(\overset{*}{\mathcal{M}}_{\leq s}, s)$
    $\overset{*}{\mathcal{M}}_{\leq s} \leftarrow best(\mathcal{M}^{\mathcal{C}}_{\leq s})$
    $\mathcal{M}^{\mathcal{C}}_{\leq s} \leftarrow neighborsPF(\overset{*}{\mathcal{M}}_{\leq s}, s)$
    $\overset{*}{\mathcal{M}}_{\leq s} \leftarrow best(\mathcal{M}^{\mathcal{C}}_{\leq s})$
  Jusqu'à $\overset{*}{\mathcal{M}}_{\leq s}$ converge
Jusqu'à $s >= s_{max}$
Retourner $\overset{*}{\mathcal{M}}_{\leq s}$

graphe, ne choisissant alors que les dernières. Ceci suggère une mise à jour de l'algorithme glouton à utiliser, où les opérations de mise à jour du graphe et les opérations de mise à jour des fonctions de partition peuvent être alternées. L'Algorithme 7 montre cette mise à jour. Les fonctions *neighborsDAG* et *neighborsPF* remplacent la fonction *neighbors* pour séparer la génération des candidats obtenus respectivement par voisinage sur les structures graphiques et sur les fonctions de partition.

L'approche de sélection de variables que nous proposons peut être vue comme une approche intégrée de *sélection par étapes en avant* (*forward stepwise selection method* en anglais) [57, 55] où nous commençons avec un ensemble d'attributs vides pour chaque fonction de partition et où nous alternons entre des phases d'ajout (*forward selection*) un à un des attributs paraissant significatifs pour expliquer une fonction de partition, et des phases de suppression (*backward elimination*) d'attributs qui peuvent devenir inutiles plus tard. Toutefois, deux différences existent avec les méthodes de sélection de variables énoncées : 1) la variable cible est construite et reconstruite au fil de l'apprentissage ; 2) l'espace des variables admissibles est réduit en utilisant les dépendances décrites par la structure graphique du modèle.

Comme énoncé plus haut, de nombreux autres opérateurs pourraient également être considérés au-delà de la modification des attributs de fonctions de partition, dépendant de la nature des algorithmes de partitionnement utilisés. Par exemple, l'espace du nombre de clusters peut être considéré si les temps de calcul de l'algorithme utilisé le permettent et si ce paramètre existe pour l'algorithme. De même, il pourrait être in-

téressant de construire un opérateur permettant de parcourir les valeurs d'un paramètre d'un score paramétrique comme les α-divergences où α peut évoluer et permettre une transition continue entre plusieurs mesures connues.

6.5.5 Évaluation des modèles

L'évaluation des modèles pour les MRP-IC est identique à celle que nous avons définie pour les MRP-IR dans le chapitre 3. Nous rappelons que la littérature sur les MRP-IR ne détaille pas cette évaluation, et ne propose notamment pas d'utiliser un *a priori* sur les fonctions de partition. Nous décidons au contraire d'en utiliser pour l'évaluation des MRP-IR et MRP-IC. Nos choix se portent principalement sur celui d'Ewens-Pitman [129, 29] dont la limite à l'infini est le processus de restaurant chinois, et celui de Jensen-Liu [92] proposé à l'origine afin de supprimer l'effet *rich get richer* du processus de restaurant chinois, où un groupe a plus de probabilités d'être choisi à une itération donnée s'il est plus gros que ses concurrents.

6.5.6 Comparaison des scores pour un MRP et un RB

Dans un réseau bayésien classique, après avoir évalué l'impact d'une opération de voisinage sur la mise à jour du score global, en ne recalculant que la partie modifiée localement, chaque voisin candidat est associé à sa variation de score et le choix de la modification à réaliser effectivement correspond généralement à celle menant au meilleur gain si au moins une des variations offre un tel gain. La comparaison des scores se fait en valeur absolue, ce qui ne pose pas de problème particulier. Toutefois, cela n'est pas aussi évident pour les MRP. En effet, dans la plupart des cas, le calcul du score dépend du nombre d'individus, puisqu'il se décompose en une somme de log-vraisemblances pour chaque individu de l'ensemble considéré. Lorsque le nombre d'individus est constant, comme pour les réseaux bayésiens classiques, cela ne pose pas de problème puisque l'on compare des scores faisant intervenir le même nombre d'individus. Toutefois, dans un MRP, une mise à jour de score locale ne fait pas toujours intervenir le même nombre d'individus et est au contraire propre à chaque schéma de relation pour une instance considérée. En effet, le nombre de tuples à considérer, pour le calcul du score d'une variable sachant ses parents, correspond à la quantité d'individus propre au schéma de relation de cette variable, puisque la vue générée pour calculer les paramètres de la distribution contient systématiquement un tuple par enfant. Aussi, lorsque l'algorithme glouton applique différents opérateurs de voisinage pour générer plusieurs candidats, l'étape d'évaluation leur affecte des variations de score pouvant être dans des échelles différentes, entraînant un biais dans le choix des candidats maximisant cette variation de score. Pour s'en convaincre, il est intéressant de noter que le nombre d'individus d'un schéma de relation pour une instance est fonction du nombre de références qu'elle contient. Un type d'association contenant un ensemble de références $\{f_1, \ldots, f_r\}$ contient ainsi un nombre d'individus en $\mathcal{O}(n_1 n_2 \ldots n_r)$ si aucun doublon n'est possible, où n_i est le nombre d'entités du type référencé par f_i pour l'instance considérée. Il est donc probable que le nombre réel d'individus d'un type d'association soit bien plus grand que le nombre d'individus des types d'entités impliqués et donc des scores d'échelle potentiellement différente. Pour pallier ce problème, nous comparons pour les MRP-IC les scores de façon relative et choisissons à chaque étape la variation de score amenant le plus fort gain relatif, c.-à-d. celui multipliant le plus le score local précédent. Formellement, nous proposons de calculer une variation

de score de la façon suivante pour les MRP :

$$\Delta s_{MRP} = \frac{s_1 - s_0}{s_0},$$

alors qu'elle se calcule de la façon suivante pour un RB classique :

$$\Delta s_{RB} = s_1 - s_0,$$

où s_0 est le score précédant l'application de l'opération à évaluer et s_1 est le score en résultant.

6.6 Travaux connexes

Notre travail est proche de celui de Getoor et al. [66] à propos des MRP avec incertitude de structure. Plus précisément, nous étendons le travail effectué sur le paradigme d'incertitude de référence, dans le but d'apprendre des modèles plus précis, tout en réduisant les contraintes imposées sur l'espace des modèles considérés, utilisant des stratégies de fonction de partition pour gérer l'incertitude de liens. Nous introduisons en outre explicitement l'usage d'un a priori sur les fonctions de partition, contrairement à ce qui est fait pour les MRP-IR. Ce paradigme est notablement non étudié dans les autres modèles graphiques probabilistes, où le paradigme d'incertitude d'existence est très répandu. Nous affirmons qu'une stratégie d'apprentissage adéquate des fonctions de partition peut permettre d'améliorer à la fois la précision et le temps d'exécution des algorithmes d'apprentissage de structure, ces méthodes permettant d'offrir des résumés de la topologie décrite par les associations d'intérêt.

Il est important de noter que le problème de clustering avec les MRP a été traité dans citeTaskar2001, mais en tant qu'objectif, à partir d'un MRP orienté attributs. Dans notre travail, nous faisons le contraire, à savoir utiliser des méthodes de clustering pour apprendre des MRP-IC.

La définition explicite de groupes latents pour l'apprentissage de modèles génératifs a également été investiguée dans plusieurs papiers. Dans [109], les auteurs définissent un modèle génératif pour gérer l'existence de liens parmi des individus en fonction de leur appartenance à des groupes. Dans [108], les auteurs définissent un modèle autorisant la génération de liens de différents types avec des données temporelles, étant donnée une appartenance des individus à des groupes. Plus tard, Neville et Jensen [138] ont défini un modèle de groupes latents décrivant une distribution de probabilités sur les groupes (appris par clustering spectral), les attributs et les liens entre eux, et ont montré de bons résultats. Ce dernier travail est proche du nôtre en ce sens qu'il définit une distribution sur les liens et les groupes d'individus en plus des attributs présents dans les données, et utilise des algorithmes de clustering dans le but de rendre la tâche d'apprentissage calculable. Toutefois, nos travaux diffèrent sur certains points. Tout d'abord, le modèle de cet article est uniquement défini pour un seul type d'entités et un seul type d'association non dirigé. Ensuite, la structure est fixée et suppose que l'appartenance des entités aux clusters influence à la fois leurs valeurs d'attributs et leurs associations entre eux. Même si cela est possible dans des jeux de données unipartis, avec un petit nombre de types d'associations et d'attributs, c'est nettement moins probable dans des domaines multi relationnels plus complexes, où

différents sous ensembles d'attributs et de types d'associations sont mieux expliqués par des points de vue différents et donc des partitions différentes des données. Nous proposons ici une méthode d'apprentissage de structure utilisant différents partitionnements pour différents types d'association, ce qui nous permet de découvrir l'influence des différentes fonctions de partition (et donc des différents points de vue) sur les différentes variables et sur les probabilités d'associations entre individus. Pour aller plus loin, cela peut également nous permettre d'étudier l'influence de la topologie d'un type d'association sur un autre, pour découvrir des connaissances entre les associations elles-mêmes.

L'utilisation d'algorithmes de clustering pour apprendre des modèles graphiques probabilistes de second ordre [100] a également été récemment investigué par Kok et Domingos pour les réseaux logiques de Markov (MLN) dans [103] et [104]. Dans [103], du clustering agglomératif est réalisé dans le but de regrouper des entités tendant à avoir des patrons d'association proches avec d'autres entités, dans le but de construire un hyper graphe lifté à partir du modèle à plat défini par la base de données. Ensuite, les clauses candidates pour la sélection de modèles sont recherchées par l'utilisation de techniques de découverte de chemins relationnels dans cet hyper graphe. Dans [104], la construction de l'hyper graphe est réalisée par la découverte de motifs structurels dans le modèle à plat original, regroupant les différentes symétries trouvées dans les données. Cette dernière approche permet de trouver de plus longues clauses candidates que la méthode précédente. Ces travaux sont différents des nôtres, puisqu'ils sont focalisés sur les modèles MLN, et proposent des méthodes initialisées sur le modèle à plat complet d'une base de données selon une approche *bottom-up*, alors que nous apprenons de façon séparée les fonctions de partition pour chaque type d'association, permettant d'adopter une stratégie basée sur le principe «diviser pour mieux régner», moins coûteuse en mémoire.

De nombreux travaux dans le cadre des modèles graphiques (liftés ou non) essaient de trouver des clusters d'individus, en définissant à la main des structures de modèles, incluant les variables latentes supposées représenter les variables cluster, pour ensuite utiliser des méthodes d'apprentissage de paramètres avec données manquantes (tel que EM) ou des méthodes d'échantillonnage (p.ex. Gibbs) dans le but de trouver les assignations aux clusters, p.ex. les travaux récents de [10, 9] dans un contexte de recommandation. Même si ces méthodes peuvent être utilisées pour les fonctions de partition de nos modèles, la tâche est différente de ce que nous faisons en apprentissage de MRP-IC, où l'usage d'un algorithme de clustering externe permet de trouver une structure plutôt que de la définir.

Un autre sujet lié est celui du clustering multi relationnel [182], dont le but est de trouver un partitionnement pour chaque type d'entité étant donnés tous les types d'associations entre individus. Cette famille de travaux est différente des nôtres puisque nous nous situons dans un contexte d'apprentissage de structure génératif, avec un objectif d'apprentissage localement précis de connaissances, nécessitant ainsi des fonctions de partition localement précises pour réaliser de la prédiction de liens pour toute association considérée. Par conséquent, nous choisissons d'apprendre une fonction de partition pour chaque paire (type d'entité, type d'association) au lieu de ne considérer qu'une seule fonction de partition par type d'entité. Les variables cluster de différentes fonctions de partition peuvent encore être dépendantes les unes des autres, et dans un cas extrême, les mêmes résultats que pour un clustering multi relationnel peuvent

		NMI $\times 10^2$			
Données	Algo.	Association 1 (A1)		Association 2 (A2)	
		FP1	FP2	FP1	FP2
1	IR	$.9 \pm .7$	0 ± 0	21 ± 40	100 ± 0
	IC	$1.3 \pm .5$	0 ± 0	$1.2 \pm .4$	0 ± 0
5D	IR	4 ± 7	18 ± 0	20 ± 4.7	0 ± 0
	IC	80 ± 35	0 ± 0	74 ± 37.6	0 ± 0
5S	IR	$.6 \pm .7$	0 ± 0	$.4 \pm .4$	0 ± 0
	IC	2.2 ± 1.6	0 ± 0	2.9 ± 1.1	0 ± 0

TABLE 6.1 – Résultats d'évaluation des fonctions de partition de MRP-IC et MRP-IR appris sur 3 jeux de données synthétiques.

Données	Algo.	LL $\times 10^{-3}$		T (s.)
		A1	A2	
1	IR	-420 ± 0.9	-416 ± 2	3 459
	IC	-368 ± 6	-367 ± 5	636
	IC opt	-374 ± 2.1	-370 ± 0.9	155
5D	IR	-848 ± 16	-858 ± 2.7	8 015
	IC	-789 ± 12.2	-785 ± 16.4	1 353
	IC opt	-686 ± 15	-669 ± 18	522
5S	IR	-81 ± 1.6	-81 ± 2.1	9 652
	IC	-70 ± 1.3	-72 ± 1.2	472
	IC opt	-62.6 ± 1	-62.2 ± 1.2	240

TABLE 6.2 – Résultats d'évaluation des structures de MRP-IC et MRP-IR apprises sur 3 jeux de données synthétiques.

être obtenus si toutes les variables de fonctions de partition sont dépendantes de façon déterministe.

6.7 Expériences sur données synthétiques

Dans cette section, nous évaluons et comparons les résultats d'apprentissage de structure des MRP-IC et MRP-IR. Plus précisément, nous considérons le résultat de 3 paramétrages d'algorithmes sur 3 jeux de données synthétiques différents. Les algorithmes sont : 1) apprentissage de structure de MRP-IR avec clustering par produit cartésien des attributs ; 2) apprentissage de structure de MRP-IC utilisant l'algorithme de Louvain pour le clustering sur le graphe construit à partir de l'hyper graphe support enrichi avec les similarités intra mode (calculées avec la distance de Hamming) pour la prise en compte des attributs de fonction de partition ; 3) apprentissage de structure de MRP-IC avec les fonctions de partition connues, dénoté MRP-IC Opt, utilisé comme une borne supérieure pour l'évaluation. Tous les jeux de données ont deux types d'entités avec 10 variables attributs et 2 types d'association avec seulement deux références vers les deux types d'entités. Les instances générées contiennent 1000 entités pour chaque type et un nombre variable d'associations de chaque type, générées comme suit. Nous considérons, pour chaque type d'entité, deux partitionnements de leurs individus, soit un pour chaque type d'association. Nous considérons ensuite deux paramètres pin et $pext$ (avec $pin > pext$), définissant respectivement la probabilité d'existence d'une

association entre deux entités d'un même cluster, et entre deux entités de clusters différents. Les associations sont alors générées en considérant chaque paire d'entités et leurs valeurs de clustering, puis en générant les associations entre elles, utilisant pin ou $pext$ en fonction des valeurs de cluster. Ce processus de génération est proche d'un graphe de type Erdös Rényi, mais avec deux paramètres.

Dans le premier jeu de données (1), un attribut est choisi aléatoirement pour expliquer de façon déterministe l'appartenance aux clusters de chaque entité relativement à chaque type d'association. Huit attributs sont donc indépendants dans le modèle pour chaque type d'entité. Le jeu de données est simple et supposé favorable à l'approche MRP-IR avec produit cartésien. Nous nous attendons à trouver des résultats proches pour toutes les évaluations de modèles. Nous choisissons $pin = 0.5$ et $pext = 0.1$ pour la génération des associations, de façon à obtenir des densités claires dans nos données. Dans les second (5D) et troisième (5S) jeux de données, 5 attributs exclusifs sont corrélés à chaque partitionnement, définissant à eux seuls l'existence des associations. Pour chaque attribut A_i corrélé à une variable cluster C_j, nous définissons la distribution de probabilités $P(A_i|C_j) = [0.8, 0.2]$ ou $[0.2, 0.8]$. Dans le second jeu de données, nous générons des associations denses avec $pin = 0.5$ et $pext = 0.1$, alors que dans le troisième jeu de données, nous explorons un contexte creux, avec $pin = 0.05$ et $pext = 0.01$.

Nous évaluons les modèles par une validation croisée à 5 itérations, par l'utilisation de différentes mesures de qualité. Tout d'abord, les fonctions de partition apprises sont comparées aux réelles en utilisant le score d'information mutuelle normalisée (NMI). Ensuite, la log-vraisemblance des associations à partir du jeu de test, étant donnée l'appartenance inférée des individus aux clusters, est calculée. Les temps d'exécution sont en outre enregistrés pour une exécution sur une machine possédant un processeur Intel Core i7-3630QM cadencé à 2.4 GHz, ainsi que 16 Go de RAM et un disque dur ayant une vitesse de rotation de 5400 tours par minute.

Les résultats moyens et les écarts types pour NMI, la log-vraisemblance des associations de test (LL) sont donnés dans les tables 6.1 et 6.2 de même que la moyenne des temps d'exécution (T). Les 4 colonnes NMI correspondent aux scores pour chaque fonction de partition FP1 et FP2 pour chaque type d'association binaire A1 et A2, alors que les deux colonnes LL correspondent à chaque type d'association (LL est évalué par individu, chacun ayant une valeur pour ses deux variables cluster). Nous pouvons voir que l'apprentissage des MRP-IC atteint toujours de meilleures évaluations de log-vraisemblance sur les jeux de test que l'apprentissage des MRP-IR. En comparant tous les ensembles de 5 mesures pour MRP-IC avec Louvain et MRP-IR avec produit cartésien, ils diffèrent toujours significativement comme le montre un test de Wilcoxon, où nous obtenons toujours W = 0 et Z = -2.0226, pour une p-valeur en dessous de 0.05. Toutefois, nous pouvons observer que la qualité du clustering, comme calculée avec NMI, est souvent faible pour les deux algorithmes, ou a des moyennes et variances élevées. Aussi, le lien entre clustering et qualité de prédiction n'est pas facilement lisible à travers ces figures.

En regardant de plus près les fonctions de partition elles-mêmes, il apparaît que les MRP-IR utilisent généralement un seul attribut de fonction de partition, et non nécessairement ceux qui ont été utilisés pour la génération de données, convergeant vers des optima locaux à cause des corrélations non souhaitées entre certaines paires de

variables : étant données deux variables cluster dans l'association, avec 0 attribut de fonction de partition et 1 cluster chacun à l'initialisation, une première opération plausible est l'ajout d'un attribut de fonction de partition à l'une des deux variables cluster. À cette étape, la meilleure variable, a_1, est celle qui minimise l'entropie pour l'instance d'apprentissage, puisque les variables n'ont pas encore de parents dans le graphe. La prochaine étape est d'ajouter une variable cluster comme parent de l'autre, puisqu'une variable est encore constante. Finalement, la troisième étape est de trouver un attribut de fonction de partition pour la seconde variable, qui sera probablement celui avec la plus forte dépendance envers a_1. Ce processus ne garantit pas d'obtenir un bon modèle et les dépendances entre variables dans le type d'association ne sont pas adéquates pour la prédiction de liens et ne permettent pas de refléter correctement les données. Notez que l'algorithme pourrait continuer d'ajouter et de supprimer des attributs de fonction de partition, mais cela requiert d'explorer de nombreuses combinaisons, puisque l'exploration se fait à l'aveugle sans prendre en considération les données des associations. De plus, l'utilisation d'un a priori pour les fonctions de partition pénalise et donc limite significativement l'ajout de plusieurs attributs dans le contexte d'un clustering par produit cartésien, puisque le nombre de clusters est doublé pour chaque nouvel attribut. Contrairement aux MRP-IR, l'utilisation d'algorithmes de co-clustering pour l'apprentissage des MRP-IC permet de considérer les données avant d'explorer l'espace des attributs de fonctions de partition et de trouver des dépendances plus précises entre les variables du type d'association, puisque le co-clustering est fait pour cette tâche. En pratique, pour cette expérience, la plupart du temps aucun attribut n'est nécessaire en plus des résultats de l'algorithme de co-clustering.

Nous pouvons finalement observer que même si les log-vraisemblances des MRP-IC sont significativement en dessous des résultats obtenus avec les MRP-IR, il est toujours possible de les améliorer pour deux jeux de données sur trois, comme l'indiquent les résultats pour MRP-IC Opt. Cela semble indiquer que l'usage de Louvain sur un graphe biparti ne permet pas de découvrir les meilleurs clusters. Ce n'est pas très surprenant puisque Louvain a originalement été proposé pour des graphes quelconques, qui n'ont pas les mêmes contraintes. Le choix d'algorithmes de clustering plus précis dans le contexte biparti est une perspective pour améliorer l'apprentissage des MRP-IC.

Considérant les temps d'exécution, nous pouvons voir que l'apprentissage des MRP-IR est toujours plus long que pour les MRP-IC. En regardant plus en détail les étapes d'apprentissage des deux modèles, nous pouvons observer que le nombre de voisinages de fonction de partition explorés avec les MRP-IR est bien plus important (plus de 1500 pour chaque modèle) que pour les MRP-IC (moins de 100 par modèle, parfois 0). C'est la conséquence de notre contrainte sur l'opération d'ajout d'attributs uniquement si la variable cluster correspondante en est dépendante. Puisque le coût d'une mise à jour de fonction de partition est plus important que pour une mise à jour du graphe, cela rend l'apprentissage des MRP-IR plus long.

Dans la prochaine section, nous réalisons des expériences sur des jeux de données réels.

	Algo.	Movie x Genres	Movie x Users
Q2	MRP-IR	0.023 ± 0	0.022 ± 0
	MRP-IC	0.299 ± 0.023	0.137 ± 0.057
LL $\times 10^{-4}$	MRP-IR	-3.77 ± 0.824	-698.7 ± 4.95
	MRP-IC	-3.59 ± 0.696	-692.4 ± 5.19
T (s.)	MRP-IR	18 991	
	MRP-IC	5 577	

TABLE 6.3 – Résultats d'évaluation de structures de MRP-IR et MRP-IC apprises sur des données IMDB/MovieLens.

	Algo.	Movie x Users low	medium	high
Q2	MRP-IR	0.0162 ± 0	0.021 ± 0	0.026 ± 0
	MRP-IC	$0.148 \pm .016$	$0.105 \pm .058$	0.177 ± 0.018
LL $\times 10^{-4}$	MRP-IR	-112.8 ± 2.45	-180.6 ± 5.83	-406.9 ± 3.81
	MRP-IC	$-109.9e6 \pm 2.09$	-178.38 ± 4.87	-397.6 ± 10.6
T (s.)	MRP-IR	67 800		
	MRP-IC	19 623		

TABLE 6.4 – Résultats d'évaluation de structures de MRP-IR et MRP-IC apprises sur des données IMDB/MovieLens divisées en plusieurs types d'associations.

6.8 Expériences sur données réelles : IMDB/MovieLens

Nous considérons également un jeu de données réel multi relationnel, fusionnant des données des bases IMDB[1] et MovieLens[2]. Plus précisément, nous considérons 3 types d'entités : *movies* (7 967 entités), *genres* (24) et *users* (6 040). Dans la première configuration, nous apprenons à la fois la structure de MRP-IR et MRP-IC pour ces 3 types d'entités et 2 types d'association : *movie_genres* (18 537 associations), décrivant les multiples genres de chaque film, et *movie_users* (813 545) représentant les films qu'un utilisateur a notés et donc vus. Dans une seconde configuration, nous divisons le type movie_users en plusieurs types d'association pour différents intervalles de notes : les notes 1-2 (resp. 3, 4-5) sont déplacées dans le type d'association *movie_users_low* (resp. *movie_users_medium*, *movie_users_high*) contenant 134 555 (resp. 213 392, 465 598) individus.

Dans les expériences précédentes, nous utilisions l'algorithme Louvain [16, 26] sur différents graphes, incluant similarités entre individus d'un même mode et associations entre modes, et avions constaté les limites d'une telle approche, tant sur le plan de la prise en compte simultanée des données des associations et des attributs, que sur le côté sous-optimal de l'algorithme pour les données bimodales. Dans ce second contexte d'expériences, nous décidons de changer de stratégie et d'utiliser une approche bipartie pour le co-clustering, appelée LP&BRIM [121]. Nous gérons l'ajout d'information des attributs par une méthode de consensus entre les clusterings obtenus à partir du graphe biparti et ceux obtenus avec chaque matrice de similarité (obtenue par distance de Hamming) uniquement, pour chaque ensemble d'entités impliqué. Nous évaluons les modèles par une validation croisée à 5 itérations où les jeux de test contiennent $1/5^{me}$ de chaque ensemble d'entités et leurs associations.

[1] http://www.imdb.com/interfaces
[2] http://grouplens.org/datasets/movielens/

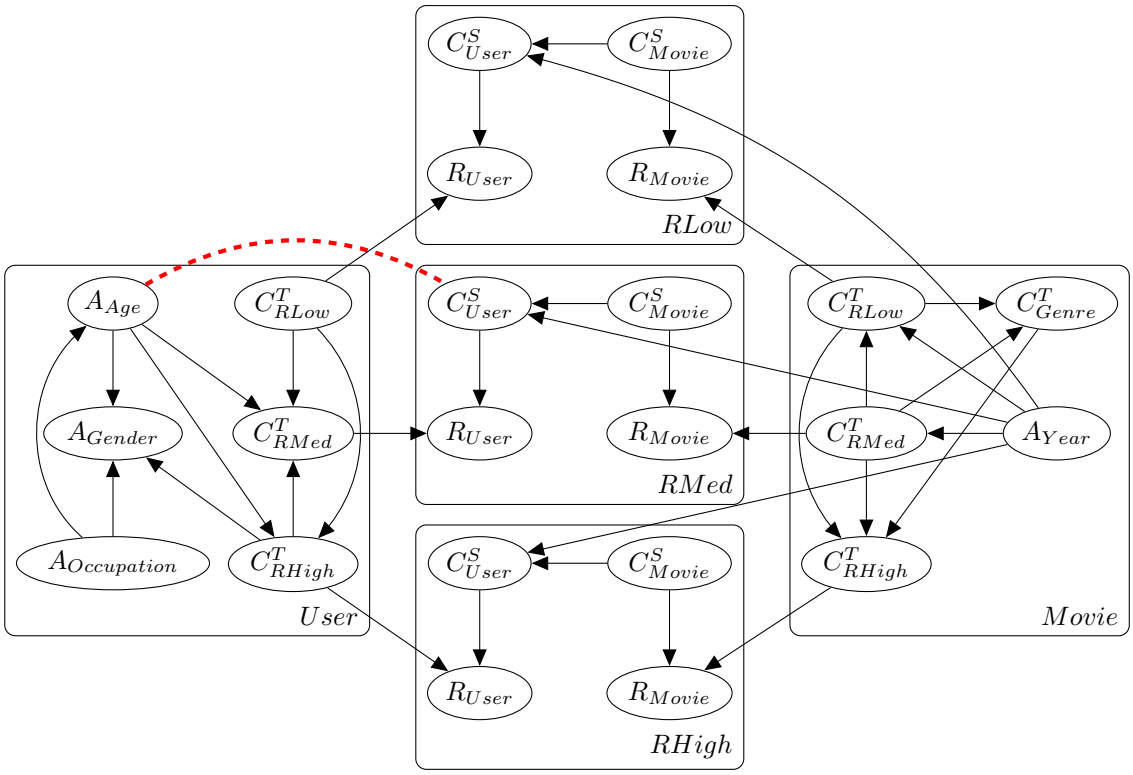


FIGURE 6.5 – Exemple de MRP-IC appris dans le second contexte de données IMDB/-MovieLens étendu

Les résultats moyens et les écarts-types pour la bimodularité à maximiser (Q2) [134] (une mesure d'évaluation de clustering biparti interne), la log-vraisemblance de chaque association de test considérée (LL), et les temps d'exécution sont données dans les tables 6.3 et 6.4. De plus, un exemple de MRP-IC obtenu dans le second contexte de données est donné Figure 6.5. Encore une fois, nous pouvons voir des résultats de LL favorables aux MRP-IC, toujours de façon significative comme le montre un résultat de test de Wilcoxon de W = 0 et Z = -2.0226, pour une p-valeur en dessous de 0.05. Le second contexte est particulièrement intéressant puisque les MRP-IC appris contiennent des dépendances entre les variables cluster cible pour chaque type d'association dérivé de *movie_users* (cf. Figure 6.5). Ceci est particulièrement intéressant puisque ce type de dépendances entre les partitionnements de différents types d'associations ne peut être trouvé dans les modèles MRP-IR.

Nous pouvons également observer que les scores de bimodularité sont significativement (Wilcoxon) meilleurs pour les MRP-IC, rendant les méthodes de co-clustering plus adaptées à l'apprentissage d'incertitudes de référence. Toutefois, les MRP-IC n'ont pas l'air de contenir beaucoup d'attributs de fonctions de partition, suggérant que les attributs des entités ne sont pas très importants pour la prédiction d'associations dans ce contexte. Quelques exemples d'attributs de fonctions de partition incluent l'âge des utilisateurs et l'année des films dans le type d'association *movie_users_medium*.

Finalement, les temps d'exécution montrent des résultats proches de ceux obtenus sur les jeux de données synthétiques.

6.9 Discussion

La qualité de l'apprentissage des MRP-IC dépend de la qualité des algorithmes de clustering utilisés. La difficulté est de trouver un algorithme capable de gérer à la fois les associations k-parties et l'augmentation d'attributs. Dans les expériences réalisées pour les MRP-IC, nous avons envisagé deux approches. Nous avons tout d'abord construit un graphe pour chaque instanciation d'un type d'association, puis ajouté des arcs entre les individus d'un même mode dont les poids dépendaient des similarités de ces individus relativement aux attributs de fonction de partition. Nous avons ensuite exécuté un algorithme de détection de communautés dans les graphes, appelé Louvain, sur ce graphe augmenté. Nous pouvons voir plusieurs limites à cette approche. D'abord, puisque les informations d'attributs et d'association sont fusionnées dans la même structure de données, les topologies des deux informations le sont également. Par conséquent, dans le cas où l'information des associations est suffisante pour trouver les bons clusters, fusionner cette information avec une autre topologie peut changer les résultats de partitionnement de façon défavorable. Ensuite, l'usage de cet algorithme fonctionnant à l'origine sur un graphe quelconque pour des données k-parties nécessite que les densités de chaque mode coïncident, notamment en ce qui concerne leurs régularisations par attributs, sous peine de ne pas trouver de densité globalement élevée permettant d'extraire le co-cluster. Enfin, une structure de données unique lie toutes les fonctions de partition ensemble et requiert le calcul de toutes ces fonctions de partition si une seule régularisation pour un mode change.

Dans la seconde série d'expériences, nous avons considéré une approche différente, où seul le graphe support d'un type d'association est utilisé pour réaliser un premier partitionnement, utilisant un algorithme dédié aux données biparties, LP&BRIM. Une régularisation a alors été réalisée pour un mode en exécutant un algorithme de recherche de consensus entre le partitionnement relationnel, pour le mode considéré, et un clustering utilisant uniquement les attributs de fonctions de partition. Ceci a permis d'obtenir un clustering indépendant pour chaque mode, une fois l'apprentissage relationnel réalisé, permettant de mettre à jour chaque fonction de partition pour un mode sans impacter les autres, tout en profitant de bons résultats grâce à l'approche bipartie dédiée.

6.10 Conclusion

Dans ce chapitre, nous avons proposé un second modèle plus étendu que les MRP-IR+ du chapitre 5 dans le but de pallier les limites des fonctions de partition et de la structure graphique des MRP-IR évoqués dans le chapitre 3. Ce modèle, appelé MRP avec incertitude de références, relâche de la même façon les contraintes des MRP-IR pour les fonctions de partition. De plus, l'ajout de variables aléatoires permet de pallier l'absence de déterminisme d'appartenance à un cluster en fonction de la valeur des attributs de fonction de partition, en permettant l'inférence d'appartenance de groupes pour les nouveaux individus. Cette extension a été étudiée plus en détail, tant sur le plan de l'inférence que sur celui de l'apprentissage conjoint des fonctions de partition et de la structure graphique, et nous avons fini par réaliser des expériences où l'apprentissage de structure des MRP-IC était comparé à celui des MRP-IR sur des données synthétiques, puis sur des données réelles. Ces résultats ont montré que la vraisemblance des associations inconnues était plus grande après apprentissage de MRP-IC qu'après apprentissage de MRP-IR, à la fois sur les données synthétiques et

réelles. De plus, pour le cas des données réelles, les MRP-IC obtenus ont permis de découvrir des connaissances impossibles à trouver pour les MRP-IR, sous la forme de dépendances probabilistes entre les variables de clustering concernant des types d'association différents.

Comme pour le chapitre précédent, nous avons vu durant tout le chapitre l'importance de choisir des bons algorithmes de clustering pour obtenir des résultats plus ou moins représentatifs. Nous avons notamment changé de stratégie entre les différentes expériences réalisées, en fonction des problèmes que nous rencontrions après analyse des résultats. Nous avions ainsi vu dans le chapitre précédent que les approches NMF possédaient de nombreuses extensions pour gérer plusieurs types de données, proposant même une façon native d'introduire une régularisation par attributs grâce aux régularisations laplaciennes. Toutefois, lors de leur usage, nous avions observé des temps d'exécutions très longs et des problèmes de convergence en fonction de l'initialisation pouvant mener jusqu'à l'arrêt de l'algorithme lorsque des matrices devenaient singulières.

Nous avons fait des choix différents dans ce chapitre, et utilisé pour commencer une approche plus rapide de détection de communautés dans les graphes, Louvain [16, 26], mais nous avons dû pour cela construire un graphe contenant à la fois l'information bipartie pour les relations binaires des expériences réalisées, et une information de régularisation, menant à des performances hasardeuses à cause des topologies non nécessairement compatibles de toutes ces différentes informations.

Nous avons ensuite exploré une méthode rapide de détection de communautés dans les graphes bipartis, LP&BRIM [121], nous permettant de découvrir des fonctions de partition plus fiables à l'égard des associations, mais ne proposant pas de possibilité d'ajout de liens entre individus d'un même mode. Nous avons alors considéré la régularisation par attributs par le biais d'un consensus entre plusieurs fonctions de partition, chacune apprise soit sur le graphe support du type d'association instancié, soit sur les seules informations de similarité entre individus d'un mode. Cette dernière approche a montré de bons résultats et permet un découplage des fonctions de partition une fois celles-ci initialisées. Aussi, cette approche mériterait d'être détaillée davantage.

Il est important de noter que de nombreuses autres options peuvent encore être explorées, certaines étant énoncées dans le chapitre 4 mais la littérature autour du clustering est bien plus foisonnante [1].

7

PILGRIM-Relational : une plateforme logicielle pour les modèles relationnels probabilistes

Sommaire

7.1 Introduction

Malgré les travaux existant sur les MRP et leurs extensions, il est intéressant de voir que les outils utilisés pour réaliser les expériences ne sont généralement pas distribués. Plusieurs outils sont apparus plus tard et ont été rendus disponibles, mais ils sont très limités et n'implémentent qu'une partie très restreinte de ce qui est détaillé dans la littérature. À titre d'exemple, tous les outils trouvés supportant les MRP ne se focalisent que sur les MRP orientés attributs, et tous se focalisent principalement sur l'inférence à partir de modèles prédéfinis. Des fonctionnalités d'apprentissage sont parfois présentes, mais uniquement sur les seuls paramètres.

Afin de pouvoir réaliser nos propres expériences, nous avons dû implémenter notre propre ensemble d'outils pour la manipulation des MRP et des extensions d'intérêt, l'apprentissage de paramètres et de structure, ainsi que l'inférence. Afin de pouvoir travailler en groupe sur ces modèles et éviter de refaire chacun de notre côté les mêmes fonctionnalités, nous avons décidé de développer quelque chose de réutilisable et extensible, sous la forme d'une bibliothèque logicielle, appelée PILGRIM-Relational. Cette bibliothèque est écrite en C++ pour pouvoir aisément s'appuyer sur une autre bibliothèque mature permettant la manipulation de réseaux bayésiens, appelée *ProBT*[1], permettant de réutiliser certaines structures de base et déléguer une grosse partie du travail d'inférence après génération d'un RBP.

Dans ce chapitre, nous commençons par inventorier l'ensemble des outils existant à notre connaissance pour la programmation probabiliste et étudions les différentes fonctionnalités offertes par chacun pour exhiber les manques liés à nos besoins. Nous présentons ensuite l'architecture principale de PILGRIM-Relational et précisons les contributions effectuées durant cette thèse. Nous détaillons ensuite l'ensemble des grands composants principaux de la bibliothèque, à savoir : les fonctions de création de schémas relationnels, les classes façades d'instances, les structures permettant de manipuler des variables aléatoires liftées ou non, les différents modèles graphiques effectivement disponibles et plus en détail les modèles avec incertitude de référence, le concept d'algorithme de clustering et le lien avec celui de fonction de partition dans PILGRIM-Relational, et enfin les fonctionnalités permettant de réaliser de l'inférence sur une instance donnée.

7.2 Outils logiciels pour les MRP

De nombreux outils logiciels ont été développés pour réaliser ce qui est appelé de la *programmation probabiliste* impliquant la définition, l'inférence et éventuellement l'apprentissage de modèles probabilistes. Nous avons identifié 25 outils existants, que nous listons dans la Figure 7.1 avec certaines de leurs propriétés, incluant le type de modèles de la littérature supportés, leurs fonctionnalités parmi l'inférence, l'apprentissage de paramètres et de structure, et enfin la disponibilité du code source.

Nous pouvons voir que l'essentiel des outils disponibles permet la définition de modèles à la main, peu importe leur type, ainsi que l'inférence dans ces modèles. Un grand sous-ensemble de ces outils permet également l'apprentissage de paramètres

[1] http://www.probayes.com/fr/Bayesian-Programming-Book/downloads/

Outil	URL	Modèles	Inférence	Apprentissage		Source
				Paramètres	Structure	
aGrUM	https://forge.lip6.fr/projects/aGrUM/wiki	**RBOO, MRP**	**oui**	non	non	non
Alchemy	http://alchemy.cs.washington.edu/	MLN	**oui**	**oui**	**oui**	**oui**
Alch. Lite	http://alchemy.cs.washington.edu/lite/	Tractable MLN	**oui**	non	non	**oui**
Anglican	http://www.robots.ox.ac.uk/~fwood/anglican/index.html	Ad-hoc	**oui**	non	non	**oui**
BayesiaLab	http://www.bayesia.com/fr/produits/bayesia-engines.php	BN	**oui**	**oui**	non	non
BLOG	https://github.com/BayesianLogic/blog	BLOG	**oui**	non	non	**oui**
BUGS	http://www.mrc-bsu.cam.ac.uk/software/bugs/	Ad-hoc	**oui**	non	non	**oui**
Church	http://projects.csail.mit.edu/church/wiki/Church	Ad-hoc	**oui**	**oui**	non	**oui**
Dimple	http://dimple.probprog.org/	MGP	**oui**	**oui**	non	**oui**
Figaro	https://www.cra.com/work/case-studies/figaro	**Relationnels**	**oui**	**oui**	non	**oui**
FACTORIE	http://factorie.cs.umass.edu/	**Relationnels**	**oui**	**oui**	non	**oui**
Infer.NET	http://research.microsoft.com/en-us/um/cambridge/projects/infernet/	MGP	**oui**	**oui**	non	non
Primula	http://people.cs.aau.dk/~jaeger/Primula/download.htm	RBN	**oui**	**oui**	non	**oui**
ProbLog	https://dtai.cs.kuleuven.be/problog/	BLP	**oui**	**oui**	non	**oui**
ProbReM	http://cs.mcgill.ca/~fkaeli/probrem/about.html	DAPER	**oui**	**oui**	non	**oui**
PyMC	https://github.com/pymc-devs/pymc	MGP	**oui**	**oui**	non	**oui**
ProbCog	http://ias.cs.tum.edu/software/probcog	MLN, BLP	**oui**	**oui**	non	**oui**
ProBT	http://old.probayes.com/index.php/en/produits/sdk/probt	BN	**oui**	**oui**	**oui**	non
Proximity	https://kdl.cs.umass.edu/display/public/Proximity	RBC, RPT, RDN	**oui**	**oui**	**oui**	**oui**
R2	http://research.microsoft.com/en-us/projects/r2/	Ad-hoc	**oui**	**oui**	non	non
Stan	http://mc-stan.org/	MGP	**oui**	non	non	**oui**
Tuffy	http://i.stanford.edu/hazy/tuffy/home	MLN	**oui**	**oui**	non	**oui**
UnBBayes	http://unbbayes.sourceforge.net/	**RBOO, MRP**	**oui**	non	non	**oui**
Venture	http://probcomp.csail.mit.edu/venture/	**Relationnels**	**oui**	non	non	non
WebPPL	http://webppl.org/	MGP	**oui**	**oui**	non	**oui**

FIGURE 7.1 – Résumé des différents outils de programmation probabiliste disponibles avec différentes propriétés identifiant s'ils permettent de réaliser de l'inférence à partir d'un modèle, d'apprendre les paramètres et la structure de ces derniers, et si le code source est disponible. De plus, les types de modèles supportés sont donnés par des abréviations : *RBOO = Réseaux bayésiens Orientés Objets* [106, 7] ; *MRP = Modèles Relationnels Probabilistes* [62] ; *MLN = Markov Logic Network* [160] ; *BLOG = Bayesian LOGic* [131] ; *MGP = Modèles Graphiques Probabilistes* (sous-entendu non relationnels) ; *RBN = Relational Bayesian Network (sensu Jaeger)* [90] ; *BLP = Bayesian Logic Program* ; *RBC = Relational Bayesian Classifiers* [141] ; *RPT = Relational Probability Tree* [140] ; *RDN = Relational Dependency Network* [139] ; *BN = Bayesian Network* [152]. Les propriétés appropriées pour nos besoins sont identifiées en gras et la bibliothèque ProBT utilisée dans PILGRIM-Relational est identifiée en italique.

pour les modèles considérés. Toutefois, de très rares outils proposent également des algorithmes d'apprentissage de structure. Parmi ces derniers, nous pouvons citer :

- *Alchemy*[2] [102] : permet la définition, l'inférence et l'apprentissage de réseaux logiques de Markov (MLN) [160]. Il est important de noter qu'une version *Lite* de cet outil permet de travailler sur des réseaux logiques de Markov tractables [54] mais n'offre que des fonctionnalités d'inférence ;

- *Proximity*[3] : permet de travailler avec différents modèles relationnels comme les classifieurs relationnels bayésiens [141], les arbres de probabilités relationnels [140] et les réseaux de dépendance relationnels [139].

Ces outils ne permettent pas de travailler sur des MRP. Même s'il est possible de convertir en théorie un MRP en MLN, le contraire n'est pas nécessairement vrai. De ce fait, la perte de la direction des arcs dans les modèles appris peut poser problème lors de la tentative de reconversion d'un MLN en MRP.

Si nous nous intéressons maintenant aux librairies qui permettent de travailler avec des MRP, nous pouvons citer :

- *aGrUM*[4] : permet de définir et réaliser de l'inférence dans différents types de modèles propositionnels ou relationnels (OOBN et MRP). Toutefois, les fonctionnalités de cet outil se focalisent essentiellement sur l'inférence ;

- *Figaro*[5] et *FACTORIE*[6] [128] : outils assez semblables dans l'approche, écrits en SCALA, permettant une programmation modulaire des modèles relationnels et offrant des fonctionnalités d'inférence et d'apprentissage de paramètres pour des modèles dirigés ou non dirigés. Toutefois, cet outil ne propose pas d'algorithme d'apprentissage de structure ;

- *UnbBayes*[7] [28] : un outil populaire écrit en JAVA permettant de définir de nombreux types de modèles et offrant des fonctionnalités d'inférence et d'apprentissage de paramètres et de structure. Toutefois, aucun algorithme d'apprentissage n'est disponible pour le cas des MRP, alors que l'on peut pourtant en définir dans cet outil ;

- *Venture*[8] : outil simple permettant de définir et de réaliser de l'inférence dans des modèles probabilistes relationnels. Aucune fonction d'apprentissage n'est toutefois proposée à notre connaissance et le code source ne semble pas être diffusé.

Nous remarquons qu'aucun de ces outils ne propose d'algorithme d'apprentissage de structure, voire d'estimation de paramètres, ni ne propose d'interface permettant de créer de nouveaux algorithmes d'apprentissage de structure sans toucher au cœur du code.

[2] http://alchemy.cs.washington.edu/
[3] http://kdl.cs.umass.edu/proximity/
[4] https://forge.lip6.fr/projects/agrum
[5] https://www.cra.com/work/case-studies/figaro
[6] http://factorie.cs.umass.edu/
[7] http://unbbayes.sourceforge.net/
[8] http://probcomp.csail.mit.edu/venture/

Le constat des limitations des outils existant nous a amenés à envisager la création de notre propre plateforme logicielle permettant de définir des MRP et de réaliser de l'inférence et de l'apprentissage de paramètres et de structure. Avoir le contrôle sur cet outil nous permet d'adopter une implémentation au plus proche des MRP définis dans la littérature, et notamment des extensions de ceux-ci, dont la simple définition est par ailleurs indisponible dans les outils existants.

7.3 Le projet PILGRIM

La plateforme PILGRIM[9] (ProbabIListic GRaphIcal Models) en cours de développement regroupe un ensemble de projets développés au sein du Laboratoire d'Informatique de Nantes Atlantique (LINA), interdépendants, extensibles, écrits en C++, ayant pour but de proposer des outils logiciels efficaces permettant de définir, réaliser de l'inférence et apprendre différents modèles graphiques probabilistes. Elle est actuellement orientée vers les modèles dirigés, en particulier les RB et les MRP ainsi que certaines des extensions de ces derniers. La plateforme est organisée en quatre sous-projets distincts :

- **PILGRIM General** : propose des fonctionnalités communes à tous les autres projets, telles que la définition et la sérialisation d'un réseau bayésien, l'accès à des jeux de données au format CSV, et certaines mesures comme la KL-divergence ;
- **PILGRIM Structure Learning** : propose des algorithmes d'apprentissage de structure pour les RB (p.ex. recherche gloutonne, MMHC) et des implémentations de fonctions de score (p.ex. BDeu, BIC), de tests statistiques (p.ex. information mutuelle), et de mesures pour évaluer la qualité d'une structure d'un RB appris par rapport à la structure connue du modèle d'origine (p.ex. distance structurelle de Hamming) ;
- **PILGRIM Relational** : propose de nombreuses fonctionnalités pour la définition, l'inférence et l'apprentissage dans un contexte de MRP et de leurs extensions. C'est ce projet que nous avons étendu et utilisé pour réaliser nos expérimentations durant cette thèse. C'est pourquoi nous nous focalisons dessus dans la suite de ce chapitre ;
- **PILGRIM Applications** : contient un ensemble d'implémentations d'algorithmes dédiés à l'application des différentes briques de PILGRIM pour résoudre des problèmes. Actuellement, ce projet propose des implémentations orientées systèmes de recommandations à base de MRP.

PILGRIM sera bientôt diffusée comme logiciel libre sous licence GPL[10].

7.4 PILGRIM-Relational

La bibliothèque PILGRIM-Relational est l'une des bibliothèques écrites en C++ du projet PILGRIM. Elle est constituée de plusieurs composants interconnectés les uns aux autres dont le diagramme de dépendances simplifié est donné dans la Figure 7.2 :

[9] http://www.pilgrim.univ-nantes.fr/
[10] http://www.gnu.org/licenses/gpl-3.0.en.html

- *Schéma* contient l'ensemble des classes liées à la définition d'un schéma relationnel. Ce module est indépendant et fournit des services à la majorité des autres modules ;
- *Variables* contient l'ensemble des classes permettant de définir et de manipuler des variables aléatoires de MRP, des collections de ces types ainsi que des valeurs particulières de telles collections. Ce composant nécessite des services du composant *Schéma*, car chaque variable est associée à un schéma de relation et repose généralement sur des informations d'un attribut descriptif ou d'une variable de référence du schéma relationnel ;
- *Distributions* contient l'ensemble des classes permettant de définir des distributions de probabilités sur les variables aléatoires susmentionnées. Par conséquent, ce composant nécessite d'utiliser des services fournis par le composant *Variables* ;
- *Instance* contient les classes façade permettant d'interroger et manipuler des jeux de données. Une instance dépend toujours d'un schéma relationnel et les requêtes effectuées peuvent porter sur n'importe quelle variable aléatoire d'un MRP. De ce fait, ce composant dépend de *Schéma* et *Variables* ;
- *Modèle* contient les classes permettant de définir des MRP et s'appuie par définition sur les composants *Distributions*, *Variables* et *Schéma*. De plus, comme les classes de modèle contiennent leur propre méthode d'estimation des paramètres, ce composant nécessite également les services du composant *Instance* ;
- *Apprentissage* contient les classes liées à l'apprentissage de structure des MRP et est donc client des services des composants *Modèle* et *Instance* ;
- *Clustering* contient les classes permettant de définir les concepts d'algorithme de clustering et de fonction de partition, ainsi que d'en fournir diverses implémentations. Ce composant est donc dépendant des mêmes modules que *Modèle*, excepté pour les types de distributions ;
- *Modèle IR/IC* : ce composant peut être vu comme une sous-partie du module *Modèle* et en hérite toutes les dépendances. Il représente l'ensemble des classes de modèles liées au paradigme d'incertitude de référence. De ce fait, il dépend en plus du composant *Clustering* ;
- *Apprentissage IR/IC* étend le module *Apprentissage* pour les MRP-IR et MRP-IC et est donc dépendant de *Modèle IR/IC* sans toutefois nécessiter une dépendance directe avec *Modèle*.

Puisque l'inférence dans les modèles est théoriquement possible par la génération d'un RBP, qui n'est autre qu'un réseau bayésien classique, les fonctionnalités d'inférence se résument à la génération des RBP à partir d'un MRP et d'une instance, utilisant des structures de données définies dans la bibliothèque *ProBT* dédiée aux réseaux bayésiens non relationnels. Ainsi, l'interrogation du RBP pour effectivement réaliser l'inférence est faite via des fonctions définies dans cette bibliothèque tierce.

Il est important de noter que PILGRIM-Relational inclut également un module pour réaliser des expériences, lié à tous les autres composants. Ce module est détaillé brièvement dans la section 7.14.

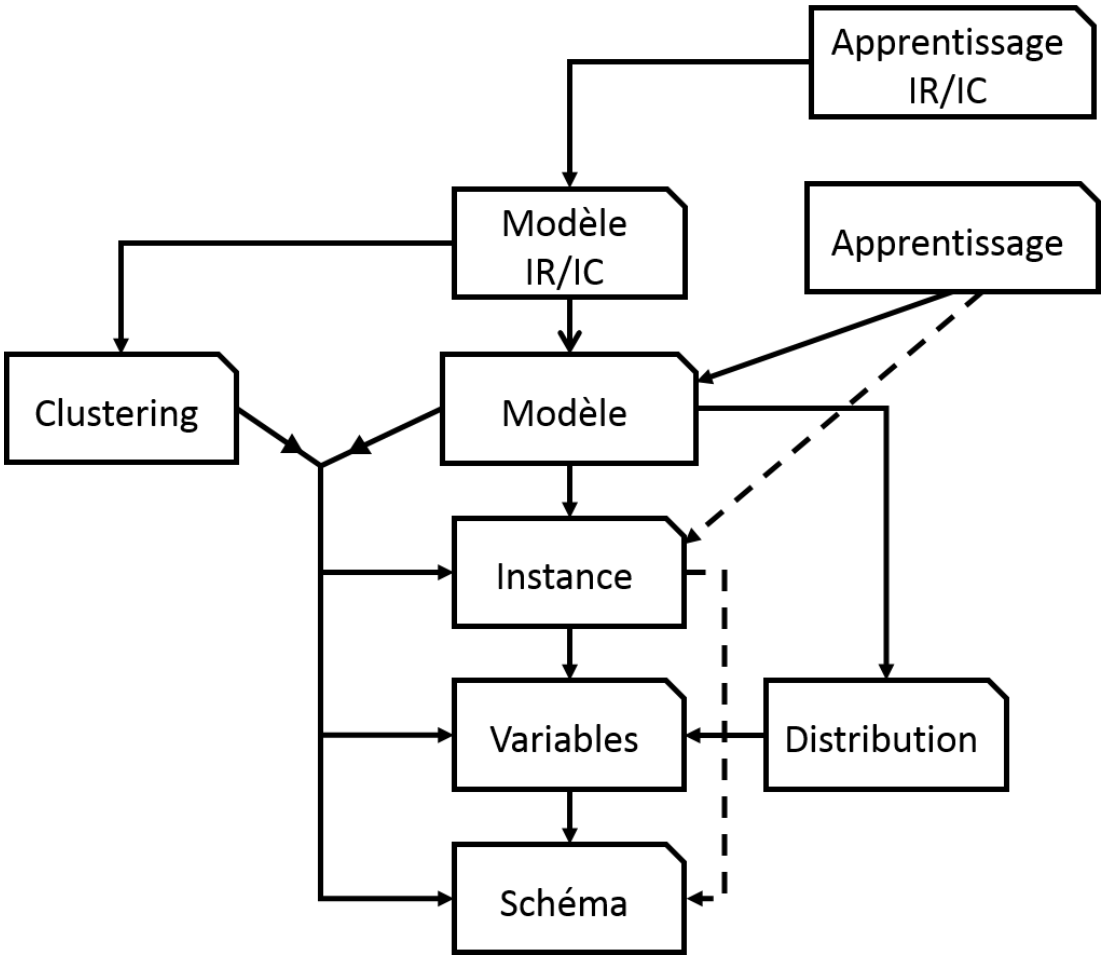


FIGURE 7.2 – Schéma de dépendances entre les différents composants de PILGRIM-Relational. Une flèche indique une dépendance entre composants. Les styles des arcs n'ont pas de sémantique particulière, mais servent uniquement à identifier les différents arcs se chevauchant.

7.5 Contributions

La bibliothèque PILGRIM-Relational est actuellement développée au sein du Laboratoire d'Informatique de Nantes Atlantique et est le fruit du travail de plusieurs acteurs. Notre rôle particulier durant cette thèse sur ce projet inclut :

- la définition de l'architecture générale de la bibliothèque et des différents types permettant de définir les schémas relationnels et les modèles (la majorité des composants *Modèle*, *Modèle IR/IC* et *Schéma*) ;
- la participation à l'écriture des algorithmes d'estimation des paramètres ;
- la définition des interfaces du composant d'instance ainsi que la majorité des implémentations des services d'interrogation des bases de données relationnelles, incluant l'optimisation itérative des requêtes SQL (la majorité du composant *Instance*) ;
- la définition des interfaces du composant lié au clustering et l'implémentation de divers algorithmes de clustering et d'interfaçage entre algorithmes de clustering et MRP-IR/IC (l'intégralité du module *Modèle IR/IC* ;
- l'implémentation de fonctionnalités de sérialisation des modèles ;
- l'implémentation d'algorithmes d'apprentissage de structure des MRP-IR/IC (tout le composant *Apprentissage IR/IC*) mais avec une intervention faible sur l'apprentissage de structure des MRP classiques ;
- diverses factorisations et remaniements de code au fur et à mesure pour maintenir une structure de classes cohérente et permettre la relation de hiérarchie entre MRP et MRP-IR/IC.
- l'intégralité du module d'expérimentations.

Les autres fonctionnalités ont principalement été réalisées par Mouna Ben Ishak [11] et Rajani Chulyadyo, durant leur doctorat dans l'équipe de recherche.

Dans la suite, nous présentons les différentes parties de la bibliothèque.

7.6 Schéma relationnel

Dans PILGRIM-Relational, toute définition de modèle relationnel et d'instance nécessite de s'appuyer sur un schéma relationnel. Le diagramme de classes simplifié du composant est donné dans la Figure 7.3. Un schéma relationnel (*RelationalSchema*) contient un ensemble de schémas de relation et définit un graphe dirigé entre ceux-ci, en fonction des contraintes d'intégrité référentielles définies. Chaque contrainte de référence est un triplet (Schéma de relation source, Attribut de référence, Schéma de relation cible) où l'attribut doit nécessairement appartenir au schéma source.

À partir d'un schéma relationnel, il est possible de manipuler l'ensemble des schémas de relation (consulter, modifier), ainsi que l'ensemble des contraintes d'intégrité référentielles. Un autre ensemble de méthodes permet également de parcourir le graphe de références selon un schéma de relation de départ et une séquence de références, via

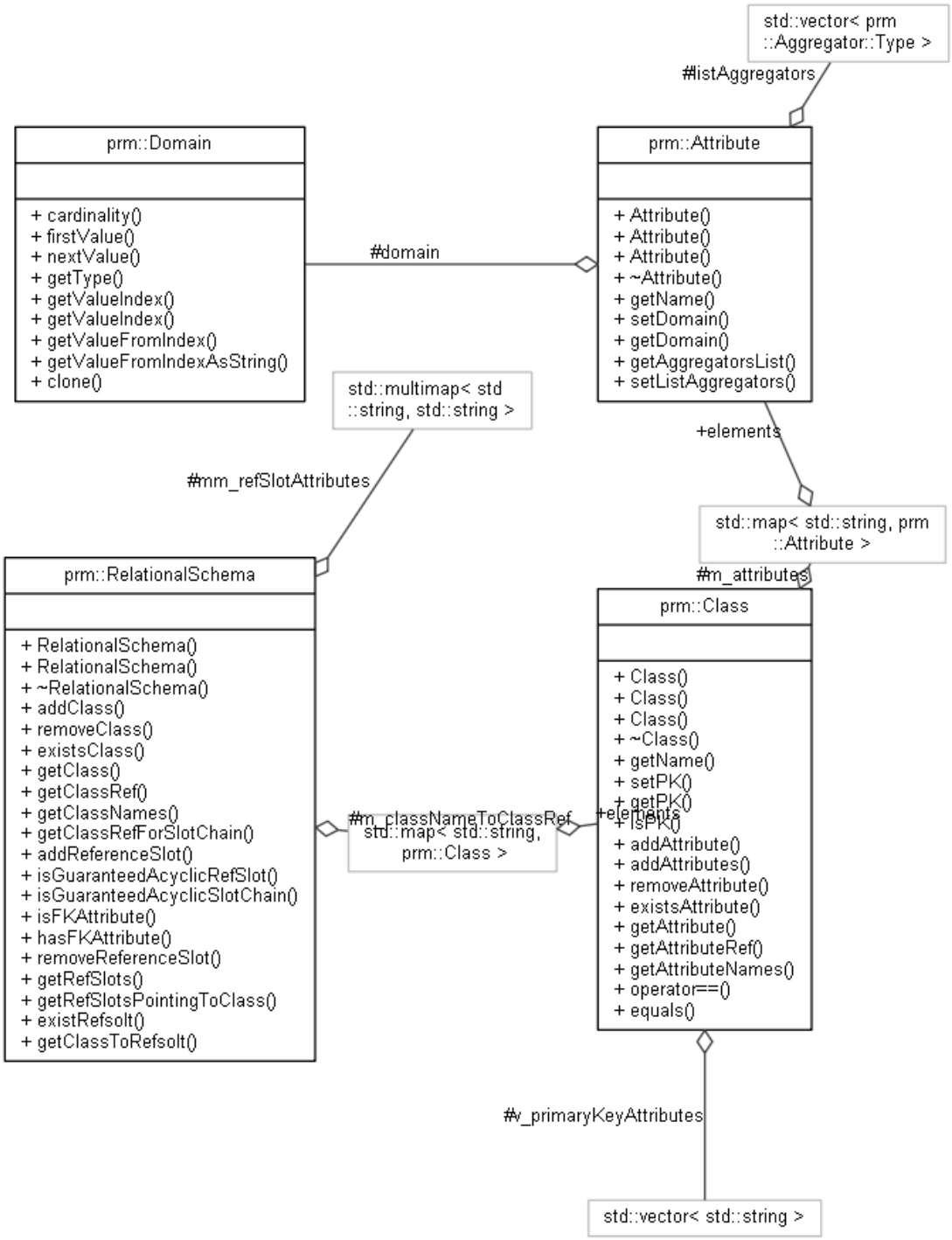


FIGURE 7.3 – Diagramme de classe simplifié du composant schéma relationnel

les méthodes *getClassRefForSlotChain()* et *isGuaranteedAcyclicSlotChain()*.

Le type *Class* représente un schéma de relation. Il contient un ensemble d'attributs et définit les contraintes internes de ce schéma. À l'heure actuelle, la seule contrainte qu'il est possible de définir est celle de clé primaire (les clés étrangères sont définies dans le schéma relationnel). La classe *Class* peut ainsi manipuler son ensemble d'attributs et l'ensemble des clés primaires.

Un attribut (*Attribute*) représente un attribut réel dans un schéma de relation. Il possède un nom, un domaine de définition, et l'ensemble d'agrégateurs qu'il est possible d'utiliser sur un ensemble de valeurs de cet attribut. Les agrégateurs possibles dépendent des propriétés de l'attribut (nominal, ordinal, discret, continu, ...).

Un domaine de définition (*Domain*) représente un ensemble fini ou non de valeurs possibles pour un attribut de schéma de relation ou pour une variable aléatoire d'un modèle probabiliste. Il contient un ensemble de valeurs ordonnées de façon à toujours pouvoir les parcourir de la même manière, ce qui est vital pour la définition de certaines distributions de probabilité. Actuellement, le composant a essentiellement été pensé pour le cas des domaines discrets. Aussi, un domaine propose toujours des méthodes permettant d'itérer sur ses valeurs, via les méthodes *firstValue()* et *nextValue()*, et une cardinalité que l'on attend finie.

La classe *Domain* est abstraite. Ces sous-classes concrètes actuellement implémentées sont : ***MultinomialDomain*** pour les ensembles finis de valeurs, ***ProBTDomain*** pour supporter tous les domaines (finis) de ProBT, ***ContainingNullDomain*** permettant de rapidement créer un domaine à partir d'un autre en ajoutant uniquement une valeur nulle en plus (utilisée pour les types agrégés, où il est possible qu'une agrégation ait un résultat indéterminé en plus des valeurs du type d'origine), et ***CompositeDomain*** permettant d'interfacer deux types existant en un seul.

7.7 Noeuds, Variables, Séquences

Dans PILGRIM-Relational, nous faisons la distinction entre attributs de base de données et variables aléatoires d'un modèle relationnel. En effet, alors que les attributs définissent les réelles informations utiles disponibles dans la base de données, les domaines de définition des variables aléatoires peuvent être différents. Par exemple, lors de la construction d'un parent distant dont la séquence de références contient une partie inversée, il est nécessaire d'agréger les valeurs des tuples atteints de cette façon. Or, une agrégation peut parfois être indéterminée, soit parce qu'il n'y a pas de tuples réellement atteints pour tuple enfant donné, soit parce qu'il y a une égalité dans l'ensemble (p.ex. le mode d'un ensemble contenant deux valeurs majoritaires n'est pas défini). De ce fait, nous ajoutons dans ce cas une valeur dans le domaine de la variable aléatoire représentant cet état indéterminé. Or, cette modification est contextuelle à une situation dans le modèle, et ne concerne pas l'attribut de façon générale.

En outre, des variables aléatoires peuvent exister dans les modèles sans s'appuyer sur aucun attribut. Par exemple, dans les MRP-IR et MRP-IC, les variables sélecteur ou cluster sont des ajouts par rapport au schéma relationnel initial. Aussi, nous avons besoin de séparer attributs du schéma relationnel et variables aléatoires.

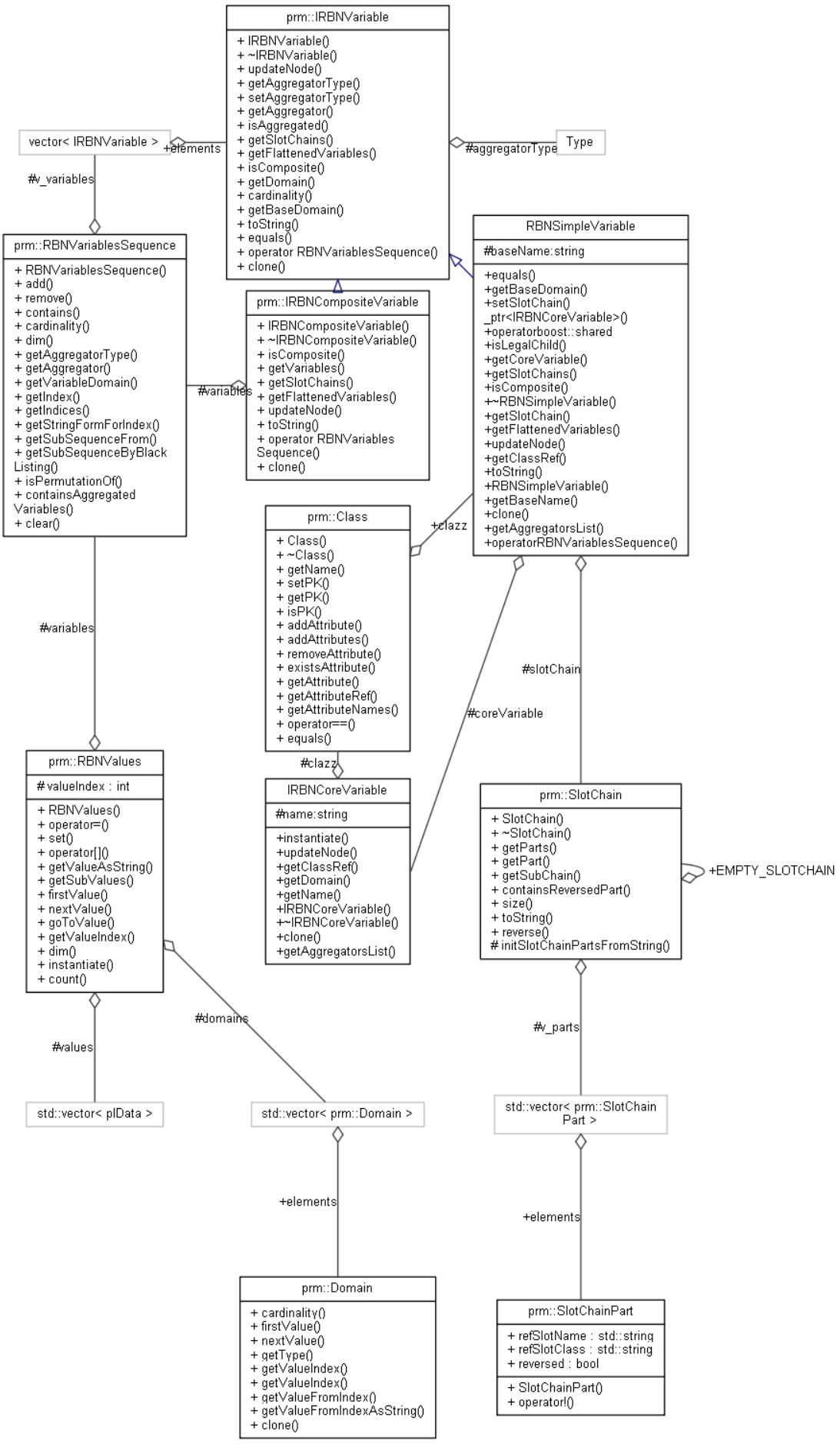


FIGURE 7.4 – Diagramme de classe simplifié des types liés aux variables aléatoires

L'ensemble des classes liées aux variables aléatoires, à leurs séquences et à leurs valeurs est donné dans la Figure 7.4.

Toute variable aléatoire dans PILGRIM-Relational est de type *IRBNVariable*. Cette classe contient un type d'agrégation, dont les valeurs possibles sont actuellement ***NO*** (aucune agrégation), ***MODE***, et ***AVERAGE*** (pour la moyenne). Un objet de cette classe possède un domaine de définition de base (sans prendre en compte l'agrégation par exemple), un domaine de définition (prenant en compte l'agrégation par exemple), une cardinalité, est potentiellement lié à plusieurs séquences de référence, et peut potentiellement être aplati en plusieurs variables simples. Cette classe est abstraite et ne contient pas de membre en tant que tel. Elle retourne toutes ses composantes via des méthodes implémentées par les classes descendantes.

Deux classes raffinent *IRBNVariable* : *RBNSimpleVariable* est le type (concret) de toutes les variables simples ; *IRBNCompositeVariable* est le type abstrait de toutes les variables composites, construites à partir de plusieurs variables selon un patron de conception *composite*. Les variables composites permettent de gérer le concept d'opérateur multi ensemble, défini dans la littérature lors de certains travaux autour des MRP [88]. Nous ne détaillerons pas ce concept davantage ici puisqu'il n'est pas utilisé dans les travaux de cette thèse.

La classe *RBNSimpleVariable* représente ainsi une variable aléatoire simple, et seules ces variables sont utilisées dans les travaux de cette thèse. Une variable simple possède en plus d'un agrégateur une séquence de références, qui peut être vide, et un schéma de relation *Class* pour lequel elle est définie. Plusieurs méthodes héritées de *IRBNVariable* supposées retourner des ensembles ne retournent que des singletons dans cette sous-classe (une seule séquence de références pour *getSlotChains()* par exemple). La fonction *isLegalChild()* vérifie si la variable simple peut être utilisée comme enfant dans une distribution de probabilités, c.-à-d. ce qui est le cas si son agrégateur est nul et sa contrainte de références vide.

Un objet de la classe *RBNSimpleVariable* définit une décoration (ajout d'une agrégation et d'une séquence de références) d'une variable de base de type *IRBNCoreVariable*. Cette dernière interface encapsule de son côté les différents types de variables pouvant être définis dans un MRP. À l'heure actuelle, plusieurs types de variables aléatoires de base sont disponibles dans la bibliothèque : ***RBNAttributeVariable*** crée une variable aléatoire à partir d'un attribut de schéma de relation ; ***RBNClassIdsVariable*** représente une variable aléatoire de référence ; ***RBNSelectorVariable*** et ***RBNTargetSelectorVariable*** représentent des variables aléatoires associées à une fonction de partition et sont utilisés pour les sélecteurs ou variables cluster source et cible dans les MRP-IR et MRP-IC.

Une séquence de références (*SlotChain*) définit une séquence de contraintes de références directes ou inversées de telle sorte que la séquence définisse un chemin possible dans le schéma relationnel. Un objet de cette classe est composé de parties (*SlotChainPart*) chacune consistant en un triplet (Schéma de relation, Référence, Booléen pour indiquer un éventuel état inversé). La classe *SlotChain* contient des méthodes pour gérer les parties et certains accesseurs haut niveau pour récupérer des sous-séquences ou vérifier si la séquence contient au moins une partie inversée (de-

mandant alors d'agréger les tuples atteints).

La classe *RBNVariablesSequences* représente une séquence de variables, chacune pouvant être simple ou composite. Organiser les variables en séquence est très important, car l'ordre permet d'interpréter les résultats obtenus pour les comptages sur les jeux de données et la façon de lire une distribution de probabilités. La classe contient des méthodes pour manipuler les différents éléments de la séquence, et certains accesseurs permettant de comparer les séquences entre elles ou récupérer des sous-séquences.

La classe *RBNValues* va de pair avec la classe *RBNVariablesSequence* et un de ses objets représente une configuration particulière des variables de la séquence à laquelle elle est associée. Pour chaque élément de la séquence de variables, un objet de valeurs contient une paire (Domaine, Valeur), au même index. La classe permet, en plus des accesseurs et modificateurs habituels, de gérer l'itération de l'ensemble des valeurs de la séquence, selon l'ordre suivant : considérant une valeur de séquence de variables, la valeur suivante est obtenue en incrémentant la valeur la plus à droite possible (celle qui n'est pas déjà à sa dernière valeur), puis toutes les valeurs restant à la droite de celle modifiée sont réinitialisées à leur valeur de départ.

À titre d'exemple, si une séquence contient 3 variables binaires. Alors, si la valeur est dans l'état 010 avant incrémentation, elle passera à 011 puis 100 aux deux incrémentations suivantes.

7.8 Instance

Le diagramme de classe simplifié des différentes classes associées aux instances est donné dans la Figure 7.5. La classe *Instance* s'appuie sur un schéma relationnel. La classe propose plusieurs méthodes pour consulter un jeu de données dont la structure de données correspond au schéma. En pratique, la classe *Instance* est abstraite et deux classes concrètes sont actuellement disponibles : ***MockInstance*** proposant des méthodes pour créer un jeu de données à la main (potentiellement long et fastidieux mais permettant de réaliser des tests sans dépendance à une base de données extérieure) et ***DBInstance*** implémentant la version pour les jeux de données stockés dans des bases de données relationnelles, s'appuyant sur *SQL*.

Les classes d'instance ont été pensées comme des classes façade, c.-à-d. rassemblant toutes les méthodes relatives aux jeux de données à un seul endroit. Les méthodes de ces classes peuvent être regroupées en plusieurs familles, énumérées ci-dessous.

La famille principale de méthodes permet de compter les occurrences de différentes configurations de valeurs de variables. Les méthodes de cette famille incluent *countAll()* permettant de compter tous les individus d'une séquence de variables particulières (donc d'une vue donnée) sans considération de valeurs précises, *countForOneState()* permettant de considérer uniquement une configuration précise de valeurs pour une séquence de variables, *countForSomeStates()* qui étend la méthode précédente à une liste de configurations de valeurs, et enfin *countForAllStates()* qui réalise un ensemble de comptages pour toutes les configurations de valeurs possibles dans l'ordre obtenu par un itérateur de type *RBNValues* sur la vue considérée. Ces méthodes sont utilisées pour calculer les paramètres et autres indicateurs statistiques lors de l'apprentissage de modèles.

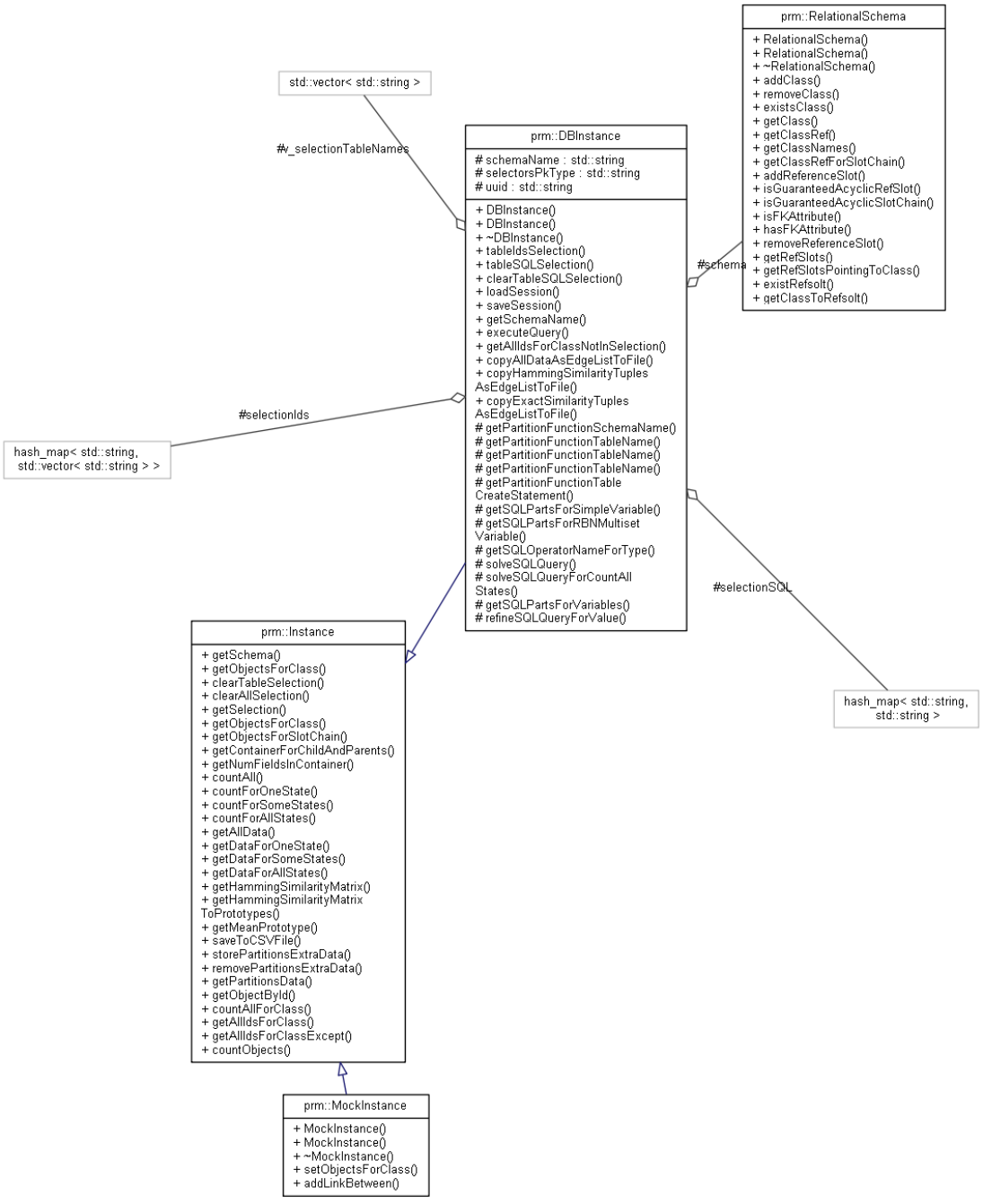


FIGURE 7.5 – Diagramme de classe simplifié du composant instance

De façon parallèle, la seconde famille de méthodes permet de récupérer des valeurs dans le jeu de données. L'ensemble des méthodes est organisé de la même façon que les comptages et contient donc *getAllData()* pour récupérer une vue complète, *getDataForOneState()* pour récupérer les tuples d'une configuration particulière, *getDataForSomeStates()* pour plusieurs configurations, et enfin *getDataForAllStates()* pour toutes les configurations. Ces méthodes sont utilisées essentiellement par les méthodes de partitionnement pour les extensions des MRP basées sur le paradigme d'incertitude de références.

Une troisième famille de méthodes permet de gérer la visibilité du jeu de données représenté par l'instance. En effet, afin de gérer jeux de données d'apprentissage et jeux de données de test sans diviser physiquement les informations dans plusieurs bases de données, nous avons permis de filtrer les informations de chaque schéma de relation au niveau des classes d'instance. Les méthodes *clearTableSelection()*, *clearAllSelection()*, *getSelection()*, mais aussi les méthodes de *DBInstance tableIdsSelection()*, *tableSQLSelection()* et *clearTableSQLSelection()* permettent de gérer des filtres par relation, actuellement soit par une liste blanche d'identifiants de clé primaire, soit par une requête SQL le cas échéant, soit les deux.

Une quatrième famille de méthodes permet de gérer des services dédiés à certains algorithmes de clustering. La méthode *getHammingSimilarityMatrix()* permet ainsi de calculer une matrice de similarités entre individus d'un même schéma de relation dont les éléments sont binaires et où un 1 n'est obtenu que si deux individus sont identiques. La méthode *getHammingSimilarityMatrixToPrototypes()* permet quant à elle de calculer une matrice de similarité entre les tuples d'une relation et des prototypes moyens donnés. La méthode *getMeanPrototype()* permet enfin de récupérer un prototype moyen à partir d'un ensemble de tuples d'une même relation. D'autres variantes existent, soit en terme de format de sortie (les méthodes *copyXXX()* permettent d'utiliser des techniques d'écriture sur fichier des SGBD au lieu de tout renvoyer en mémoire, pour des performances accrues), ou en terme de scores (pour calculer une matrice de similarités réelles et non binaires par exemple).

Les méthodes non énoncées ci-dessus sont soit des accesseurs pour des informations quelconques, soit des méthodes utilisées pour des raisons techniques qui ne sont pas détaillées ici. De même, puisque la classe *MockInstance* n'est utilisée que pour les tests, celle-ci ne sera pas détaillée non plus.

7.9 Distributions

La définition de distributions de probabilités sur un ensemble de variables aléatoires se fait grâce à la hiérarchie des classes *RBNDistribution*. La Figure 7.6 montre le diagramme très simple des classes de base. Nous pouvons voir deux classes principales. *RBNDistribution* est la superclasse de tous les types de distributions, qu'ils soient conditionnels ou non. Les distributions conditionnelles héritent quant à elle d'une sous-classe dédiée appelée *RBNConditionalDistribution*. Toutes ces classes sont principalement associées à un ensemble de variables aléatoires. Cet ensemble de variables est un singleton pour une distribution non conditionnelle et contient au contraire plus d'une

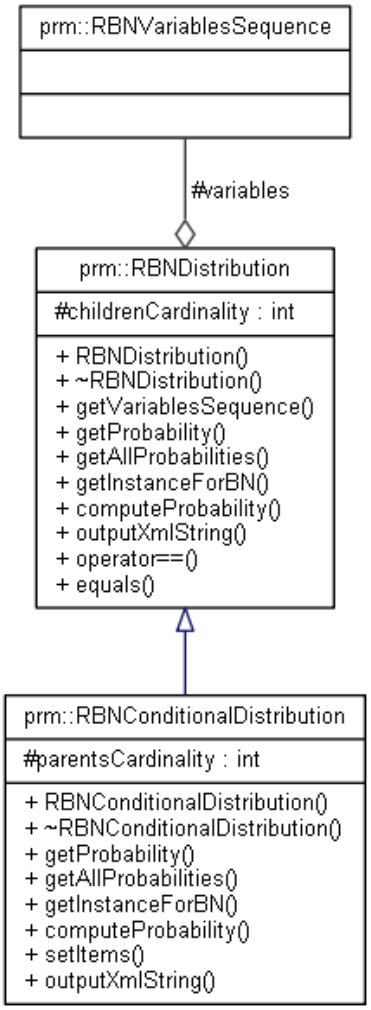


FIGURE 7.6 – Diagramme des classes de distributions

variable pour une distribution conditionnelle, correspondant à l'enfant de la distribution, suivi des parents. Une distribution de probabilités contient toujours une cardinalité de la variable enfant, et une distribution conditionnelle contient en plus la cardinalité des variables parentes, obtenue par produit des cardinalités individuelles.

Les classes de distribution proposent différentes méthodes : *getProbability()* permet de connaître la probabilité d'occurrence d'une configuration particulière de valeurs de la séquence de variables de la distribution ; *getAllProbabilities()* renvoie l'ensemble des probabilités pour toutes les configurations possibles ; *getInstanceForBN()* génère une distribution de RBP (objet ProBT compatible avec les RBP générés) à partir de la distribution de MRP ; computeProbability() est enfin le foncteur qui est utilisé par *getInstanceForBN()* pour instancier les distributions, et renvoie une probabilité d'occurrence pour une séquence de valeurs obtenues dans le RBP. Cette fonction est appelée par la bibliothèque ProBT lors de l'inférence sur un RBP.

À l'écriture de ces lignes, différents types de distributions non conditionnelles sont disponibles dans PILGRIM-Relational : ***RBNDistributionUniform*** définit une distribution uniforme ; ***RBNDistributionProBT*** permet de définir n'importe quelle distribution d'un type proposé par ProBT. Ces deux types sont aussi proposés en version conditionnelle. En plus de cela, les distributions conditionnelles disponibles incluent :

RBNConditionalDistributionTable pour définir une table de probabilités conditionnelles ; ***RBNLinkUncertaintyDistribution*** utilisée avec les modèles à incertitude de référence pour résoudre l'ensemble des parents avant de déterminer la configuration réelle des variables pour enfin obtenir la probabilité associée ; ***RBNConditionalDistributionAllClassEntities*** est la distribution des variables de référence des MRP-IR, déterministe sachant un partitionnement et le groupe dans lequel choisir un individu. Une version est également proposée pour les MRP-IC prenant en compte les variables cluster cible.

7.10 Modèles

La classe ***RBN*** (pour *Relational Bayesian Network*) est la classe de base de tous les modèles relationnels probabilistes. Le diagramme de classe simplifié des différents modèles et classes associées est donné dans la Figure 7.7. Les objets instanciant directement cette classe représentent des MRP orientés attributs. Un MRP contient principalement un ensemble de nœuds, qui sont tous des *IRBNCoreVariable*. Chaque nœud est associé à un ensemble de parents, instance de *RBNVariablesSequence*, et à une distribution, instance de *RBNDistribution*. En outre, un objet de type *RBN* est associé à un schéma relationnel. La classe *RBN* contient de nombreuses méthodes selon le principe du patron de conception *façade* déjà utilisé pour les classes d'instance. Les méthodes de cette classe peuvent ainsi être regroupées en plusieurs familles.

La première famille de méthodes permet de manipuler la structure graphique du MRP : *addNode()*, *removeNode()*, *existsNode()*, *getNode()*, *getNodesForClass()* et *getNodes()* permettent de manipuler les nœuds du modèle ; *clearParents()*, *getParents()*, *hasParent()*, *hasChild()*, *addParent()*, *setParents()* et *getNodeAndParents()* permettent de manipuler les relations de parenté entre nœuds ; les méthodes *getVariables()* et *getVariable()* permettent de récupérer des variables ou séquences de variables selon une syntaxe dédiée pour faciliter le travail de l'utilisateur. Chaque *RBN* s'occupe de la mise à jour transparente de son graphe de dépendances à chaque appel aux méthodes ci-dessus. Si une de ces opérations conduit à la perte de l'acyclicité de ce graphe, alors une exception est levée.

La seconde famille de méthodes permet de manipuler les distributions des noeuds du MRP comme *getDistribution()*, *setDistribution()*, *setDistributionTable()* et *setUniformDistribution()*. Les deux dernières permettent d'accélérer la définition manuelle d'un MRP pour les cas les plus fréquents d'utilisation.

Une troisième famille de méthodes permet la génération d'un RBP à partir d'une instance, supposant l'état courant du *RBN* : *generateGroundBayesianNetwork()* est la méthode principale faisant successivement appel aux sous-méthodes *generateGroundBayesianNetworkNodes()*, *generateGroundBayesianNetworkDependencies()*, *generateGroundBayesianNetworkDistributions()* et finalement *setGroundBayesianNetworkEvidence()*.

Enfin, une dernière famille de méthodes incluant *learnParameters()* et *learnNodeParameters()* permet de réaliser l'estimation des paramètres, c.-à-d. l'instanciation automatique des variables de distributions pour chaque nœud, à partir d'une instance pour la structure de MRP courante. Ces fonctions d'apprentissage prennent en argument une méthode d'estimation de paramètre de type *ParameterEstimationMethod* dont les

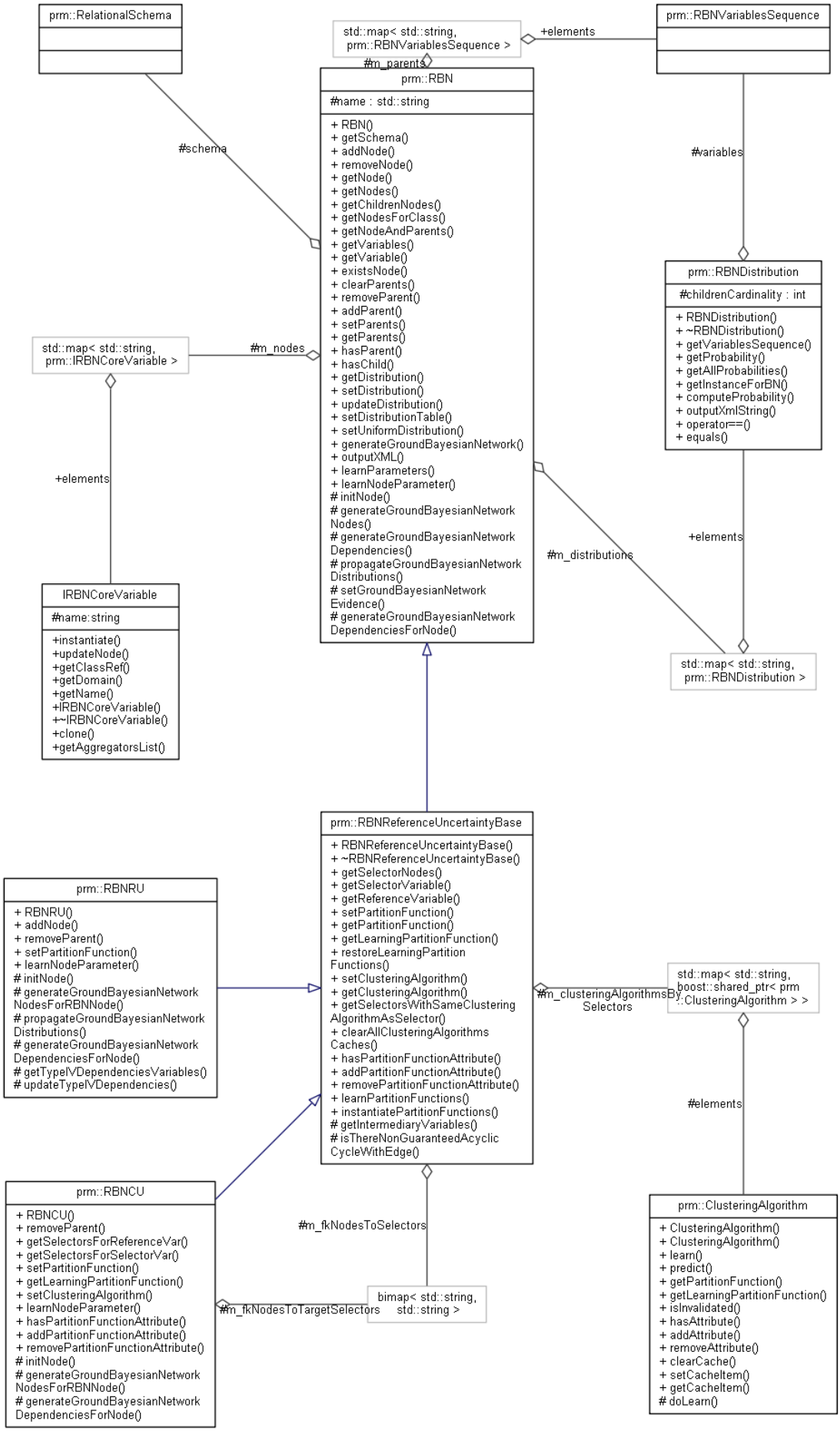


FIGURE 7.7 – Diagramme de classe simplifié du composant modèle probabiliste

sous-types concrets implémentés pour le moment sont ***MaximumLikelihood***, ***MaximumAPosteriori***, ***ExpectedAPosteriori*** et ***Laplace***.

Des extensions des MRP sont disponibles comme autant de sous-classes dans la bibliothèque. En ce qui concerne cette thèse, la sous-classe abstraite *RBNReferenceUncertaintyBase* regroupe le comportement commun des MRP-IR et MRP-IC et conceptualise les modèles relationnels selon le paradigme d'incertitude de référence de façon plus générale. En plus des membres définis dans *RBN*, c.-à-d. nœuds, parents et distributions, cette classe est associée à un ensemble d'algorithmes de clustering (*ClusteringAlgorithm*), un pour chaque nœud sélecteur (ou cluster). De ce fait, elle propose également en plus des méthodes de sa superclasse des méthodes permettant de manipuler ces algorithmes de clustering, et les fonctions de partition obtenues à partir de ceux-ci. En outre, elle propose des méthodes permettant d'apprendre les fonctions de partition de chaque sélecteur (ou variable cluster) étant donné la structure courante du MRP et les algorithmes de clustering définis, et des méthodes permettant de modifier les attributs de fonction de partition pour un nœud.

Il est important de noter la différence entre algorithme de clustering et fonction de partition pour la classe *RBNReferenceUncertaintyBase*. En effet, chaque nœud sélecteur (ou cluster) est associé à une référence vers un objet de type *ClusteringAlgorithm*. Deux noeuds sélecteurs peuvent ainsi être associés au même objet de ce type qui peut potentiellement produire plusieurs fonctions de partition, notamment dans le cas d'algorithmes de co-clustering. La classe *ClusteringAlgorithm* gère donc potentiellement les fonctions de partition de plusieurs variables de référence dans le modèle et les attributs de fonction de partition de chacune. L'ensemble des fonctions de partitions produites par un algorithme de clustering est mis à jour lorsque les attributs de fonction de partition d'un mode sont modifiés.

Par conséquent, les méthodes de la classe *RBNReferenceUncertaintyBase* concernant les algorithmes de clustering et fonctions de partition sont dépendantes les unes des autres. Ainsi, un appel à *setClusteringAlgorithm()* entraîne l'invalidation de la fonction de partition du noeud associé, et il sera nécessaire de réapprendre ces fonctions de partition invalidées avant de pouvoir inférer sur le modèle. De même, un appel à *addPartitionFunctionAttribute()* ou *removePartitionFunctionAttribute()* entraîne la mise à jour de l'algorithme de clustering associé au nœud ciblé et donc l'invalidation de la fonction de partition.

La classe *RBNReferenceUncertaintyBase* a deux sous-classes concrètes : ***RBNRU***, correspondant aux MRP-IR, et ***RBNCU***, correspondant aux MRP-IC. La première classe est proche de la superclasse. La différence majeure dans *RBNCU*, outre les redéfinitions de méthodes, est l'existence des variables cluster cible, et la maintenance d'une structure de données permettant d'assurer que variables cluster source et cible sont liées à la même fonction de partition et au même algorithme de clustering.

7.11 Algorithmes de clustering, Fonctions de partition

Dans la section précédente, nous avons commencé à identifier les différences entre les concepts d'algorithme de clustering et de fonction de partition dans PILGRIM-Relational. Nous nous focalisons davantage sur ces classes dans cette section. Le dia-

gramme de classe simplifié relatif à ces concepts est donné dans la Figure 7.8.

Un algorithme de clustering (*ClusteringAlgorithm*) est potentiellement associé à plusieurs fonctions de partition. Il contient ainsi une séquence de variables aléatoires de référence, et propose une fonction de partition pour chacune. En outre, il maintient à jour pour chaque variable de référence l'ensemble d'attributs de fonction de partition associés. Les méthodes proposées par cette classe permettent principalement de mettre à jour les informations d'attributs de fonction de partition et d'accéder à l'ensemble des informations de la classe. Il est important de noter également que cette classe bénéficie d'un système de mise en cache permettant d'éviter les calculs intempestifs de fonctions de partition pour une configuration d'attributs déjà rencontrés précédemment. Enfin, les fonctions principales de cette classe sont *learn()* et *predict()* qui permettent respectivement d'apprendre les fonctions de partition selon l'état courant défini et d'inférer des fonctions de partition pour de nouveaux individus, en utilisant les fonctions de partition déjà apprises.

La classe *ClusteringAlgorithm* est une classe abstraite dont les raffinages concrets sont actuellement : ***CartesianProduct*** proposant la méthode de partitionnement historique des MRP-IR ; ***NMF***, ***NMTF***, ***FastNMTF*** proposant diverses méthodes de factorisation de matrice utilisées dans les expériences du chapitre 6 ; ***Louvain***, ***LouvainOnNearestNeighborsGraph***, ***LouvainOnSimilarityGraph***, et ***LouvainOnSimilarityAugmentedRelationGraph*** proposant des variantes de l'application de l'algorithme Louvain sur le graphe multi parties du type d'association considéré, sur les informations de similarité à l'intérieur d'un seul mode de données, ou les deux simultanément ; ***LPBRIM*** implémentant la méthode de détection de communautés éponyme dédiée aux graphes bipartis ; ***HypergraphPartitioningConsensus***, un méta-algorithme de clustering où les fonctions de partition sont obtenues par consensus entre plusieurs autres algorithmes de clustering, par construction d'un hyper graphe où chaque nœud est une entité à partitionner, et où un hyper arc existe pour chaque partie d'un partitionnement ; ***NoClusteringAlgorithm***, un type permettant de renseigner une fonction de partition à la main, indiquant qu'il n'y a donc pas réellement d'algorithme de clustering. La fonction *predict()* de la plupart de ces algorithmes consiste en une prédiction basée sur le mode statistique des valeurs de cluster des individus auxquels l'individu à partitionner est connecté. Par exemple, si un nouvel individu t d'un mode est connecté à deux individus s_a et s_b du même ou d'un autre mode, et si s_a et s_b sont connus et ont été partitionnés dans le co cluster c, alors t sera prédit comme faisant partie du co cluster c.

La classe concrète ***PartitionFunction*** définit une fonction de partition pour un ensemble d'individus de même schéma de relation. Elle se compose principalement d'un ensemble d'identifiants d'individus, d'un ensemble d'étiquettes de cluster, et d'un ensemble d'assignation des individus aux clusters. Elle contient de plus l'ensemble d'attributs de fonction de partition ayant mené à ce résultat, le schéma relationnel, le schéma de relation et la variable de référence du type d'association correspondant à ce partitionnement, ainsi que le schéma de relation du type d'entité des individus à partitionner. Elle maintient en outre à jour les fréquences de chaque cluster, ainsi que les distributions *a priori* d'appartenance à un cluster pour un nouvel individu dont on ne sait rien de plus. La classe *PartitionFunction* autorise certains individus à ne pas avoir de cluster. Dans ce cas, ces individus ne sont pas comptabilisés dans les fréquences et distributions mentionnées, et il est possible de récupérer le nombre d'individus réellement partitionnés. Cette classe permet de récupérer les assignations de chaque individu sous

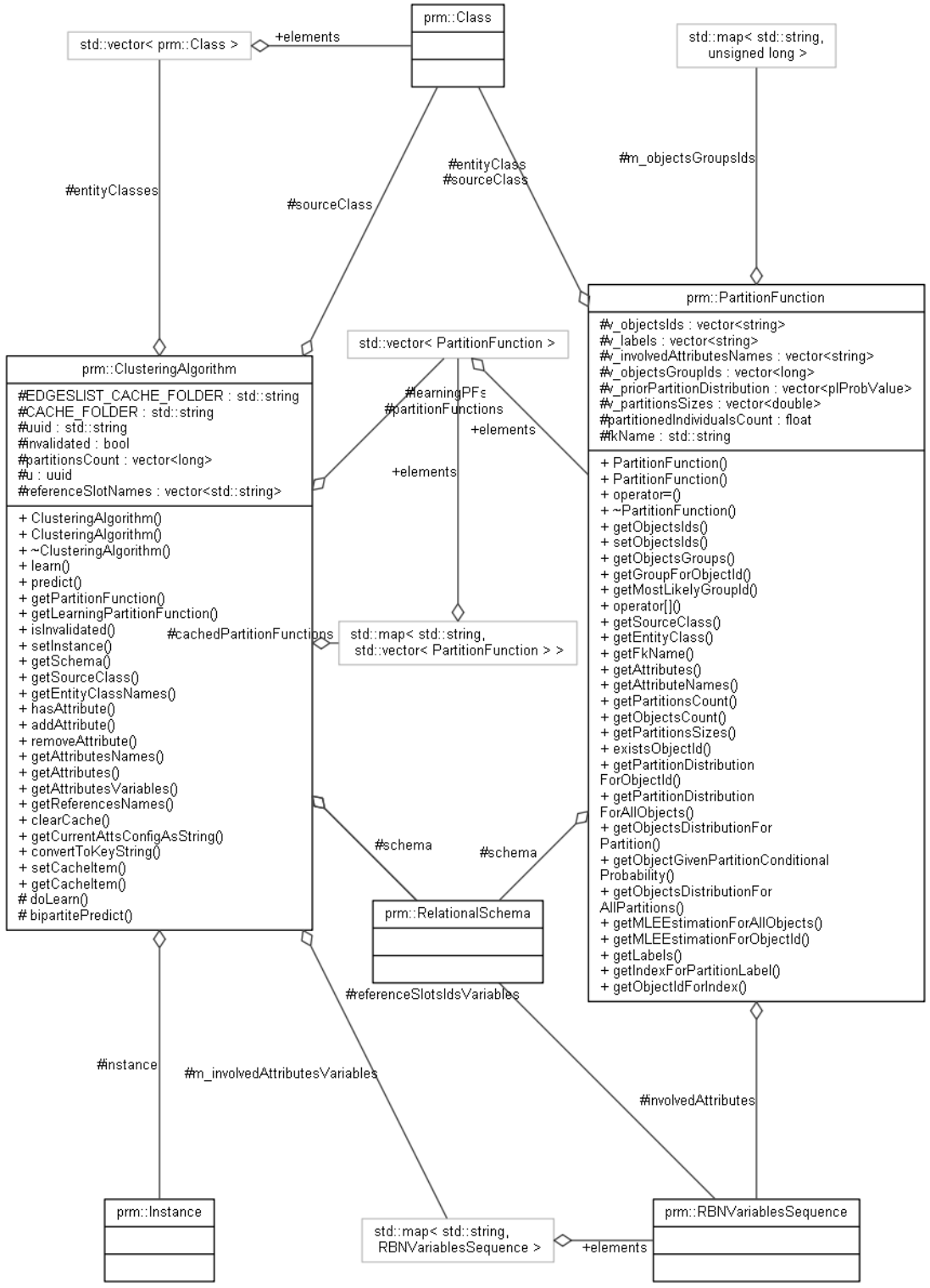


FIGURE 7.8 – Diagramme de classe simplifié du composant fonction de partition et algorithme de clustering

la forme d'estimateur maximisant la vraisemblance, via les méthodes *getMLEEstimationForObject()*, *getMLEEstimationForAllObjects()* et *getObjectsGroups()*, ou sous la forme de distributions de probabilités, via les méthodes *getPartitionDistributionForObjectId()*, *getPartitionDistributionForAllObjects()* et assimilés. Toutefois, en pratique, les distributions sur les clusters ne sont pas utilisées et ont été développées en anticipation de travaux futurs.

7.12 Apprentissage de dépendances

Contrairement à l'apprentissage de paramètres (et de fonctions de partition pour les extensions), l'apprentissage de dépendances n'est pas inclus dans les classes de modèle. À la place, chaque algorithme d'apprentissage possède sa classe et est associé à un modèle fourni en argument. La Figure 7.9 montre le diagramme de classe concernant l'apprentissage de structure dans PILGRIM-Relational.

Actuellement, seules des approches gloutonnes sont proposées pour apprendre les structures des modèles. La classe *AlgoRGS* propose une méthode d'apprentissage gloutonne relationnelle pour les MRP orientés attributs, et est également superclasse de tous les algorithmes gloutons. Elle est principalement associée à une instance d'apprentissage et à un modèle relationnel. Ce dernier est fourni en argument, ce qui lui permet d'être initialisé si nécessaire. En outre, un algorithme de recherche gloutonne est associé à une fonction de score décomposable, afin de calculer l'impact de chaque opération de l'algorithme sur le modèle pour choisir le candidat maximisant à chaque itération ce score. Un algorithme de recherche gloutonne propose intuitivement deux méthodes principales en plus des accesseurs classiques : *generateNeighbours()* permet de générer un ensemble de candidats voisins du modèle courant (par opérations sur le graphe) ; *evaluateDAGNeighbours()* permet d'évaluer l'ensemble des candidats et d'obtenir le meilleur d'entre eux ainsi que son score. Ces méthodes sont appelées successivement dans *computeRBNForSlotChainLength()* pour une valeur constante de taille maximale de séquences de référence, elle même appelée plusieurs fois dans *computeRBN()* pour une taille maximale de séquences de référence croissante jusqu'à atteindre un maximum fourni en argument. La fonction *computeDeltaScore()* calcule la variation de score à partir de deux scores, de façon à retourner un ratio, tel qu'expliqué dans le chapitre 6.

Une sous-classe de ***AlgoRGS*** existe pour chaque extension des MRP disponible. Ainsi, la classe ***AlgoRGSRU*** propose une mise à jour de l'algorithme glouton pour les MRP-IR. Cette classe ajoute deux méthodes principales : *generatePFNeighbours()* et *evaluatePartitionFunctionsNeighbours()* qui respectivement génèrent et évaluent des candidats du modèle courant obtenus par des opérations de voisinage sur les fonctions de partition. Dans la classe *AlgoRGSRU*, tous les candidats sont générés pour l'ensemble des opérations de voisinage (graphe et partitions) et sont ensuite évalués conjointement à chaque itération. Dans la dernière classe, ***AlgoRGSCU***, la génération et l'évaluation des candidats obtenus par opérations sur le graphe sont effectuées avant celles réalisées pour les fonctions de partition à chaque itération, conformément à ce qui est décrit dans le chapitre 6.

La classe abstraite *RScore* définit les propriétés communes de toutes les fonctions

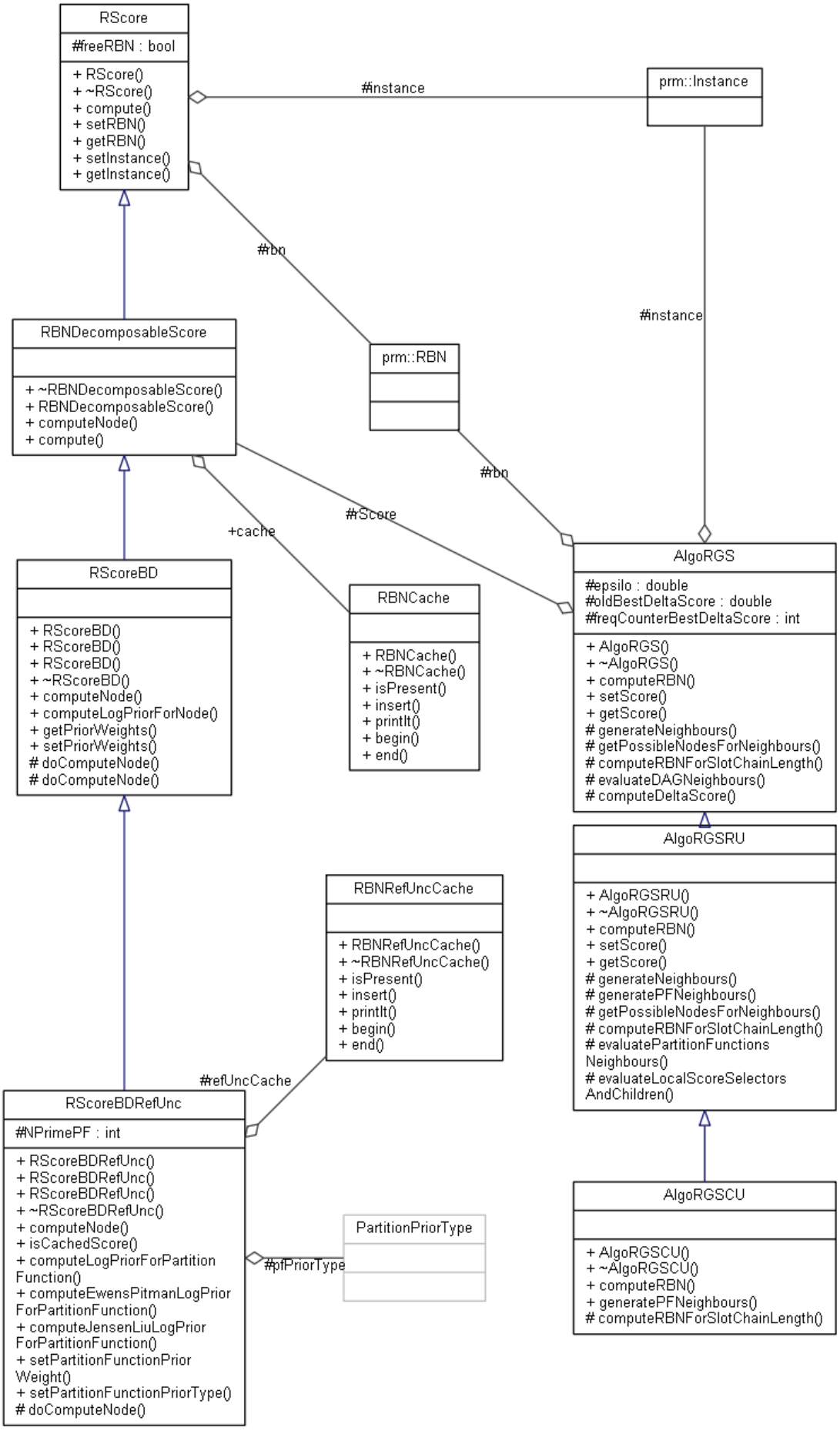


FIGURE 7.9 – Diagramme de classe des types liés à l'apprentissage de modèles

de score utilisées pour l'évaluation des modèles relationnels. Elle contient principalement une référence vers le modèle à évaluer et vers l'instance d'apprentissage, et propose une méthode *compute()* permettant de calculer le score d'un modèle utilisant l'instance considérée. Comme recalculer un score pour un modèle complet est généralement coûteux, nous sommes intéressés par les scores décomposables, qui peuvent être obtenus par somme de scores locaux sur chaque nœud. La sous-classe *RBNDecomposableScore* est une classe abstraite permettant de représenter cette famille de scores. Elle propose une nouvelle méthode appelée *computeForNode()* permettant de calculer le score pour un seul nœud du modèle utilisant l'instance considérée, et implémente *compute()* pour dépendre entièrement de cette méthode. Pour des raisons de performance, nous forçons les algorithmes gloutons à utiliser des scores décomposables par l'architecture des classes. À l'heure actuelle, seul le score ***Bayesian Dirichlet*** est implémenté via les classes ***RScoreBD*** (pour les MRP orientés attributs) et ***RScoreBDRefUnc*** (pour les MRP avec incertitude de référence). La différence entre les deux classes s'explique par les mécanismes de gestion de cache induits, différents, et par l'ajout d'un *a priori* sur les fonctions de partition pour les modèles concernés. Les *a priori* actuellement disponibles sont celui d'Ewens-Pitman et celui de Jensen-Liu (cf. chapitre 3). Les classes de cache ne sont pas détaillées ici mais leur différence s'explique par la clé d'une entrée dans le cache, qui nécessite d'ajouter les attributs de fonction de partition pour les modèles à incertitude de référence, ceux-ci conditionnant également la valeur des scores en plus des seuls parents pour une variable donnée.

7.13 RBP, Inférence

L'inférence dans PILGRIM-Relational passe par la génération d'un RBP, possible à partir des classes de modèle *RBN*, *RBNRU* et *RBNCU*. Considérant l'état courant du modèle, ainsi qu'une instance, l'appel à la fonction *generateGroundBayesianNetwork()* conduit à la génération d'un objet dont le type est dérivé de la classe ***plBayesianNetwork*** de *ProBT*. L'ensemble des fonctionnalités d'inférence sur le RBP est ainsi complètement délégué à la bibliothèque dédiée aux réseaux bayésiens, une fois le modèle à plat généré. Pour plus d'information sur l'inférence dans *ProBT*, le lecteur est encouragé à considérer la documentation sur le sujet[11].

Il est important de noter que le choix de déléguer complètement l'inférence limite les possibilités à l'heure actuelle de PILGRIM-Relational. En effet, la littérature récente sur les modèles graphiques probabilistes liftés [19, 168] montre les limites d'une génération exhaustive d'un modèle propositionnel pour une instance donnée, à la fois en terme du coût des inférences qui peuvent y être faites ensuite, qu'en terme de coût de génération dû à la combinatoire forte entre le nombre de variables du MRP et le nombre d'individus de chaque relation. Des travaux se sont ainsi intéressés à réaliser une génération partielle de ces modèles à plat pour améliorer les performances. Un travail futur pour PILGRIM-Relational sera de considérer une telle approche pour l'inférence et donc d'intégrer l'inférence dans la bibliothèque.

[11] http://www.probayes.com/fr/Bayesian-Programming-Book/downloads/

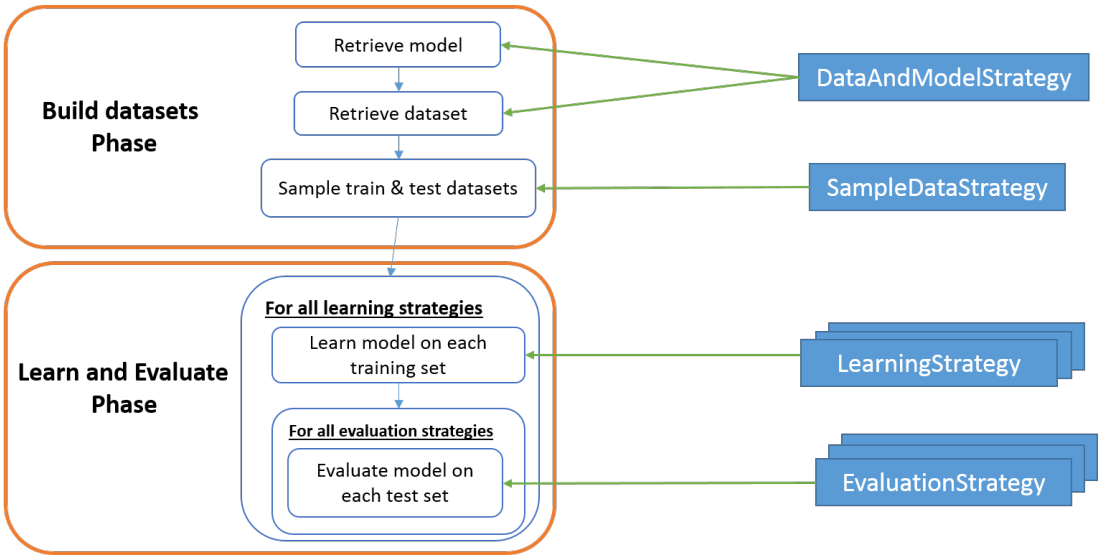


FIGURE 7.10 – Workflow simplifié d'une expérience dans le module dédié de PILGRIM-Relational

7.14 Workflow d'expériences

Au-delà des composants permettant de définir, apprendre et réaliser de l'inférence dans des modèles relationnels probabilistes, PILGRIM-Relational propose enfin un module permettant de réaliser des expérimentations à travers un workflow enchaînant des tâches de différents types abstraits. Pour chacun de ces types, il est possible d'étendre les possibilités du module en implémentant de nouvelles stratégies pour ce type. Instancier une expérience revient alors à composer avec les différentes stratégies disponibles en fonction des besoins.

La Figure 7.10 montre le workflow simplifié de la classe *Experiment* disponible dans la bibliothèque. Celui-ci se compose de deux phases principales. La première correspond à la phase de construction du jeu de données et fait appel à plusieurs stratégies. Les tâches de récupération d'un modèle puis de récupération (ou génération) d'un jeu de données à partir de ce modèle font partie du type de stratégie appelé *DataAndModelStrategy*. Différentes possibilités sont actuellement implémentées sur la plateforme : ***TwoEntitiesOneRelationship*** génère des données pour un modèle comportant deux types d'entités avec un nombre variable d'attributs et un type d'association liant les deux ; ***OneEntityOneRelationPRMCRU*** fait de même pour un type d'entité avec un type d'association binaire uni modale ; ***No*** permet enfin de ne pas déterminer de modèle *a priori* ni de générer des jeux de données, permettant alors de travailler sur des jeux de données réels. Après génération éventuelle de données à partir d'un modèle, ou simplement après définition d'une instance (pour le cas de *No*), la tâche suivante à réaliser est la mise en place des jeux de données d'apprentissage et de test, matérialisée par une stratégie de type *SampleDataStrategy*. Actuellement, seule ***TrainTest*** est proposée, permettant la génération de n sous-jeux de données d'apprentissage et des jeux de test correspondant par complémentation.

La seconde grande phase du workflow d'expérimentation correspond à l'apprentissage des modèles et à leur évaluation. Pour chaque sous-jeu de données d'apprentissage obtenu par l'objet *SampleDataStrategy*, un apprentissage de différents modèles

est réalisé, soit un pour chaque objet dans la liste des stratégies de type *LearningStrategy* de l'expérience. Les apprentissages possibles à l'heure actuelle dans PILGRIM-Relational sont : ***PartitionFunctionAndParameter*** conservant une structure fixe de modèle avec incertitude de référence et apprenant uniquement les fonctions de partition et les paramètres ; ***RelationalGreedySearchCU*** préparant un objet de type *AlgoRGSCU* et lançant l'algorithme sur un MRP-IC initialisé ; ***RelationalGreedySearchRU*** faisant de même pour un objet de type *AlgoRGSRU* et un MRP-IR.

Ensuite, pour chaque modèle appris et pour chaque jeu de données d'apprentissage, celui-ci est évalué pour chaque stratégie d'évaluation ajoutée à la liste des évaluations pour cette expérience, toutes de type *EvaluationStrategy*. Les évaluations actuellement possibles sont : ***AssociationsAndClustersLogLikelihood*** et ***LogLikelihood*** proposant des méthodes pour calculer la vraisemblance du jeu de données de test sur certaines variables ; ***BiModularity*** permettant de calculer la bi modularité au sens de Barber [8] pour les communautés de graphes bipartis ; ***ClusteringAccuracy*** permettant de calculer la précision d'un partitionnement par rapport à un partitionnement de référence ; ***ClustersNumber*** permettant d'obtenir le nombre de clusters d'un partitionnement ; ***NMI*** calculant l'information mutuelle normalisée entre un partitionnement et un autre de référence ; ***Perplexity*** permettant de calculer la perplexité du jeu de données de test à l'égard de certaines variables ; ***Purity*** calculant la pureté d'un partitionnement par rapport à un partitionnement de référence.

Deux types de stratégies supplémentaires sont appelés durant une expérience mais n'apparaissent pas dans le workflow simplifié. Une liste de tâches de type *OutputStrategy* peut être définie pour afficher les résultats des différentes étapes de l'expérience (apprentissage, évaluation, ...). Les tâches de ce type actuellement possibles sont : ***CSV*** permettant de sérialiser les résultats d'évaluation au format CSV ; ***ModelDump*** et ***ModelDumpXml*** permettant d'écrire dans un fichier la structure et les paramètres d'un MRP appris ; ***SVGStructure*** permettant d'afficher un modèle appris en SVG.

Enfin, un ensemble de tâches de type *EvaluationResultsStrategy* peut également être défini, permettant d'évaluer les résultats des évaluations eux-mêmes. Cela permet par exemple d'effectuer des tests de Wilcoxon ou de Friedman [47] sur les différents vecteurs d'évaluation obtenus précédemment.

7.15 Conclusion

Dans ce chapitre, nous avons exposé le problème de disponibilité de fragments logiciels pour réaliser des expérimentations concernant les modèles relationnels probabilistes et avons présenté notre plateforme logicielle PILGRIM-Relational ayant pour objectif de combler ce manque. Celle-ci est centrale pour les travaux de recherche de l'équipe liés aux MRP et à leurs extensions et permet aux différents acteurs de bénéficier d'un socle commun, fruit de la contribution de plusieurs. Toutefois, la portée de cette valorisation ne s'arrête pas à l'équipe, puisqu'elle permet la collaboration concrète avec des entreprises. De plus, à moyen terme, sa diffusion comme logiciel libre permettra à chacun d'utiliser les modèles relationnels probabilistes mais aussi de se les approprier pour aller plus loin.

8

Conclusion et Perspectives

Sommaire

8.1 Conclusion

L'apprentissage de modèles dans un contexte multi relationnel sous incertitude de liens est un problème majeur ayant de nombreuses finalités, que ce soit pour obtenir des connaissances explicatives du modèle, ou pour réaliser des prédictions pour des individus. Accroître la qualité d'apprentissage de ces modèles permettrait d'améliorer le résultat de tâches telles que la recommandation, la détection de communautés et de nombreux autres problèmes d'aide à la décision dans un contexte de données hétérogènes.

Nous avons étudié dans cette thèse l'apprentissage de modèles à partir de données multi relationnelles dans un contexte précis d'incertitude de références, où l'énumération des tuples des différents types d'objets du domaine de définition relationnel est connue, mais où les références entre tuples sont manquantes. Le but est alors de trouver la distribution des variables aléatoires de références en s'abstrayant de la liste exhaustive des tuples grâce à des algorithmes de partitionnement. Il est important de noter que, même si ce paradigme de représentation semble assez restreint, il est en revanche souvent possible de revenir dans un contexte d'incertitude de références même si l'on ne connaît pas explicitement le nombre d'associations réelles. Dans le cas d'un système de recommandations par exemple, si nous souhaitons proposer 10 objets à chaque utilisateur, nous pouvons créer ce même nombre d'associations et revenir dans notre contexte de représentation.

Nous avons analysé durant cette thèse les modèles relationnels probabilistes avec incertitude de références proposant un cadre théorique pour l'apprentissage de modèles graphiques probabilistes de second ordre dirigés dans le contexte de représentation qui nous intéresse. Nous avons vu que ces modèles sont contraints par leur définition, tant sur le plan des possibilités de partitionnement que de la sémantique des structures qu'il est possible de définir. Nous avons de ce fait proposé des extensions à ces modèles historiques de façon à pouvoir s'extraire de ces limites. Les *modèles relationnels probabilistes avec incertitude de clustering* sont l'aboutissement de nos propositions et diffèrent des modèles définis par Getoor [67, 65] : les contraintes relâchées des fonctions de partition permettent l'utilisation de méthodes de partitionnement relationnel et ainsi l'apprentissage simultané des partitions pour un même type d'association. Il est alors toujours possible dans un cas extrême d'apprendre des fonctions de partition sans aucun attribut significatif dans les types d'entités concernés ; l'ajout de variables d'appartenance des individus aux groupes permet de réaliser de l'inférence de groupe après instanciation d'un modèle factorisé, ainsi que de découvrir des connaissances entre les différents types d'association. Nous avons en outre proposé un algorithme d'apprentissage glouton pour ces modèles et détaillé les règles à suivre pour leur instanciation et le calcul des distributions de probabilités des instances.

Le relâchement des contraintes des modèles historiques au profit des algorithmes de partitionnement relationnel nous a conduits à nous intéresser à la littérature sur ces derniers. Nous avons retenu principalement trois familles de méthodes : 1) les approches de factorisation non négative de matrices (cf. 4.2.3) présentent l'avantage d'être déjà adaptées au co partitionnement de deux ensembles d'individus distincts et possèdent des extensions permettant de régulariser les résultats avec des matrices de similarités entre individus d'un même ensemble. Toutefois, elles se sont révélées diffi-

ciles à faire converger et présentent de loin les temps d'exécution les plus longs parmi les algorithmes utilisés durant cette thèse, ce qui rend compliqué leur usage intégré à une heuristique d'apprentissage de structure de modèles probabilistes ; 2) les approches de partitionnement dans des graphes présentent généralement des temps d'exécution très courts, notamment leurs versions multi niveaux (cf. 4.3.4), mais ne supportent pas nativement le partitionnement simultané de plusieurs ensembles de nœuds disjoints ni la régularisation par similarités. Nous avons toutefois découvert une équivalence entre une factorisation non négative de matrices régularisée par des matrices laplaciennes et une coupe minimale de graphe avec des liens supplémentaires de similarités, permettant de rapprocher ces deux familles et ainsi profiter des avantages de chacune ; 3) les approches de détection de communautés dans les graphes permettent de découvrir automatiquement des densités dans un graphe, souvent sans paramètres. Les communautés sont alors définies par rapport à leurs propriétés topologiques qui sont recherchées au sein du graphe entier. Nous nous sommes principalement intéressés aux méthodes optimisant le critère de modularité, où la qualité d'une communauté se mesure par l'écart de densité relatif à un graphe aléatoire avec la même distribution de degrés des nœuds. Les résultats obtenus avec l'une de ces méthodes, Louvain [16], se sont révélés intéressants dans un contexte non régularisé et dense, mais difficile à interpréter dans les autres cas, à cause de la sensibilité forte de ce type d'algorithmes à la nature multimodale des données et à l'ajout de liens de similarités entre individus d'un même ensemble d'entités. Nous avons terminé les expériences du dernier chapitre par l'utilisation d'une autre méthode, appelée LP&BRIM [134], adaptée au contexte biparti grâce à l'optimisation d'un score de *bi modularité* et effectué une régularisation par attributs, en recherchant un consensus entre les résultats obtenus par cette méthode et des partitionnements utilisant les seuls attributs de régularisation pour chaque type d'entité impliqué. Cette dernière a montré de bons résultats et permet de simplifier la mise à jour des scores des structures candidates après modification des ensembles d'attributs, la rendant prometteuse pour la suite.

8.2 Discussion et perspectives à court et moyen termes

Les travaux réalisés durant cette thèse ont permis de confirmer l'intérêt d'étendre les modèles relationnels probabilistes avec incertitude de références pour pouvoir utiliser des fonctions de partition relationnelles et permettre la découverte de dépendances entre associations, limitant ainsi le biais de représentation historique des modèles tout en utilisant un maximum d'information des jeux de données pour l'apprentissage de structure et de paramètres. Toutefois, les modèles proposés possèdent encore plusieurs axes d'amélioration conduisant à des perspectives à court et moyen termes.

En effet, les expériences réalisées durant cette thèse semblent montrer une sensibilité des résultats d'apprentissage de structure aux résultats de partitionnement. Aussi, le choix des algorithmes générant les groupes doit être fait avec précaution. Dans cette thèse, nous insistons sur la nécessité pour ces algorithmes d'être capables de réaliser du co-partitionnement, et de savoir traiter des ensembles d'individus disjoints simultanément, dans des contextes d'associations n-aires. Les capacités de régularisation semblent également apporter un avantage, mais l'ajout d'informations complémentaires semble pouvoir être réalisé par des approches de consensus, comme vu précédemment. Au-delà de ces propriétés, les règles précises de choix pour un contexte de données particulières seraient à étudier plus en détail. En outre, la nature heuristique

de la plupart des méthodes calculables en temps raisonnable rend nécessaire le suivi de processus robustes et le choix d'algorithmes relativement résistant à un changement léger dans les données. Des pistes de choix en ce sens peuvent inclure l'utilisation d'un consensus entre des partitions obtenues par usage répété d'un même algorithme sur les données, ou encore l'utilisation de méthodes fonctionnant en deux temps, simplifiant les données avant de réaliser un partitionnement comme les méthodes de coupe de graphe multi niveaux.

Ensuite, l'utilisation d'attributs dans les fonctions de partition apprises dans les modèles avec incertitude de clustering ne semble pas systématique et de nombreux résultats d'apprentissage se sont montrés exempts de tels attributs. Cela ne semble pas totalement dénué de sens, puisque si la topologie des associations est suffisamment claire, car suffisamment d'associations sont présentes dans les données, alors ajouter des informations supplémentaires ne décrivant pas exactement la même topologie peut faire diverger l'algorithme de la partition optimale. L'ajout d'attribut semble avoir un rôle plus important si la topologie des associations n'est pas claire, p. ex. s'il manque de nombreuses associations pour observer des densités, avec par exemple de nombreux individus présentant 0 ou une seule association. Aussi, l'exploration d'autres espaces d'opérateurs que celui de la mise à jour des attributs de fonctions de partitions serait à considérer dans le futur. À titre d'exemple, nous pourrions considérer l'utilisation d'opérateurs permettant de parcourir l'espace des α réels si l'on considère une fonction de partition basée sur des mesures de similarité par α-divergence (cf. 4.2.3), ou encore l'utilisation d'opérateurs permettant de fusionner des fonctions de partition si l'on découvre que leurs variables cluster sont fortement corrélées, suggérant alors une topologie proche.

Enfin, l'utilisation d'une heuristique gloutonne pour l'apprentissage de modèles graphiques probabilistes est souvent sensible à la convergence vers des optima locaux qui peuvent être de mauvaise qualité. Nous tentons de limiter les risques par l'utilisation de fonctions de co-clustering prenant en compte l'intégralité des informations d'association pour calculer un ensemble de fonctions de partition pour un type donné, et non pas séparément chaque fonction de partition. De surcroît, cette approche permet de trouver les attributs corrélés à un type d'association sans avoir à parcourir l'ensemble des combinaisons de façon heuristique, ce qui peut éviter à l'algorithme d'apprentissage certaines combinaisons d'attributs sous-optimales. Toutefois, comme énoncé précédemment, la qualité d'une structure étant dépendante de la qualité des partitions découvertes, et les algorithmes permettant de trouver ces dernières rapidement étant basés sur des heuristiques, le risque de convergence vers un optimum local est toujours possible. Il serait ainsi intéressant de faire évoluer l'algorithme d'apprentissage vers une méthode non entièrement gloutonne, comme une méthode hybride mélangeant approche par contrainte et méthode gloutonne, originalement proposée par Tsamardinos [177] pour les réseaux bayésiens et étendue au cas des modèles relationnels probabilistes par Ben Ishak [11].

8.3 Perspectives générales

Au-delà des perspectives liées à l'amélioration des modèles proposés dans cette thèse en tant que telle, les résultats présentés dans ce manuscrit conduisent également à des perspectives à plus long terme, soit vers de nouveaux modèles, ou vers la généralisa-

tion des méthodes utilisées dans cette thèse dans d'autres cadres théoriques que les modèles relationnels probabilistes.

Tout d'abord, l'utilisation de partitionnements *durs* et plats pour les fonctions de partition, bien que permettant souvent des apprentissages rapides, n'est pas nécessairement la plus adéquate et peut limiter les performances obtenues lors de l'apprentissage des modèles. Deux propriétés semblent intéressantes pour l'obtention de partitionnements qualitatifs dans les jeux de données complexes : l'assignation probabiliste des individus aux groupes ou la multi assignation d'un individu à plusieurs groupes, et la hiérarchisation des regroupements. Il est important de noter que nous avons déjà utilisé à certaines reprises des méthodes de partitionnement donnant des hiérarchies en sortie. Toutefois, nous avons considéré pour ces méthodes un unique niveau de résultats. Nous pourrions prendre en compte ces hiérarchies de façon plus importante en étendant le modèle proposé dans le chapitre 6 de cette thèse, en ajoutant par exemple une variable cluster pour chaque niveau de hiérarchie du résultat de partitionnement. Une autre façon de considérer la hiérarchie en restant dans le cadre du chapitre 6 pourrait être de considérer un espace d'opérateurs de recherche pour naviguer dans les différents niveaux de hiérarchie et n'en considérer qu'un seul à une itération donnée. Le passage à du partitionnement probabiliste est plus difficile à envisager en terme de coûts d'apprentissage durant la sélection de modèles. En revanche, la multi assignation d'un individu aux groupes pourrait être envisagée en considérant des algorithmes de partitionnement avec chevauchement tels que la méthode de percolation de cliques [48] dans un graphe.

En outre, au-delà de la notion de topologie assimilée dans cette thèse à la seule notion d'appartenance à une communauté, il serait intéressant de considérer d'autres informations pour décrire la topologie à partir de laquelle nous souhaitons apprendre nos modèles, p. ex. la notion de *rôle* d'un individu dans une communauté [110], identifiant son rapport vis-à-vis d'elle, par exemple sous la forme d'une mesure de distance de l'individu à la frontière de sa communauté.

Ensuite, nous nous sommes restreints à l'étude du paradigme d'incertitude de référence dans nos travaux, mais il nous semble que l'utilisation d'algorithmes de partitionnement serait tout aussi utile dans des paradigmes différents, notamment celui d'incertitude d'existence. En effet, nous avons vu qu'utiliser des algorithmes de partitionnement relationnels, permettant de construire simultanément les différentes partitions pour les différents modes des données, nous permettait d'utiliser l'information topologique des associations pour en déduire de la connaissance. Appliquer ces techniques à l'incertitude d'existence permettrait ainsi d'apprendre les distributions d'existence des associations en utilisant également leur topologie pour un jeu de données.

En outre, nous avons considéré dans cette thèse le cas des modèles relationnels probabilistes, mais d'autres modèles graphiques probabilistes de second ordre dirigés existent. Nous pourrions ainsi considérer une méthode d'apprentissage dans les DAPER [84, 127] utilisant des algorithmes de partitionnement relationnel. Ceci permettrait de surcroît d'étendre l'utilisation de ces modèles aux cas d'incertitude de structure, une extension inexistante à notre connaissance pour le moment. Au-delà des modèles dirigés, nous pourrions également utiliser des techniques similaires pour les modèles non dirigés tels que les réseaux logiques de Markov [160]. Kok et Domingos [103, 104] ont proposé des travaux de type *bottom up* dans lesquels, à partir d'une instance com-

plètement représentée sous forme de graphe, les nœuds étaient itérativement fusionnés en fonction des similarités de liens qu'ils avaient avec les autres. Il serait intéressant d'étudier si l'utilisation de fonctions de partition locales à chaque type d'association, comme dans nos travaux, permettrait de diviser le problème pour gérer de grandes instances.

Enfin, et contrairement à l'idée de diviser pour mieux régner défendue précédemment, il serait également intéressant de discuter le choix de conserver un apprentissage local des fonctions de partition à chaque type d'association et de considérer l'apprentissage de modèles relationnels probabilistes comportant des variables latentes de clustering obtenues par des méthodes de clustering multi relationnelles [182] où l'information de plusieurs types d'association sont prises conjointement en compte pour l'obtention d'un partitionnement unique, voire d'un ensemble de partitionnements dans un contexte multi vues [30]. Il serait alors possible de remplacer les variables de cluster définies pour les modèles du chapitre 6 par ces nouvelles variables latentes, voire de conserver les deux ensembles.

8.4 Perspectives logicielles

Comme énoncé dans le chapitre 7, les contributions de cette thèse incluent une part significative de valorisation logicielle, à travers le développement de la plateforme PILGRIM et notamment de son sous-projet PILGRIM-Relational. L'objectif de ce projet n'est pas uniquement de permettre aux différents acteurs de l'équipe de réaliser des expériences autour de l'apprentissage et de l'inférence de modèles graphiques (relationnels) probabilistes dirigés. Il a également pour but d'être partagé avec la communauté scientifique, ainsi que toute organisation publique ou privée souhaitant intégrer l'utilisation de tels modèles dans ses algorithmes. Aussi, cette plateforme est actuellement en cours d'industrialisation, et sera bientôt disponible au public dans une version GPL.

Concernant les perspectives concrètes de développements pour la plateforme, nous pouvons proposer plusieurs pistes d'extensions au-delà des implémentations relatives aux perspectives scientifiques susmentionnées, et en dehors du cadre de l'industrialisation en cours. Tout d'abord, une limite actuellement forte de PILGRIM-Relational est l'impossibilité de réaliser de l'inférence sur un MRP sans générer un RBP de façon complète. Permettre la génération de RBP partiels par des méthodes exactes ou approchées est donc d'un enjeu majeur pour la réduction de la complexité spatiale relative à l'inférence.

Ensuite, les différents algorithmes implémentés pourraient bénéficier de performances accrues grâce à une meilleure utilisation des ressources matérielles disponibles sur la machine où ils s'exécutent. La parallélisation du code permettant l'usage des différents processeurs de la machine serait un objectif prioritaire en ce sens, p.ex. dans le but de calculer de façon simultanée l'impact de différentes opérations candidates lors d'un apprentissage glouton. En outre, l'utilisation des processeurs graphiques présents sur certaines machines permettrait également d'étendre les capacités de calcul disponibles et leur utilisation dans la plateforme serait aussi pertinente. Une optimisation plus aboutie des caches maintenus par l'application pourrait aussi être considérée, en permettant leur sauvegarde partielle sur le disque, tout en conservant un temps d'accès court à ces structures.

Pour les modèles relationnels probabilistes utilisant des algorithmes de partitionnement, tels que les MRP-IR et MRP-IC, l'ajout de nouvelles implémentations de clustering permettrait de traiter des cas de données divers et variés, tout en assurant une veille indispensable sur ce domaine dynamique. Plus généralement, l'ajout de nouvelles distributions de probabilités, de types de variables aléatoires, de protocoles de données, et autres instanciations des concepts élémentaires, fait partie des extensions indispensables pour la pérennité et l'utilisabilité de la plateforme.

Pour finir, il est nécessaire de soulever l'importance de porter une attention particulière à la simplicité d'utilisation de la plateforme au fil des mises à jour. Cette idée a motivé certains choix d'implémentation jusqu'ici, permettant ainsi la définition textuelle de séquence de parents complexes pour un nœud incluant des séquences de références et des agrégations, ou encore l'ajout de méthodes dans les classes de modèles permettant de définir des distributions usuelles de façon simple pour les variables aléatoires impliquées. Il est d'autant plus important de mentionner ce point puisque la plateforme est développée et maintenue en équipe, pour des besoins variés au fil de travaux de recherche différents. Un objectif d'implémentation en ce sens serait de définir un langage textuel simple permettant de définir manuellement des modèles partiels sur lesquels réaliser de l'inférence, de l'apprentissage de structure ou de paramètres.

Bibliographie

[1] Charu C. Aggarwal and Chandan K. Reddy. *Data Clustering : Algorithms and Applications (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series)*. 2013. 154

[2] Edoardo M. Airoldi, David M. Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9 :1981–2014, 2008. 30

[3] Richard D. Alba. A graph theoretic definition of a sociometric clique. *The Journal of Mathematical Sociology*, 3(1) :113–126, 1973. 89

[4] David J. Aldous. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII*, pages 1–198, 1983. 72

[5] Kevin Bache and Moshe Lichman. UCI Machine Learning Repository [http ://archive.ics.uci.edu/ml]. Irvine, CA : University of California, School of Information and Computer Science, 2013. 100

[6] Olav Bangsø, Helge Langseth, and Thomas D. Nielsen. Structural learning in object oriented domains. In *Proceedings of the 14th Florida Artificial Intelligence Research Society (FLAIRS) Conference*, pages 340–344, 2001. 41

[7] Olav Bangsøand Pierre-Henri Wuillemin. Object Oriented Bayesian Networks A Framework for Topdown Specification of Large Bayesian Networks and Repetitive Structures. 2000. 39, 157

[8] Michael J. Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6) :1–11, 2008. 92, 93, 180

[9] Nicola Barbieri, Giuseppe Manco, Riccardo Ortale, and Ettore Ritacco. Balancing Prediction and Recommendation Accuracy : Hierarchical Latent Factors for Preference Data. In *Proceedings of the 12th SIAM International Conference on Data Mining SDM*, pages 1035–1046, 2012. 147

[10] Nicola Barbieri, Giuseppe Manco, and Ettore Ritacco. A probabilistic hierarchical approach for pattern discovery in collaborative filtering data. In *Proceedings of the 19th Italian Symposium on Advanced Database Systems*, pages 239–246, 2011. 147

[11] Mouna Ben Ishak. *Probabilistic relational models : learning and evaluation. The relational Bayesian networks case.* PhD thesis, 2015. 51, 162, 184

[12] Julian Besag. Statistical Analysis of Non-lattice Data. *Journal of the Royal Statistical Society*, 24(3) :179–195, 1975. 53

[13] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1) :5, 2007. 23, 30

[14] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining, ICDM 2004*, pages 19–26, 2004. 136

[15] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4. 2006. 22

[16] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, (10) :P10008, 2008. 91, 119, 138, 151, 154, 183

[17] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional Functional Dependencies for Data Cleaning. In *Proceedings of the 23rd International Conference on Data Engineering*, pages 746–755, 2007. 32

[18] Stephen P. Borgatti and Martin G. Everett. Network analysis of 2-mode data. *Social networks*, 19 :243–269, 1997. 92

[19] Rodrigode Salvo Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In *Proceedings of the 2005 International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1319–1325, 2005. 178

[20] Rasmus Bro. PARAFAC. Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2) :149–171, 1997. 86

[21] Jean-Philippe Brunet, Pablo Tamayo, Todd R. Golub, and Jill P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the national academy of sciences*, 101(12) :4164–4169, 2004. 80

[22] Aydin Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent Advances in Graph Partitioning. In *Algorithm Engineering - Selected Topics, to app., ArXiv :1311.3144, 2014.* 88

[23] Wray Buntine. Theory refinement on Bayesian networks. *Proceedings of the 7th conference on Uncertainty in Artificial Intelligence (UAI)*, pages 52–60, 1991. 38

[24] Markus Buschle, Johan Ullberg, Ulrik Franke, Robert Lagerström, and Teodor Sommestad. A tool for enterprise architecture analysis using the PRM formalism. In *Lecture Notes in Business Information Processing*, volume 72 LNBIP, pages 108–121, 2011. 53

[25] Deng Cai, Xiaofei He, Xiaoyun Wu, and Jiawei Han. Non-negative Matrix Factorization on Manifold. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 63–72. Ieee, 2008. 84

[26] Romain Campigotto, Patricia Conde Céspedes, and Jean-Loup Guillaume. A Generalized and Adaptive Method for Community Detection. *arXiv preprint arXiv :1406.2518*, pages 1–18, June 2014. 91, 93, 151, 154

[27] John D. Caroll, Geert De Soete, and Sandra Pruzansky. Fitting of the latent class model via iteratively reweighted least squares CANDECOMP with nonnegativity constraints. In R. Coppi and S. Bolasco, editors, *Multiway Data Analysis*, pages 463–472. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 1989. 86

[28] Rommel Carvalho, Kathryn Laskey, Paulo Costa, Marcelo Ladeira, Laecio Santos, and Shou Matsumoto. UnBBayes : Modeling Uncertainty for Plausible Reasoning in the Semantic Web. In *Semantic Web*, chapter 1, pages 1–26. 2010. 158

[29] George Casella, Elías Moreno, and F. Javier Girón. Cluster Analysis, Model Selection, and Prior Distributions on Models. *Bayesian Analysis*, 9(3) :613–658, September 2014. 72, 145

[30] Kamalika Chaudhuri, Sham M. Kakade, Karen Livescu, and Karthik Sridharan. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 129–136, 2009. 136, 186

[31] Chung-Kuan Cheng. The Optimal Partitioning Of Networks. *Networks*, 22(3) :297–315, 1992. 86

[32] David M. Chickering. Learning equivalence classes of Bayesian-network structures. *The Journal of Machine Learning Research*, 2 :445–498, 2002. 37

[33] David M. Chickering. Optimal structure identification with greedy search. *The Journal of Machine Learning Research*, 3 :507–554, 2002. 37

[34] C. K. Chow and C. N. Liu. Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory*, 14(3) :462–467, 1968. 37

[35] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun Ichi Amari. *Nonnegative Matrix and Tensor Factorizations : Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. John Wiley and Sons, 2009. 117

[36] Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6) :377–387, 1970. 31

[37] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3) :393–405, 1990. 35

[38] Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4) :309–347, 1992. 37

[39] M. Cooper and J. Foote. Summarizing video using non-negative similarity matrix factorization. In *IEEE Workshop on Multimedia Signal Processing.*, pages 25–28, 2002. 80

[40] Anthony Coutant, Hoel Le Capitaine, and Philippe Leray. On the equivalence between regularized NMF and similarity-augmented graph partitioning. In *Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) Conference*, pages 531–536, 2015. 25, 80, 103

[41] Anthony Coutant, Philippe Leray, and Hoel Le Capitaine. Learning Probabilistic Relational Models Using Non-Negative Matrix Factorization. In *Proceedings of the 27th International Florida Artificial Intelligence Research Society Conference*, pages 490–495, 2014. 25, 106

[42] Anthony Coutant, Philippe Leray, and Hoel Le Capitaine. Probabilistic Relational Models with Clustering Uncertainty. In *Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN)*, pages 439–446, 2015. 25, 126

[43] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*, volume 1. Cambridge University Press, 2009. 35

[44] Christopher J. Date. *An Introduction to Database Systems*. 2003. 31

[45] Rina Dechter. Bucket elimination : A unifying framework for probabilistic inference. In *Learning in graphical models*, pages 75–104. 1998. 48

[46] Charo I. Del Genio, Thilo Gross, and Kevin E. Bassler. All Scale-Free Networks Are Sparse. *Physical review letters*, 107(17) :178701, 2011. 134

[47] Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7 :1–30, 2006. 180

[48] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical Review Letters*, 94(16), 2005. 185

[49] Inderjit S. Dhillon. Co-clustering documents and words using Bipartite Spectral Graph Partitioning. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 269–274, 2001. 80, 87

[50] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted Graph Cuts without Eigenvectors : A Multilevel Approach. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 29(11) :1944–1957, 2007. 88, 95, 99, 138

[51] Chris Ding, T Li, W Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 126–135, 2006. 74, 83, 84, 97, 109, 119

[52] Ying Ding. Community detection : topological vs. topical. *Journal of Informetrics*, 5(4) :498–514, 2011. 134

[53] Yefim A. Dinitz. Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. *Soviet Math. Doklady*, 11 :1277–1280, 1970. 87

[54] Pedro Domingos and William Austin Webb. A Tractable First-Order Probabilistic Logic. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1902–1909, 2012. 158

[55] Norman R. Draper and Harry Smith. *Applied Regression Analysis*. New York : Wiley, 1981. 144

[56] Lucas Rego Drumond, Ernesto Diaz-Aviles, Lars Schmidt-Thieme, and Wolfgang Nejdl. Optimizing Multi-Relational Factorization Models for Multiple Target Relations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 191–200, 2014. 85

[57] M. A. Efroymson. Multiple regression analysis. In *Mathematical methods for digital computers*, pages 191–203, 1960. 144

[58] Charles M. Fiduccia and Robert M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181, 1982. 87

[59] Lester Randolph Ford and Delbert Ray Fulkerson. *Flows in Networks*. Princeton Univ. Press, 1962. 87

[60] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5) :75–174, June 2010. 137

[61] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 36–41, 2007. 122

[62] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning Probabilistic Relational Models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1300–1309, 1999. 24, 30, 46, 47, 48, 49, 50, 62, 130, 157

[63] Eric Gaussier and Cyril Goutte. Relation between PLSA and NMF and Implications Categories and Subject Descriptors. In *Proceedings of the 28th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 601–602, 2005. 83

[64] A. George and Joseph W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. 1981. 87

[65] Lise Getoor. *Learning Statistical Models from Relational Data*. PhD thesis, Stanford, 2001. 24, 47, 50, 56, 59, 72, 74, 76, 182

[66] Lise Getoor, Nir Friedman, and Daphne Koller. Probabilistic Relational Models. In *Introduction to Statistical Relational Learning*, chapter 5, pages 129–174. 2007. 30, 50, 52, 56, 76, 146

[67] Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *The Journal of Machine Learning Research*, 3 :679–707, 2003. 24, 30, 56, 76, 182

[68] Lise Getoor and John Grant. PRL : A probabilistic relational language. *Machine Learning*, 62(1-2 SPEC. ISS.) :7–31, January 2006. 76

[69] Lise Getoor, Eran Segal, Ben Taskar, and Daphne Koller. Probabilistic models of text and link structure for hypertext classification. In *IJCAI workshop on text learning : beyond supervision*, pages 24–29, 2001. 30

[70] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. 2007. 23, 30, 69

[71] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. *Proceedings of the 14th ACM international Conference on Information and Knowledge Management (CIKM)*, 189(3–4) :195–200, 2005. 23

[72] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4) :921–940, 1988. 87

[73] Quanquan Gu and Jie Zhou. Co-clustering on manifolds. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discoverry and Data Mining (KDD)*, pages 359–367. ACM Press, 2009. 84

[74] Jean Loup Guillaume and Matthieu Latapy. Bipartite structure of all complex networks. *Information Processing Letters*, 90(5) :215–221, 2004. 92, 135

[75] Jean-Loup Guillaume and Matthieu Latapy. Bipartite graphs as models of complex networks. *Combinatorial and Algorithmic Aspects of Networking*, 3405 :127–139, 2005. 92, 135

[76] Roger Guimerà and Luís a Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028) :895–900, 2005. 91

[77] Roger Guimerà, Marta Sales-Pardo, and Luís A Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 70(2), 2004. 91

[78] Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral. Module identification in bipartite and directed networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 76(3) :18–20, 2007. 92, 135

[79] Richard A. Harshman. Foundations of the PARAFAC procedure : Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(10) :1–84, 1970. 86

[80] Richard A. Harshman. PARAFAC2 : Mathematical and technical notes. *UCLA working papers in phonetics*, 22(10) :30–44, 1972. 86

[81] David Heckerman. A Tutorial on Learning With Bayesian Networks. *Innovations in Bayesian Networks*, 1995(November) :33–82, 1996. 37

[82] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *The Journal of Machine Learning Research*, 1 :49–75, 2001. 23, 52

[83] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks : The combination of knowledge and statistical data. *Machine Learning*, 20(3) :197–243, 1995. 38

[84] David Heckerman, Christopher Meek, and Daphne Koller. Probabilistic entity-relationship models, PRMs, and plate models. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004. 52, 185

[85] Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of the 22nd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999. 83, 97

[86] Thomas Hofmann and Jan Puzicha. Latent Class Models for Collaborative Filtering. *Machine Learning*, 16(i) :688–693, 1999. 97

[87] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5 :1457–1469, 2004. 83

[88] Zan Huang, D Zeng, and Hsinchun Chen. A unified recommendation framework based on Probabilistic Relational Models. *14th Annual Workshop on Information Technologies and Systems (WITS)*, pages 8–13, 2004. 76, 166

[89] Alexander T. Ihler, John Iii, and Alan S. Willsky. Loopy belief propagation : Convergence and Effects of Message Errors. *Journal of Machine Learning Research*, 6 :905–936, 2005. 35

[90] Manfred Jaeger. Relational bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial intelligence*, pages 266–273, 1997. 52, 157

[91] Manfred Jaeger. Relational bayesian networks : a survey. *Electronic Transactions in Artificial Intelligence*, 6(60), 2002. 49

[92] Shane T. Jensen and Jun S. Liu. Bayesian Clustering of Transcription Factor Binding Motifs. *Journal of the American Statistical Association*, 103(481) :188–200, 2008. 72, 145

[93] Ian Jolliffe. *Principal Component Analysis*. John wiley edition, 2002. 81

[94] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1) :359–392, 1998. 87, 88, 99, 138

[95] George Karypis and Vipin Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs. *Journal of Parallel and Distributed Computing*, 48(1) :96–129, 1998. 87

[96] Brian W. Kernighan and Shen Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49 :291–307, 1970. 87

[97] Kristian Kersting, Luc De Raedt, and Stefan Kramer. Interpreting Bayesian logic programs. In *Proceedings of the AAAI-2000 workshop on learning statistical models from relational data*, pages 29–35, 2000. 52

[98] Hyunsoo Kim, Haesun Park, and Lars Eldén. Non-negative tensor factorization based on alternating large-scale non-negativity-constrained least squares. In *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering*, pages 1147–1151, 2007. 86

[99] Jin H. Kim and Judea Pearl. A Computational Model for Causal and Diagnostic Reasoning in Inference Systems. In *Proceedings of the 8th international joint conference on Artificial intelligence*, pages 190–193, 1983. 35

[100] Angelika Kimmig, Lilyana Mihalkova, and Lise Getoor. Lifted Graphical Models : A Survey. *Machine learning*, 99(1) :1–45, 2015. 23, 30, 52, 147

[101] S Kirkpatrick, C D Gelatt, and M P Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220(4598) :671–680, 1983. 138

[102] Stanley Kok. The Alchemy System for Statistical Relational AI : User Manual. Technical Report 2005, 2007. 158

[103] Stanley Kok and Pedro Domingos. Learning Markov logic network structure via hypergraph lifting. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 505–512, 2009. 147, 185

[104] Stanley Kok and Pedro Domingos. Learning Markov logic networks using structural motifs. In *Proceedings of the 27th Annual International Conference on Machine Learning (ICML)*, pages 551–558, 2010. 147, 185

[105] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models : Principles and Techniques*. MIT Press, 2009. 23

[106] Daphne Koller and Avi Pfeffer. Object-oriented Bayesian networks. *Proceedings of the 13th conference on Uncertainty in artificial intelligence*, pages 302–313, 1997. 30, 39, 40, 41, 157

[107] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proceedings of the 5th ACM international conference on Web search and data mining (WSDM)*, pages 173–182. ACM Press, 2012. 23

[108] Jeremy Kubica, David Cohn, and Jeff Schneider. cGraph : A Fast Graph-Based Method for Link Analysis and Queries. In *Proceedings of the 2003 IJCAI Text-Mining & Link-Analysis Workshop*, volume 1, pages 22–31, 2003. 146

[109] Jeremy Kubica, Andrew Moore, Jeff Schneider, and Yiming Yang. Stochastic Link and Group Detection. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 798–804, 2002. 146

[110] Vincent Labatut, Nicolas Dugué, and Anthony Perez. Identifying the community roles of social capitalists in the Twitter network. In *IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pages 371–374, 2014. 134, 185

[111] Helge Langseth and Olav Bangsø. Parameter learning in object-oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1-4) :221–243, 2001. 41

[112] Pedro Larrañaga, Hossein Karshenas, Concha Bielza, and Roberto Santana. A review on evolutionary algorithms in Bayesian network learning and inference tasks. *Information Sciences*, 233 :109–125, 2013. 35

[113] Steffen Lauritzen and David Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2) :157–224, 1988. 35

[114] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755) :788–791, 1999. 74, 80, 82, 83

[115] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, pages 556–562, 2001. 80, 83

[116] Mihee Lee, Haipeng Shen, Jianhua Z. Huang, and J. S. Marron. Biclustering via Sparse Singular Value Decomposition. *Biometrics*, 66(4) :1087–1095, 2010. 80

[117] Tom Leighton and Satish Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 422–431, 1988. 86

[118] Stan Z. Li, Xin Wen Hou, HongJiang Zhang, and QianSheng Cheng. Learning spatially localized, parts-based representation. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, pages 207–212, 2001. 80

[119] Chih-jen Lin. Projected Gradient Methods for Non-negative Matrix Factorization. *Neural computation*, 19(10) :2756–2779, 2007. 100

[120] Xin Liu, Weichu Liu, Tsuyoshi Murata, and Ken Wakita. A framework for community detection in heterogeneous multi-relational networks. *Advances in Complex Systems*, 17(6) :1–27, July 2014. 136

[121] Xin Liu and Tsuyoshi Murata. Community Detection in Large-Scale Bipartite Networks. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 50–57, 2009. 93, 151, 154

[122] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2) :129–137, March 1982. 100

[123] Bo Long, Zhongfei Mark Zhang, and Philip S. Yu. A probabilistic framework for relational clustering. In *Proceedings of the 13th international conference on Knowledge discovery and data mining*, pages 470–480. ACM Press, 2007. 23

[124] László Lovász. Random walks on graphs : A survey. *Combinatorics : Paul Erdös is Eighty*, 2(1) :353–397, 1993. 88

[125] Qing Lu and Lise Getoor. Link-based classification. *Proceedings of the 20th International Conference on Machine Learning ICML*, 3 :496–503, 2003. 23, 39

[126] Suzanne M. Mahoney and Kathryn B. Laskey. Network Engineering for Complex Belief Networks. In *Proceedings of the 12th international Conference on Uncertainty in Artificial Intelligence*, pages 389–396, 1996. 39

[127] Marc Maier, Katerina Marazopoulou, David Arbour, and David Jensen. A Sound and Complete Algorithm for Learning Causal Models from Relational Data. In *arXiv preprint arXiv :1309.6843.*, 2013. 52, 185

[128] Andrew McCallum, Karl Schultz, and Singh Singh. Factorie : Probabilistic programming via imperatively defined factor graphs. *Advances in Neural Information Processing Systems*, 22 :1249–1257, 2009. 158

[129] Peter McCullagh and Jie Yang. Stochastic classification models. *International Congress of Mathematicians*, 3 :669–686, 2006. 72, 145

[130] Lilyana Mihalkova, Walaa E. Moustafa, and Lise Getoor. Learning to predict web collaborations. In *Workshop on User Modeling for Web Applications*, 2011. 30

[131] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG : Probabilistic Models with Unknown Objects. In *Introduction to Statistical Relational Learning*, chapter 13, pages 373–398. 2007. 52, 157

[132] Kurt T. Miller, Thomas L. Griffiths, and Michael I. Jordan. Nonparametric Latent Feature Models for Link Prediction. *Advances in Neural Information Processing Systems*, 10(1) :1–9, 2009. 23

[133] Robert J. Mokken. Cliques, clubs and clans. *Quality & Quantity*, 13(2) :161–173, 1979. 89

[134] Tsuyoshi Murata. Detecting communities from bipartite networks based on bipartite modularities. *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering*, 4(3) :50–57, 2009. 74, 92, 93, 152, 183

[135] Aniruddh Nath and Pedro Domingos. Learning Multiple Hierarchical Relational Clusterings. In *ICML-12 Workshop on Statistical Relational Learning*, 2012. 30

[136] Aniruddh Nath and Pedro Domingos. Learning tractable statistical relational models. In *Workshops at the 28th AAAI Conference on Artificial Intelligence*, 2014. 52

[137] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proceedings of the AAAI 2000 Workshop on Learning Statistical Models from Relational Data*, pages 42–49. AAAI Press, 2000. 23, 30, 39

[138] Jennifer Neville and David Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the 5th International Conference on Data Mining ICDM*, pages 322–329, 2005. 146

[139] Jennifer Neville and David Jensen. Relational dependency networks. *The Journal of Machine Learning Research*, 8 :653–692, 2007. 24, 52, 157, 158

[140] Jennifer Neville, David Jensen, Lisa Friedland, and Michael Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 625–630. ACM Press, 2003. 157, 158

[141] Jennifer Neville, David Jensen, and Brian Gallagher. Simple estimators for relational Bayesian classifiers. In *Proceedings of the 3rd International Conference on Data Mining ICDM*, pages 609–612, 2003. 157, 158

[142] Mark E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70(5) :056131, 2004. 90

[143] Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), 2006. 91

[144] Mark E. J. Newman. Community detection and graph partitioning. *EPL (Europhysics Letters)*, 103(2) :28003, May 2013. 90

[145] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004. 89, 90

[146] Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical review. E*, 64(2) :026118, 2001. 92, 135

[147] Jack Newton. *Hierarchical Probabilistic Relational Models for Recommender Systems*. PhD thesis, 2005. 76

[148] Tore Opsahl. Triadic closure in two-mode networks : Redefining the global and local clustering coefficients. *Social Networks*, 35(2) :159–167, 2013. 92, 135

[149] Vitaly Osipov and Peter Sanders. n-Level graph partitioning. In *Algorithms ESA 2010*, pages 278–289, 2010. 87

[150] Pentti Paatero and Unto Tapper. Positive matrix factorization : A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2) :111–126, 1994. 80, 82

[151] Sayan Pathak, David Haynor, Christopher Lau, and Michael Hawrylycz. Non-negative matrix factorization framework for dimensionality reduction and unsupervised clustering. *The Insight Journal*, 2007. 112

[152] Judea Pearl. Probabilistic Reasoning in Intelligent Systems. 1988. *San Mateo, CA : Kaufmann*. 23, 33, 34, 48, 52, 157

[153] Judea Pearl. Reverend Bayes on inference engines : A distributed hierarchical approach. In *Proceedings of the AAAI National Conference on AI*, pages 133–136, 1982. 35

[154] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011. 100

[155] Jim Pitman. Some Developments of the Blackwell-Macqueen Urn Scheme. *Lecture Notes-Monograph Series*, 30 :245–267, 1996. 72

[156] David Poole. First-order probabilistic inference. In *Proceedings of the 18th international Joint Conference on Artificial Intelligence*, pages 985–991, 2003. 49

[157] Alexandrin Popescul and Lyle H. Ungar. Cluster-based concept invention for statistical relational learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 665–670. Acm, 2004. 30

[158] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9) :2658–2663, 2004. 89

[159] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), 2007. 93

[160] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2) :107–136, January 2006. 24, 52, 157, 158, 185

[161] Stefan Schamberger and Jens-Michael Wierum. A Locality Preserving Graph Ordering Approach for Implicit Partitioning : Graph-Filling Curves. In *Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems*, pages 51–57, 2004. 87

[162] Kirk Schloegel, George Karyis, and Vipin Kumar. Graph Partitioning for High-Performance Scientific Simulations. In *Sourcebook of parallel computing*, pages 491–541. 2000. 87

[163] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2) :461–464, 1978. 38

[164] Stephen B. Seidman and Brian L. Foster. A graph theoretic generalization of the clique concept. *The Journal of Mathematical Sociology*, 6(1) :139–154, 1978. 89

[165] Fanhua Shang, L.C. Jiao, and Fei Wang. Graph dual regularization non-negative matrix factorization for co-clustering. *Pattern Recognition*, 45(6) :2237–2250, June 2012. 84, 94, 99

[166] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8) :888–905, 2000. 86, 100

[167] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *Proceedings of the 6th International Conference on Data Mining (ICDM)*, pages 572–582, 2006. 30

[168] Parag Singla and Pedro Domingos. Lifted First-Order Belief Propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, volume 2, pages 1094–1099, 2008. 49, 178

[169] Parag Singla, Aniruddh Nath, and Pedro Domingos. Approximate Lifted Belief Propagation. *Statistical Relational Artificial Intelligence*, pages 92–97, 2010. 49

[170] F. J. Solis and R. J.-B. Wets. Minimization by Random Search Techniques, 1981. 138

[171] Peter Spirtes, Clark Glymour, and Richard Scheines. From probability to causality. *Philosophical Studies*, 64(1) :1–36, 1991. 36

[172] Richard C. Sprinthall. *Basic Statistical Analysis*. Pearson ed edition, 2011. 112

[173] Martin A. Tanner and Wing Hung Wong. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398) :528–540, 1987. 53

[174] Ben Taskar, Pieter Abbeel, Ming-Fai Wong, and Daphne Koller. Relational Markov Networks. In *Introduction to Statistical Relational Learning*, chapter 6, pages 175–200. 2007. 52

[175] Ben Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence IJCAI*, pages 870–876, 2001. 30, 137

[176] Ben Taskar, Ming-Fai Wong, Abbeel Pieter, and Daphne Koller. Link Prediction in Relational Data. In *Advances in Neural Information Processing Systems*, pages 659–666, 2004. 30

[177] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1) :31–78, 2006. 51, 184

[178] Ledyard R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to Mathematical Psychology*, pages 110–127. New York, New York, USA, 1964. 86

[179] Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3) :279–311, 1966. 86

[180] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 220–227, 1990. 36

[181] Chris H. Walshaw, Mark Cross, and Martin G. Everett. A Localized Algorithm for Optimizing Unstructured Mesh Partitions. *International Journal of High Performance Computing Applications*, 9(4) :280–295, 1995. 87

[182] Hua Wang, Heng Huang, and Chris Ding. Simultaneous clustering of multi-type relational data via symmetric nonnegative matrix tri-factorization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 279–284, 2011. 85, 147, 186

[183] Hua Wang, Feiping Nie, Heng Huang, and Chris Ding. Nonnegative matrix tri-factorization based high-order co-clustering and its fast implementation. *Proceedings of the 11th International Conference on Data Mining (ICDM)*, pages 774–783, December 2011. 85, 116, 119

[184] Bo Wu and Ram Nevatia. Cluster boosted tree classifier for multi-view, multi-pose object detection. In *Proceedings of the 11th IEEE International Conference on Computer Vision*, pages 1–8, 2007. 136

[185] Yang Xiang, David Poole, and Michael P. Beddoes. Multiply Sectioned Bayesian Networks and Junction Forests For Large Knowledge-Based Systems. *Computational Intelligence*, 9(2) :171–220, 1993. 40

[186] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 267–273, 2003. 80

[187] Nevin Lianwen Zhang and David Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5 :301–328, 1996. 35

l'université nantes angers le mans
COMMUNAUTÉ D'UNIVERSITÉS ET ÉTABLISSEMENTS

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Anthony COUTANT

Modèles Relationnels Probabilistes et Incertitude de Références

Apprentissage de structure avec algorithmes de partitionnement

Probabilistic Relational Models and Reference Uncertainty

Structure learning with clustering algorithms

Résumé

Nous sommes entourés de données hétérogènes et interdépendantes. L'hypothèse i.i.d. a montré ses limites dans les algorithmes considérant des jeux de données tabulaires, constitués d'individus dotés du même domaine de définition et sans influence mutuelle. L'apprentissage relationnel statistique a pour objectif la représentation de connaissances, le raisonnement et l'apprentissage dans des contextes de jeux de données multi relationnels avec incertitude et les modèles graphiques probabilistes de second ordre sont une solution pour l'apprentissage génératif dans ce contexte. Nous étudions dans cette thèse un type de modèles graphiques probabilistes de second ordre dirigés, appelés *modèles relationnels probabilistes*, dans un contexte d'*incertitude de références*, c.-à-d. où les individus d'un jeu de données peuvent présenter à la fois une incertitude sur la valeurs de leurs attributs descriptifs, et sur leurs implications dans des associations avec d'autres individus, et ayant la particularité de s'appuyer sur des fonctions de partitionnement des individus pour découvrir des connaissances générales. Nous présentons les limites des modèles existant pour l'apprentissage dans ce contexte et proposons des extensions présentant l'intérêt de pouvoir utiliser des méthodes de partitionnement relationnel, plus adaptées au problème, et proposant un biais de représentation simplifié autorisant la découverte de connaissances supplémentaires, notamment entre les différents types d'association du domaine de définition relationnel.

Mots clés

Apprentissage, Modèles Relationnels Probabilistes, Partitionnement, Incertitude de Références, Paradigme Multi-Relationnel.

Abstract

We are surrounded by heterogeneous and interdependent data. The i.i.d. assumption has shown its limits in the algorithms considering tabular datasets, containing individuals with same data domain and without mutual influence on each other. Statistical relational learning aims at representing knowledge, reasoning, and learning in multi-relational datasets with uncertainty and lifted probabilistic graphical models offer a solution for generative learning in this context. We study in this thesis a type of directed lifted graphical model, called *probabilistic relational models*, in the context of *reference uncertainty*, i.e. where dataset's individuals can have uncertainty over both their internal attributes description and their external memberships in associations with others, having the particularity of relying on individuals partitioning functions in order to find out general knowledge. We show existing models' limits for learning in this context and propose extensions allowing to use relational clustering methods, more adequate for the problem, and offering a less constrained representation bias permitting extra knowledge discovery, especially between associations types in the relational data domain.

Key Words

Learning, Probabilistic Relational Models, Clustering, Reference Uncertainty, Multi-Relational Paradigm