

Étude Comparative des outils manipulant les réseaux bayésiens

—
Délivrable n°7

Sourour AMMAR – Sourour.Ammar@etu.univ-nantes.fr

LITIS – INSA Rouen

LINA – Université de Nantes

Philippe LERAY – Philippe.Leray@univ-nantes.fr

LINA – Université de Nantes

Table des matières

I	Etude bibliographique	6
1	Les réseaux bayésiens	7
1.1	Définition	7
1.2	Inférence	9
1.3	Apprentissage des paramètres	10
1.4	Apprentissage de structure	11
1.5	Conclusion	12
II	Etude comparative : choix d'un outil de programmation	13
2	Comparaison des librairies	14
2.1	BNJ	14
2.2	JavaBayes	16
2.3	PNL	16
2.4	ProBT	18
2.5	Récapitulatif	19
3	Comparaison entre PNL et ProBT	21
3.1	Protocole de comparaison	21
3.2	Expérience avec PNL	21
3.3	Expérience avec ProBT	22
3.4	Conclusions et choix	23
A	Apprentissage de structure des réseaux bayésiens	29
A.1	Introduction	29
A.2	Apprentissage de structure à partir de données complètes	29
A.2.1	Recherche des indépendances conditionnelles : algorithmes IC et PC	30
A.2.2	Notion de score	32

La recherche gloutonne	34
Recherche dans l'espace des arbres	37
Recherche dans l'espace des représentants des classes d'équi- valence de Markov	37
Algorithme K2	38
A.3 Apprentissage de structure à partir de données incomplètes	38
A.3.1 Algorithme Expectation-Maximisation	39
A.3.2 Méthodes basées sur un score avec données incomplètes	40
A.4 Conclusion	43

Introduction

De nos jours, et avec l'ouverture de plus en plus importante des systèmes d'informations sur les réseaux de grande échelle tel que Internet, la sécurité informatique s'avère un besoin de pointe. L'objectif principal de la sécurité des systèmes d'information est de concevoir et développer des systèmes d'information dont le fonctionnement se conforme à la spécification d'une politique de sécurité (ensemble de règles qui définissent les autorisations, les interdictions et les obligations des sujets qui peuvent accéder au système d'information). Un attaquant (également appelé intrus) correspond à un agent malveillant qui essaye de porter atteinte à la politique de sécurité. Une attaque est donc une tentative délibérée de violer la politique de sécurité.

Les approches classiques de la sécurité des systèmes d'informations sont fondées sur la définition et la mise en oeuvre de moyens pour protéger le système contre les attaques. Il s'agit par exemple d'implanter des mécanismes d'authentification, de contrôle d'accès ou des protocoles cryptographiques pour empêcher l'occurrence d'attaques tel que les pare-feus. Toutefois, ces pare-feus ne fournissent pas de solution satisfaisante pour certains cas.

La détection d'intrusions constitue une approche attractive dans le domaine de la sécurité informatique. L'objectif n'est plus d'empêcher l'occurrence d'attaques mais de surveiller l'activité d'un système d'information afin d'y détecter des activités malveillantes de manière à préparer la contre-mesure la plus adaptée. La détection d'intrusion présente l'avantage sur les approches classiques fondées sur la protection de pouvoir s'adapter plus rapidement aux nouvelles attaques. La détection d'intrusion est un domaine de recherche actif depuis une vingtaine d'années.

Dans cette thématique, le projet PLACID (Modèles graphiques probabilistes et logiques de descriptions pour la corrélation d'alertes en détection d'intrusion) est lancé. Ce projet, sélectionné lors de l'appel à projet SetIn 2006 (programme "Sécurité et Informatique") lancé par l'Agence National de la Recherche, vise entre autres, à proposer un système de corrélation d'alertes basé sur le formalisme des réseaux bayésiens causaux.

Le laboratoire LITIS est l'un des trois partenaires du projet PLACID. Dans ce cadre, j'étais amené à effectuer cette première étude afin de faire un état des lieux des librairies "réseaux bayésiens" libres en C++ ou Java, et choisir la meilleure permettant de manipuler les réseaux bayésiens.

Nous nous intéresserons en particulier à l'apprentissage de structure des réseaux bayésiens à partir de bases d'observations. Cet intérêt revient à l'aspect que prennent les systèmes réels. Ces systèmes, notamment les systèmes informatiques, sont complexes et de grande taille, il est donc très difficile de demander à un expert de fixer

toutes les relations entre les différentes parties (variables) du système.

En plus, généralement, pour un cas de figure, seules des observations sont disponibles pour un tel système. L'apprentissage de structure de ces modèles graphiques à partir de données peut être un refuge intéressant pour résoudre ces problèmes.

Ce rapport est organisé autour de deux parties, nous essayerons dans la première de présenter la notion des réseaux bayésiens ainsi que le principe de raisonnement de ces modèles. La deuxième partie sera consacrée à la présentation d'une étude comparative de différents outils manipulant les réseaux bayésiens. Nous finirons cette comparaison par choisir l'outil qui nous semblera le mieux adapté à nos besoins.

Nous finirons par présenter alors quelques conclusions.

En annexe, nous présentons d'une manière détaillée la notion d'apprentissage de structure des réseaux bayésiens à partir de masses de données.

Première partie
Etude bibliographique

Chapitre 1

Les réseaux bayésiens

Avec la croissance de la taille des bases de données dans notre époque est née la nécessité d'automatiser le traitement de cette grande masse de données, automatiser le raisonnement et la prise de décision à partir de ces masses énormes. Il serait donc intéressant d'avoir un ou plusieurs systèmes permettant de faire le lien entre les observations et la réalité pour un objectif précis (aide à la prise de décision), et cela, même lorsque les observations sont incomplètes et/ou imprécises. Les réseaux bayésiens (RB) apportent des solutions efficaces à ces insuffisances par leurs représentations graphiques compactes des problèmes réels complexes et leur rapidité en temps de calcul. En effet, l'utilisation des réseaux bayésiens s'est répandue sur plusieurs domaines d'applications dont nous citons : la détection d'intrusions [21][9], les systèmes de diagnostic, filtres de spams [34], etc.

Nous commençons par introduire la notion des réseaux bayésiens ainsi que leur principe de raisonnement. Nous mettrons l'accent en particulier sur les principaux critères d'étude dans les outils de développement.

1.1 Définition

Les réseaux bayésiens, initiés par [27], sont des *modèles graphiques* qui représentent les relations probabilisées entre un ensemble de variables.

Un réseau bayésien $B = (G, \theta)$ est défini par :

- L'ensemble des variables aléatoires observables $X = \{X_1, \dots, X_n\}$.
- $G = (X, E)$, graphe dirigé sans circuit (DAG), où chaque noeud est associé à une variable de X .
- $\theta = \{\theta_i\} = \{P(X_i | Pa(X_i))\}$, ensemble des distributions de probabilités de chaque noeud X_i conditionnellement à ses parents immédiats dans le graphe G .

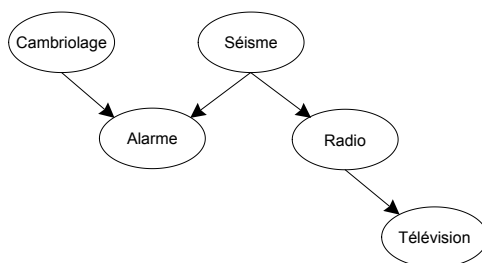
Les variables aléatoires peuvent avoir des distributions discrètes comme elles peuvent être à distributions continues. Le traitement de ces deux types de variables ne se fait pas de la même façon, donc la manipulation des variables discrètes (*critère : Variables discrètes*) et continues (*critère : Variables continues*) constitue un critère d'étude important pour les outils de développement et de manipulations des réseaux bayésiens.

Dans le cas discret, un paramètre θ_i est un tableau contenant l'ensemble des probabilités de la variable X_i pour chacune de ses valeurs possibles sachant chacune des valeurs prises par l'ensemble de ses parents $Pa(X_i)$ dans le graphe G .

Le graphe d'un réseau bayésien permet ainsi de représenter d'une façon visuelle les relations (dépendances et indépendances) entre les variables du système. Les distributions de probabilités permettent d'enrichir cette structure graphique par une quantification de ces relations. Les probabilités dans un réseau bayésien permettent de représenter l'aspect incertain qui relie les variables.

Le graphe d'un réseau bayésien peut être statique (*critère : Graphes statiques*) ne dépendant pas du temps, comme il peut être dynamique (*critère : Graphes dynamiques*) dont la structure varie en fonction du temps.

La figure 1.1 est un exemple de réseau bayésien modélisant la probabilité de déclencher une alarme suite aux deux causes possibles : cambriolage et séisme. Il modélise aussi la probabilité que la radio et la télévision annoncent un séisme sachant l'état vrai ou faux (incertain) de la variable "séisme". Donc cet exemple modélise les relations qui relient les variables : le cambriolage et le séisme provoquent le déclenchement de l'alarme, un séisme entraîne l'annonce de cet événement à la télévision et par la suite à la radio...



TAB. 1.1 – Exemple de réseau bayésien

Les réseaux bayésiens permettent de représenter de manière compacte la distribution de probabilité jointe sur l'ensemble des variables :

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | Pa(X_i)) \quad (1.1)$$

Cette décomposition de la loi jointe permet de faire des réseaux bayésiens un modèle économique pour représenter des distributions de probabilités. Ces simplifications apportées par cette loi jointe ont permis de rendre l'apparition des algorithmes d'inférence. Ces algorithmes permettent, à partir de la structure d'un réseau bayésien et des distributions des probabilités associées, de calculer la probabilité de n'importe quelle variable du modèle à partir de l'observation même partielle des autres variables.

1.2 Inférence

Supposons que nous disposions d'un réseau bayésien défini par un graphe et la distribution de probabilité associée. Le problème général de l'inférence est de calculer $P(X_i | Y)$, avec $Y \subset X$ et $X_i \notin Y$, c'est à dire la probabilité du noeud X_i sachant qu'on a observé l'ensemble des variables Y .

Les algorithmes d'inférences utilisent la notion de probabilités conditionnelles ainsi que le théorème de Bayes.

Plusieurs algorithmes d'inférence ont été développés. Certains utilisent des méthodes d'inférence exactes (*critère : Inférence exacte*), d'autres des méthodes d'inférence approchées (*critère : Inférence approchée*).

Parmi les méthodes d'inférence exactes citons la méthode "Messages locaux" [27] [19], "Ensemble de coupes" [28], "Arbre de jonction" [22] [18], "Inversion d'arcs" [35] et "Élimination de variables" [39].

Lorsque la dimension des réseaux bayésiens augmente, le temps de calcul est de plus en plus important. Lorsque les tables de probabilités conditionnelles sont issues de données (par apprentissage), ces tables ne sont pas exactes. Dans ce cas il est intéressant de ne pas perdre du temps en faisant de l'inférence exacte sur des probabilités non exactes, donc le recours aux méthodes d'inférence approchée.

Parmi les méthodes d'inférence approchées existent les méthodes basées sur la "simulation stochastique par Chaîne de Monte-Carlo" [14], "Loopy belief propagation" [29], les méthodes variationnelles [23], les méthodes de recherche de masse [17] [32], d'autres sont basées sur la simplification du réseau [20].

Les deux familles de méthodes d'inférence présentées précédemment s'appliquent sur les réseaux bayésiens une fois que la structure et les tables de probabilités de chaque variable sont connues. Or, dans la réalité ce n'est pas toujours le cas. Parfois, un expert arrive à fixer la structure du réseau bayésien modélisant le problème réel, mais les tables de probabilités restent inconnues. D'autres situations peuvent être réelles aussi, dans le cas où même la structure du réseau bayésien n'est pas connue, et seul un ensemble de données (observations réelles) est disponible pour le problème à traiter. Dans ce cas, un apprentissage est nécessaire. Deux sortes d'apprentissages peuvent être envisagés :

- L'apprentissage des paramètres, où nous supposons que la structure du réseau a été fixée, et où il faudra estimer les probabilités conditionnelles de chaque noeud du réseau.
- L'apprentissage de structure, où le but est de trouver le meilleur graphe représentant la tâche à résoudre.

1.3 Apprentissage des paramètres

L'apprentissage des paramètres d'un réseau bayésien se fait à partir de données relatives au problème à modéliser. Toutefois, ces données peuvent être complètes ou incomplètes. Les algorithmes d'apprentissage des paramètres ne sont pas les mêmes dans ces deux cas.

Dans le cas où toutes les variables sont observées (*critère : Apprentissage de paramètres données complètes*), la méthode la plus simple et la plus utilisée est l'estimation statistique de la probabilité d'un événement par la fréquence d'apparition de l'évènement dans la base de données. Cette méthode est appelée maximum de vraisemblance.

Une autre méthode dite "estimation bayésienne" est aussi utilisée. Elle suit un principe différent. Elle consiste à trouver les paramètres les plus probables sachant que les données ont été observées, en utilisant des a priori sur les paramètres.

Dans le cas où les données ne sont pas complètes (*critère : Apprentissage de paramètres données incomplètes*), la méthode d'estimation des paramètres couramment utilisée est fondée sur l'algorithme itératif EM (Expectation Maximisation) dont nous détaillons le principe dans la partie A.3.1.

1.4 Apprentissage de structure

Lorsque la structure du réseau bayésien n'est pas fournie a priori par un expert, il est possible d'apprendre cette structure à partir d'une base de données. Toutefois, dans la réalité cette base de données peut présenter un manque d'informations pour certaines mesures.

Beaucoup de travaux [24] [12] sont intéressés par ce genre de problèmes et la recherche d'éventuelles solutions. Des approches ont été proposées pour l'apprentissage de structure pour chacun des cas : données complètes ou incomplètes.

Dans le cas où notre base de données est complète (*critère : Apprentissage de structure : données complètes*), c'est-à-dire toutes les mesures sont complètes, deux familles à approches ont été proposées.

La première famille (algorithmes IC, PC etc.) consiste à déterminer dans un premier temps un graphe non orienté en tenant compte des différentes indépendances conditionnelles qui existent entre les variables de ce graphe, puis à orienter ce graphe pour obtenir un réseau bayésien. Ces algorithmes sont peu efficaces dans le cas de problèmes de grande taille puisque la détermination de ces indépendances est exponentielle en fonction du nombre des variables.

La deuxième approche consiste à parcourir tous les graphes possibles, associer un score à chaque graphe, puis choisir le graphe ayant le score le plus élevé. Toutefois, cette méthode n'est pas simple, principalement à cause de la taille super-exponentielle de l'espace de recherche en fonction du nombre de variables. Robinson a montré dans [33] que le nombre $r(n)$ de structures différentes des graphes possibles pour n variables est donné par la formule A.1 :

$$r(n) = \sum_{i=1}^n (-1)^{i+1} \binom{n}{i} 2^{i(n-1)} r(n-i) = n^{2^{O(n)}} \quad (1.2)$$

Des idées ont été proposées pour résoudre ce problème. Une première consiste à remplacer l'espace de recherche (espace des réseaux bayésiens) par un espace plus petit, l'espace des arbres (MWST ou Maximal Weight Spanning Tree). Une deuxième idée consiste à ordonner les noeuds pour limiter la recherche des parents possibles pour chaque noeud (algorithme K2). Une troisième consiste à faire une recherche gloutonne dans l'espace des réseaux bayésiens (algorithme GS).

Il existe un autre espace dans lequel nous pouvons apprendre les indépendances de la loi de probabilité : l'espace des représentants des classes d'équivalence de Markov. Cet espace est équivalent à l'espace des réseaux bayésiens sauf qu'il ne contient

que des exemplaires des classes d'équivalence de Markov, donc un peu plus petit.

En réalité, plusieurs bases de données présentent des manques de mesures (*critère : Apprentissage de structure données incomplètes*). Ce manque de données revient à plusieurs raisons. Par exemple, un instrument de mesure tombe en panne pour une période ce qui se traduit par un manque de données pour toutes les séries de mesures durant cette période, le système entre dans un état particulier imprévu alors il répond aléatoirement par l'une des réponses possibles ou il se bloque, ou encore un responsable de la prise de mesures oublie de le faire pour une cause ou une autre (fatigue, sommeil, mesures variant très rapidement ...), ou que la donnée reportée ne soit pas lisible.

Quand nous voulons modéliser le comportement de notre système à partir de cette base de données, nous aurons un problème causé par ces manques de mesures. La première idée qui nous vient à la tête est de filtrer les données et ne prendre que les exemples complètement observés de la base, sauf que nous ne disposons plus de la même quantité de données pour faire l'apprentissage et nous risquons de ne plus avoir qu'un nombre très petit de données dans la base lorsque le taux de valeurs manquantes est élevé. L'idée la plus pertinente est de compléter les données manquantes en utilisant une technique de substitution par la moyenne ou encore la méthode de maximum de vraisemblance (*algorithme EM*).

Afin de faire l'apprentissage de structure de réseau bayésien, les deux approches proposées dans le cas de données complètes (approche statistique et approche basée sur un score) sont aussi valables dans le cas de données incomplètes mais avec des rectifications.

L'annexe A.16 présente une étude plus détaillée pour l'apprentissage de structure de réseau bayésien à partir de données complètes ainsi qu'à partir de données incomplètes.

1.5 Conclusion

Les réseaux bayésiens demeurent un outil puissant dans la modélisation de problèmes complexes et le raisonnement à partir de l'incertain. Nous avons introduit dans ce chapitre la notion de ces modèles graphiques probabilistes ainsi que leur principe de raisonnement. Nous avons précisé également un ensemble de critères qui nous aideront à choisir un bon outil de développement de ces modèles graphiques. Nous présentons dans la partie suivante l'étude comparative que nous avons effectué pour choisir cet outil.

Deuxième partie

Etude comparative : choix d'un outil de programmation

Chapitre 2

Comparaison des bibliothèques

Pour la manipulation et la programmation des algorithmes traitant les réseaux bayésiens, plusieurs bibliothèques ont été initiées. Nous citons la bibliothèque BNT^{1,2} (Bayes Net Toolbox) sur laquelle plusieurs travaux de recherche ont été effectués au sein du laboratoire LITIS. Nous pouvons utiliser cette bibliothèque comme référence pour notre travail vue sa richesse en terme d'algorithmes traitant les réseaux bayésiens. D'autres travaux ont donné le jour à d'autres bibliothèques manipulant les modèles graphiques probabilistes (BNJ, JavaBayes, PNL, ProBT).

Le but de cette partie est, dans un premier temps, d'étudier ces bibliothèques : voir ce que fournit chacune d'elles, les algorithmes implémentés, les types de données supportés ainsi que les facilités proposées pour les nouveaux développeurs. Ensuite, nous allons comparer ces bibliothèques et en choisir une avec laquelle nous pouvons manipuler les réseaux bayésiens.

2.1 BNJ

BNJ³ (*Bayesian network tools in Java*) est un ensemble d'outils *Java* de recherche et de développement des réseaux bayésiens. Ce projet a été développé au sein du laboratoire KDD de l'université du Kensas. C'est un projet Open source distribué sous la licence GNU (General Public Licence). Sa dernière version 3.3+ a été publiée en Avril 2006. Cette version fournit une interface graphique qui facilite la création, la modification, l'importation et l'exportation des réseaux bayésiens. Elle fournit aussi un ensemble d'algorithmes d'inférence pour les réseaux bayésiens.

Nous détaillons dans la suite le contenu de cette boîte à outils dans sa version 3.3+.

¹<http://bnt.insa-rouen.fr/>

²<http://bnt.sourceforge.net/>

³<http://bnj.sourceforge.net/>

Pour la définition des réseaux bayésiens dans l’environnement de BNJ v3.3+, l’utilisateur peut utiliser l’interface graphique de ce système. Il est possible de définir deux types de distribution de probabilité pour les noeuds : distribution tabulaire discrète et distribution continue. Les réseaux bayésiens créés sont stockés dans des fichiers xml.

Algorithmes d’inférence : BNJ v3.3+ fournit un ensemble d’algorithmes d’inférence pour les réseaux bayésiens. Ces algorithmes se classent en deux catégories : inférence exacte et inférence approchée.

Les algorithmes d’inférence exacte développés sont "Arbre de Jonction", "Élimination des variables avec optimisation", "Singly-connected network belief propagation" ("Pearl") et "Cutset Conditioning".

Les algorithmes d’inférence approchée développés dans cette boîte à outils sont des méthodes basées sur les algorithmes d’inférence exacte. En effet, certaines méthodes utilisent la notion d’échantillonnage tels que "Adaptive Importance Sampling (AIS)", "Logic Sampling" et "Forward Sampling", d’autres méthodes appliquent les algorithmes d’inférence exacte sur une sélection d’arcs du graphe à traiter tels que "KruskalPolytree", "BCS" et "PTReduction".

Apprentissage : En parcourant l’ensemble des fichiers sources de cette boîte à outils nous ne trouvons pas d’implémentation des algorithmes d’apprentissage ni de paramètres ni de structure.

La librairie BNJ est structurée de façon arborescente facilitant ainsi la navigation dans les différents répertoires. Les fichiers du code source sont bien soignés et présentent des informations utiles pour la compréhension du rôle de chaque fichier. En effet, les différentes méthodes sont décrites par un commentaire, de plus, les variables et les méthodes portent des noms significatifs.

Nous pouvons trouver sur le site de BNJ deux fichiers documentant la version BNJ 2.03a, un destiné aux nouveaux utilisateurs BNJ et un autre destiné aux développeurs qui s’intéressent au code source. Mais la dernière version est fournie sans aucune documentation. Nous trouvons aussi un groupe de discussion. Toutefois, les messages (questions et réponses) sur ce groupe de discussion datent de l’année 2005. Le soin de la structure et du contenu des fichiers sources s’avère alors le seul refuge des nouveaux développeurs.

2.2 JavaBayes

JavaBayes⁴ est un ensemble d'outils *Java* qui permettent de créer et de manipuler des réseaux bayésiens. Il a été développé par *Fabio Gagliardi Cozman* en 1998 et il a été distribué sous la licence GNU. Ce système est composé d'un éditeur graphique, un noyau d'inférence et un ensemble d'analyseurs syntaxiques. L'éditeur graphique permet à son utilisateur de créer et de modifier des réseaux bayésiens. Les analyseurs syntaxiques permettent d'importer et d'exporter des réseaux bayésiens dans différents formats. Le moteur d'inférence est responsable de la manipulation des données. Il calcule les probabilités marginales et les espérances.

Les algorithmes d'inférence implémentés dans ce système sont "Variable Elimination" et "Bucket Tree Elimination". JavaBayes est capable d'utiliser des modèles avec des ensembles de distributions pour calculer les intervalles des distributions à posteriori ou les intervalles des espérances.

Nous remarquons que JavaBayes ne propose pas d'algorithmes pour l'apprentissage de paramètres ou de structure.

La librairie JavaBayes est structurée à base de répertoires contenant chacun les fichiers relatifs à la même fonctionnalité (inférence, ..). Les méthodes dans JavaBayes sont peu commentées et beaucoup variables ont des noms non significatifs ce qui rend la compréhension du code difficile. Sur le site de JavaBayes, nous trouvons de la documentation sous la forme de tutoriels. Ces tutoriels décrivent les étapes pour l'installation, la compilation et la mise en fonction de la librairie. Comme ils expliquent, à travers deux exemples fournis dans la librairie, quelques fonctionnalités (chargement, enregistrement et modification d'un réseau bayésien).

2.3 PNL

PNL⁵ (*Probabilistic Network Library*) est une boîte à outils qui permet la manipulation des modèles graphiques (réseaux bayésiens et chaînes de Markov). Cette librairie est implémentée en C++ et est distribuée sous la licence *Open Source* de Intel. Elle supporte les modèles dirigés et non dirigés, les variables discrètes et continues, comme elle fournit une variété d'algorithmes d'inférence et d'apprentissage. Nous détaillons dans la suite ce que fournit cette librairie.

Contrairement aux deux librairies précédentes, PNL ne fournit pas d'interface graphique pour la création et la visualisation des graphes.

⁴<http://www.cs.cmu.edu/~javabayes/Home/>

⁵<https://sourceforge.net/projects/openpnl/>

Inférence : Les programmeurs de PNL se sont intéressés à deux types de modèles graphiques : les modèles statiques et les modèles dynamiques. En effet, la librairie PNL propose un certain nombre d’algorithmes d’inférence et d’apprentissage pour chacun des deux types.

Les algorithmes d’inférence pour les modèles graphiques statiques se divisent en deux sous catégories : inférence exacte et inférence approchée. Les algorithmes d’inférence exacte dans PNL sont "Inférence naïve", "Arbre de Jonction" et "Pearl Inférence". Les algorithmes d’inférence approchée dans PNL sont "Gibbs Sampling Inférence" et "LW Sampling Inférence". "2TBN Inférence", "1 5Slice Inférence", "1 5Slice JTree Inférence", "BK Inférence" et "2TPF Inférence" sont les algorithmes d’inférence pour les modèles graphiques dynamiques dans PNL.

Apprentissage : Comme c’est le cas pour les algorithmes d’inférence, PNL propose des algorithmes d’apprentissage pour les modèles graphiques statiques ainsi que pour ceux dynamiques. Certains de ces algorithmes sont relatifs à l’apprentissage des paramètres, d’autres sont relatifs à l’apprentissage de la structure. Nous citons l’algorithme "EMLearningEngine" qui se base sur l’algorithme *Expectation-Maximisation* pour l’apprentissage des paramètres dans les réseaux bayésiens dont les noeuds sont discrets ou continus gaussiens et dans les modèles de Markov dont les noeuds sont discrets. Nous citons aussi l’algorithme "BayesLearningEngine" et "BICLearningEngine".

"MIStaticStructLearnHC" et "MWST" (développé pendant mon stage de master) sont deux algorithmes d’apprentissage de la structure d’un réseaux bayésiens dont les données d’entrée sont complètes. PNL propose un algorithme d’apprentissage de structure dans le cas de données incomplètes, cet algorithme (*Structural EM*) se base sur les deux notions *EM* et *Hill-climbing*. Nous avons ajouté dans cette catégorie l’algorithme "MWST-EM".

Pour les modèles dynamiques, nous trouvons un algorithme basé sur l’algorithme *EM* pour l’apprentissage des paramètres et nous trouvons aussi un algorithme basé sur *Hill-climbing* pour l’apprentissage de structure dans le cas de données complètes.

La librairie PNL comprend un grand nombre de classes. Les fichiers de codes sources sont tous mis dans un seul répertoire sans aucune structuration. Le code C++ ne présente pas de commentaires, heureusement les méthodes et les variables portent des noms significatifs.

Le projet PNL comprend, en plus des classes composants la librairie, des exemples d’utilisation de ces classes (exemple de création d’un réseau bayésien, d’invocation des différents moteurs d’inférence et d’apprentissage ...). Ces exemples ne constituent

pas toute la documentation pour la librairie PNL. En effet, ce projet fournit de la documentation sous forme d'un guide d'utilisation et un manuel de références. Ce document est structuré autour de trois parties :

- Une vue d'ensemble pour la librairie.
- Un guide d'utilisation illustrant les différents algorithmes implémentés ainsi que la façon d'interaction entre ces algorithmes via quelques exemples simples.
- Un manuel de références énumérant les différentes classes de la librairie, les méthodes de chaque classe ainsi que leurs rôles et la signification de leurs paramètres. Cette partie présente en plus la structure de la librairie à travers les relations entre les classes (relation d'héritage).

Sur le site de PNL nous trouvons un forum où figure tout l'historique des questions et réponses des anciens membres, organisé par thème. Une fois membre, nous pouvons poser des questions, mais ce n'est pas sûr que nous aurons des réponses, même pas justes, puisque ce sont pas des spécialistes en PNL qui répondent systématiquement. Alors ce forum peut servir dans plusieurs situations mais pas souvent.

2.4 ProBT

ProBT⁶ est une librairie C++ pour la manipulation des réseaux bayésiens. Elle a été initié depuis 10 ans au sein de la société ProBayes⁷.

Les institutions CNRS⁸, INRIA⁹, UJF¹⁰ et INPG¹¹ tiennent tous les droits de propriété sur le logiciel ProBT. L'exploitation commerciale et industrielle de ce logiciel a été concédée d'une façon exclusive à la Société Probayes. La Société Probayes donne son accord au laboratoire GRAVIR pour la distribution libre(gratuite) du code d'objet de la bibliothèque ProBT à la communauté scientifique pour qu'il puisse être utilisé pour la recherche et des buts éducatifs.

La librairie ProBT comprend deux parties : une interface d'application (API) pour la création des réseaux bayésiens, et un moteur d'inférence bayésien (BIE) permettant d'exécuter tout le calcul de probabilité dans les modes exacts ou approximatifs. ProBT supporte les deux types de variables discrètes et continues pour la définition des réseaux bayésiens, comme elle fournit un ensemble d'algorithmes d'inférence : Pour l'inférence exacte, ProBT utilise "l'Algorithme de Restrictions Successives" (SRA pour Successive Restrictions Algorithm). Pour l'inférence approchée, plusieurs

⁶<http://www-laplace.imag.fr/bayesian-programming/index.html>

⁷<http://www.probayes.com/>

⁸Centre National de la Recherche Scientifique

⁹Institut National de Recherche en Informatique et en Automatique

¹⁰Université Joseph Fourier

¹¹Institut National Polytechnique de Grenoble

arrangements d'approximation sont utilisés par ProBT comme Monte-Carlo, l'Évaluation simultanée et la Maximisation.

ProBT propose aussi des algorithmes pour l'apprentissage des paramètres. Dans le cas de données complètes, elle propose un algorithme à base de maximum de vraisemblance et un algorithme basé sur le principe de EM dans le cas de données incomplètes.

ProBT ne contenait pas d'implémentations pour l'apprentissage de structure des réseaux bayésiens. Un travail a été développé (dans le cadre d'un stage de Master, sous la direction de Mr. Philippe LERAY) pour l'implémentation des algorithmes suivants :

- L'algorithme MWST et K2 pour l'apprentissage de structure à partir de données complètes.
- L'algorithme MWST-EM et K2-EM pour l'apprentissage de structure à partir de données incomplètes.

Le code source des différentes classes de la librairie n'étant pas fourni, seules les fichiers d'entête sont consultables. Le contenu de ces fichiers est bien commenté, les noms des méthodes et des variables est significatif.

En plus, une documentation riche est fournie avec la librairie. Nous trouvons un guide d'utilisation aidant à mieux comprendre le contenu de la librairie, un document sur la programmation bayésienne et un manuel citant les différentes classes ainsi que leurs méthodes et leurs paramètres. Nous trouvons aussi sur le site de ProBT un groupe de discussion auquel nous pouvons poser des questions ou même profiter des anciennes discussions dans l'archive.

2.5 Récapitulatif

La figure 2.1 de la page 20 nous présente une comparaison entre le contenu de chacune des librairies étudiées lors de ce travail. Cette étude comparative tient compte des différents critères identifiés dans la partie bibliographique auxquels nous avons ajouté les critères relatifs à la structure et la clarté du code et de la documentation de la librairie (critères : Structure de la librairie, Clarté du code, Documentation et Forums).

Nous constatons que la librairie PNL est plus riche et plus évoluée que les deux autres librairies BNJ et JavaBayes. En effet, PNL, malgré l'absence d'une interface graphique qui permet l'affichage des graphes, propose une richesse par rapport à la librairie BNJ au niveau des algorithmes d'apprentissage et le traitement des graphes dynamiques.

La richesse de BNJ par rapport à la librairie JavaBayes est au niveau des algorithmes

Librairie	JavaBayes	BNJ	PNL	ProBT
Langage de programmation	Java	Java	C++	C++
Licence	GNU	GNU	Open GL	Licence
Interface graphique	+	++	∅	∅
Variables discrètes	+++	+++	+++	+++
Variables continues	∅	++	+++	+++
Graphes statiques	+++	+++	+++	+++
Graphes dynamiques	∅	∅	+++	+++
Inférence exacte	++	+++	+++	+++
Inférence approchée	∅	++	+++	+++
Apprentissage de paramètres : données complètes	∅	∅	+++	+++
Apprentissage de paramètres : données incomplètes	∅	∅	++	++
Apprentissage de structure : données complètes	∅	∅	++	++
Apprentissage de structure : données incomplètes	∅	∅	++	++
Structure de la librairie	+++	+++	++	++
Clarté du code	+	+++	++	+++
Documentation	+	∅	++	+++
Forums	∅	+	++	+++
Note finale	14/45	22/45	35/45	38/45

FIG. 2.1 – Comparaison

d'inférence approchée et le traitement des variables continues.

La librairie ProBT est la librairie ayant la meilleure note en comptant le degré d'intérêt de chacun des critères étudiés, mais pas trop loin de PNL.

Nous concluons alors que PNL et ProBT constituent les librairies les plus proches de répondre à nos attentes. Une étude comparative plus poussées pour ces deux librairies a été effectuée afin d'en choisir une.

Chapitre 3

Comparaison entre PNL et ProBT

Dans le chapitre précédent, nous avons présenté une comparaison de quatre librairies manipulant les réseaux bayésiens. Cette comparaison s'est basée sur un ensemble de critères (contenu en algorithmes et structure). Deux librairies (PNL et ProBT) sont retenues. Nous présentons dans ce chapitre une comparaison plus poussée pour ces deux librairies afin d'en choisir une avec laquelle nous continuerons le travail.

3.1 Protocole de comparaison

Deux stages de Master ont été réalisés, sous la direction de Mr Philippe LERAY, par Cristian BRISCAN et moi. Au cours de mon stage j'ai utilisé la librairie PNL pour développer quelques algorithmes d'apprentissage de structure des réseaux bayésiens. Cristian a programmé des algorithmes d'apprentissage de structure aussi mais en utilisant la librairie ProBT. Ces deux expériences avec les deux librairies nous a permis de mieux les comparer.

Les critères de comparaison les plus importants que nous utilisons sont (1) en premier lieu la qualité du résultat fourni par chacune des deux librairies (comparaison par rapport au résultat obtenu par les algorithmes de la librairie BNT déjà validés), (2) ensuite la facilité et la clarté du code source de la librairie ainsi que la facilité de l'intégration des nouveaux algorithmes dans chacune des librairies, (3) et enfin la documentation fournie et les supports d'aides.

3.2 Expérience avec PNL

Les algorithmes d'apprentissage de structure ajoutés à la librairie PNL sont : l'algorithme MWST dans le cas de données complètes et MWST-EM pour le cas

de données incomplètes. Le développement de ces deux algorithmes a nécessité une phase très importante et coûteuse en temps : la familiarisation avec l'outil de développement et la maîtrise des différents objets de la librairie à utiliser.

Pour ce faire, je me suis basées sur le manuel d'utilisation fourni avec la librairie sauf que cette source de documentation est limitée à quelques exemples de création des réseaux bayésiens et d'invocation des algorithmes d'inférence.

Ils existait aussi un forum associé au projet PNL. Ce forum est doté d'un historique de messages (questions et réponses) des anciens développeur en PNL. Toutefois, ces messages datent d'un certains temps. En plus, lorsque je n'arrive pas à trouver une réponse à ma problématique dans l'historique, et même si je laisse une question sur le forum, ce n'est pas sûr que j'obtient une réponse puisque il n'existe pas de spécialistes en PNL qui gèrent et répondent aux questions du forum.

La documentations pour PNL n'était pas suffisante pour s'intégrer et comprendre l'architecture très complexe de cette librairie. Ce qui m'a pris énormément de temps pour se familiariser avec les différents outils et objets que j'ai utilisé.

Une fois arrivée à mieux comprendre la structure de la librairie, le développement et l'intégration des nouveaux algorithmes n'était pas aussi difficile.

Pour la validation des algorithmes développés, nous avons opté pour une comparaison des résultats de nos algorithmes avec ceux de la librairie BNT dans les mêmes conditions et sur le même jeu de données. Cette comparaison nous a permis de tirer les conclusions : validation des algorithmes de PNL (résultats similaires) et gain en temps de calcul (par rapport à BNT).

3.3 Expérience avec ProBT

La librairie ProBT ne contenait pas d'algorithmes pour l'apprentissage de structure des réseaux bayésiens. Les algorithmes développés sont K2 et MWST pour le cas de données complètes ainsi que K2-EM et MWST-EM pour le cas de données incomplètes.

Le développement de ces algorithmes a nécessité aussi une phase de familiarisation avec la librairie et les différents objets à utiliser. Mais le développement des nouveaux algorithmes a été un peu indépendant de ce qui existait dans la librairie puisqu'il n'y avait pas de composants traitant le problème d'apprentissage de structure de réseau bayésiens dans ProBT. En plus, les responsables de cette librairie ont préféré l'indépendance maximale par rapport à ce existe déjà. Ceci a facilité l'intégration du nouveau code dans la librairie.

Une documentation pour la librairie et pour la programmation bayésienne est fournie avec la librairie ainsi qu'un manuel d'utilisation.

De plus, le forum de discussion sur le site de ProBT permet aux nouveaux développeurs de poser des questions et d'obtenir des réponses rapides et précises puisque des spécialistes de ProBT répondent aux questions. Ceci en plus de la clarté du code et des commentaires qui contiennent les fichiers sources.

Les résultats des algorithmes développés ont été testés et comparés à ceux de la librairie BNT. Cette comparaison a permis de valider les nouveaux algorithmes de ProBT ainsi que de réaliser un temps de calculs très réduit par rapport à celui en BNT vu la manière optimisée avec laquelle le code a été écrit.

3.4 Conclusions et choix

Nous avons présenté dans ce chapitre une comparaison entre la librairie ProBT et PNL suite à deux expériences avec ces deux outils. Certes le résultat des développements réalisés avec les deux librairies dans le cadre de l'apprentissage de structure des réseaux bayésiens est similaire, mais les conditions de travail au sein des deux outils est différent. En effet, le développement en ProBT trouve un soutien important tant de la documentation riche que des membres des forums, contrairement au développement au sein de PNL qui souffre de la manque de tel soutien.

Le facteur documentation s'avère très important à travers les deux expériences vécues avec les deux outils. Pour cela nous décidons de choisir la librairie ProBT pour continuer notre travail.

Conclusion et bibliographie

Conclusion

Cette étude, réalisée dans le cadre du projet PLACID, couvre le domaine du raisonnement à base des réseaux bayésiens et plus précisément l'apprentissage de structure de ces modèles graphiques à partir de masses de données. Elle avait pour but de choisir un outil de programmation pour la manipulation des réseaux bayésiens.

Nous avons introduit dans la première partie de ce rapport la notion des réseaux bayésiens en tant que modèles graphiques probabilistes ainsi que leur principe de raisonnement en général. Dans la suite nous avons présenté en particulier l'apprentissage de structure des réseaux bayésiens à partir de masses de données, ceci dans les deux cas qui peuvent se présenter dans le monde réel : données complètes et incomplètes.

Cette étude bibliographique nous a permis d'explorer le domaine du raisonnement à base de réseaux bayésiens et de choisir deux librairies les plus riches.

La comparaison à partir des deux expériences vécues avec ces deux librairies nous a conduit à effectuer un dernier choix pour une librairie unique, *ProBT*, avec laquelle nous comptons développer nos algorithmes dans le cadre du projet PLACID.

Bibliographie

- [1] M. Beal and Z. Ghahramani. The variational bayesian EM algorithm for incomplete data : with application to scoring graphical model structures. *Bayesian Statistics*, volume 7 :453–464, 2003.
- [2] C. Borgelt and R. Kruse. *Graphical Models - Methods for Data Analysis and Mining*. John Wiley & Sons, Chichester, United Kingdom, 2002.
- [3] C. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14(3) :315–332, 1992.
- [4] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3) :462–467, 1968.
- [5] I. Cohen, A. Bronstein, and F. Cozman. *Adaptive online learning of bayesian network parameters*. Technical Report HPL-2001-156, Hewlett Packard Labs, 2001.
- [6] G. Cooper and E. Hersovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9, 1992.
- [7] D. Dash and M. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *The Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI99)*, 1999.
- [8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 :1–38, 1977.
- [9] A. Faour, P. Leray, and C. Foll. Réseaux bayésiens pour le filtrage d’alarmes dans les systèmes de détection d’intrusion. In *numéro spécial de la revue RNTI (Revue des Nouvelles Technologies de l’Information) ”Extraction des connaissances : Etat et perspectives”*, volume E5, pages 69–72, 2006.
- [10] O. Francois and P. Leray. Apprentissage de structure dans les réseaux bayésiens et données incomplètes. In *5èmes journées d’Extraction et de Gestion des Connaissances (EGC 2005)*, pages 25–33, Paris, France, 2005.

-
- [11] O. François and Ph. Leray. Evaluation d'algorithmes d'apprentissage de structure pour les réseaux bayésiens. In *Le 14ème Congrès Francophone Reconnaissance des Formes et Intelligence Artificielle, RFIA*, pages 1453–1460, 2004.
- [12] Olivier François. *De l'identification de structure de réseaux bayésiens à la reconnaissance de formes à partir d'informations complètes ou incomplètes*. PhD thesis, Institut National des Sciences Appliquées de Rouen, France, 2006.
- [13] N. Friedman. The bayesian structural em algorithm. In *Cooper, G. F. & Moral, S. (Eds.), Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 1998.
- [14] W. Gilks, S. Richardson, and D. Spiegelhalter. Markov chain monte carlo in practice. *Interdisciplinary Statistics. Chapman & Hall*, 1996.
- [15] D. Heckerman, D. Geiger, and M. Chickering. Learning bayesian networks : The combination of knowledge and statistical data. In *The 10th Conference on Uncertainty in Artificial Intelligence*, 1994.
- [16] D. Heckerman, D. Geiger, and M. Chickering. Learning bayesian networks : The combination of knowledge and statistical data. *Machine Learning*, 20, 1995.
- [17] M. Henrion. An introduction to algorithms for inference in belief nets. In *the 5th Annual Conference on Uncertainty in Artificial Intelligence (UAI-90), New York, NY. Elsevier Science Publishing Company, Inc*, 1990.
- [18] F. Jensen, S. Lauritzen, and K. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4, 1990.
- [19] J. Kim and J. Pearl. A computational model for combined causal and diagnostic reasoning in inference systems. In *IJCAI-83*, 1983.
- [20] U. Kjaerulff. Approximation of bayesian networks through edge removals. *Research Report IR-93-2007, The Machine Intelligence Group, Aalborg University*, 1993.
- [21] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *ACSAC*, 2003.
- [22] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50(2) :157–224, 1988.
- [23] N. Lawrence. *Variational Inference in Probabilistic Models*. PhD thesis, University of Cambridge, U.K., 2000.
- [24] Ph. Leray. *Réseaux bayésiens : apprentissage et modélisation de systèmes complexes. Habilitation à diriger les recherches*. Novembre 2006.

-
- [25] P. Naïm, P-H. Wuillemin, P. Leray, O. Pourret, and A. Becker. *Réseaux bayésiens, 3ème édition*. Eyrolles, Paris, 2007.
- [26] R. Neal and G. Hinton. A view of the em algorithm that justifies incremental, sparse and other variants. *Dans M. I. Jordan (Ed.), Learning in Graphical Models*, pages 355–368, 1998.
- [27] J. Pearl. Bayesian networks : a model of self-activated memory for evidential reasoning. In *Technical Report 850021 (R-43), UCLA Computer Science Department Technical Report, and in Cognitive Science Society, UC Irvine*, 1985.
- [28] J. Pearl. Fusion, propagation, and structuring in belief networks. *Journal of Artificial Intelligence*, 29 :241–288, 1986.
- [29] J. Pearl. Probabilistic reasoning in intelligent systems : Networks of plausible inference. *Morgan Kaufmann*, second edition in 1991, 1988.
- [30] J. Pearl. *Causality : Models, Reasoning, and Inference*. Cambridge, England : Cambridge University Press, 2000.
- [31] J. Pearl and T. Verma. A theory of inferred causation. In *Allen, J. F., Fikes, R., & Sandewall, E. (Eds.), KR'91 : Principles of Knowledge Representation and Reasoning*, 1991.
- [32] D. Poole. Average-case analysis of a search algorithm for estimating prior and posterior probabilities in bayesian networks with extreme probabilities. In *the thirteenth International Joint Conference on Artificial Intelligence*, 1993.
- [33] R. W. Robinson. Counting unlabeled acyclic digraphs. *Little, C. H. C. (Ed.), Combinatorial Mathematics*, 622 of Lecture Notes in Mathematics :28–43, 1977.
- [34] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI Workshop on Learning for Text Categorization*, 1998.
- [35] R. D. Shachter. Intelligent probabilistic inference. *Kanal, L. N. & Lemmer, J. F. (Eds.), Uncertainty in Artificial Intelligence*, pages 371–382, 1986.
- [36] P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search*. Springer-Verlag, 1993.
- [37] P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search (2 ed)*. The MIT Press, 2000.
- [38] G. Wei and M. Tanner. A monte-carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, pages 699–704, 1990.
- [39] N. Zhang and D. Poole. A simple approach to bayesian network computations. In *the tenth Canadian Conference on Artificial Intelligence*, 1994.

Annexe A

Apprentissage de structure des réseaux bayésiens

A.1 Introduction

Lorsque la structure du réseau bayésien n'est pas fournie a priori par un expert, il est possible d'apprendre cette structure à partir d'une base de données. Toutefois, dans la réalité cette base de données peut présenter un manque d'informations pour certaines mesures.

Beaucoup de travaux [24] [12] sont intéressés par ce genre de problèmes et la recherche d'éventuelles solutions. Des approches ont été proposées pour l'apprentissage de structure pour chacun des cas : données complètes ou incomplètes.

A.2 Apprentissage de structure à partir de données complètes

Dans le cas où notre base de données est complète, c'est-à-dire toutes les mesures sont complètes, deux familles à approches ont été proposées.

La première famille (algorithmes IC, PC etc.) consiste à déterminer dans un premier temps un graphe non orienté en tenant compte des différentes indépendances conditionnelles qui existent entre les variables de ce graphe, puis à orienter ce graphe pour obtenir un réseau bayésien. Ces algorithmes sont peu efficaces dans le cas de problèmes de grande taille puisque la détermination de ces indépendances est exponentielle en fonction du nombre des variables.

La deuxième approche consiste à parcourir tous les graphes possibles, associer un score à chaque graphe, puis choisir le graphe ayant le score le plus élevé. Tou-

tefois, cette méthode n'est pas simple, principalement à cause de la taille super-exponentielle de l'espace de recherche en fonction du nombre de variables. Robinson a montré dans [33] que le nombre $r(n)$ de structures différentes des graphes possibles pour n variables est donné par la formule A.1 :

$$r(n) = \sum_{i=1}^n (-1)^{i+1} \binom{n}{i} 2^{i(n-1)} r(n-i) = n^{2^{O(n)}} \quad (\text{A.1})$$

Des idées ont été proposées pour résoudre ce problème. Une première consiste à remplacer l'espace de recherche (espace des réseaux bayésiens) par un espace plus petit, l'espace des arbres (MWST ou Maximal Weight Spanning Tree). Une deuxième idée consiste à ordonner les noeuds pour limiter la recherche des parents possibles pour chaque noeud (algorithme K2). Une troisième consiste à faire une recherche gloutonne dans l'espace des réseaux bayésiens (algorithme GS).

Il existe un autre espace dans lequel nous pouvons apprendre les indépendances de la loi de probabilité : l'espace des représentants des classes d'équivalence de Markov. Cet espace est équivalent à l'espace des réseaux bayésiens sauf qu'il ne contient que des exemplaires des classes d'équivalence de Markov, donc un peu plus petit.

Nous détaillons dans la suite chacune de ces approches.

A.2.1 Recherche des indépendances conditionnelles : algorithmes IC et PC

La recherche des indépendances conditionnelles entre les variables d'un réseau bayésien est une approche utilisée par plusieurs algorithmes d'apprentissage de structure des RB, nous citons les algorithmes IC [31], IC* [30], PC [36] et FCI [37]. Cette approche consiste en :

- La création d'une table d'indépendances conditionnelles entre les différentes variables en effectuant des tests statistiques sur la base de données.
- La construction d'un graphe non dirigé contenant les relations entre les variables issues de la table créée précédemment.
- La détection des V-structures¹ à partir de la table d'indépendances conditionnelles.
- La propagation des orientations de certains arcs en relation avec les noeuds présentant une V-structure.

¹Une sous-structure du type $A \rightarrow C \leftarrow B$ est appelée une V-structure (de noeud puits C)

La première phase de cette approche est la phase la plus importante puisque sur son résultat se basent les phases suivantes pour la détermination d'une structure à partir des données. Cette phase se base sur des tests statistiques qui permettent de remplir la matrice des indépendances entre les variables. Parmi ces tests nous citons les tests du χ^2 et du rapport de vraisemblance G^2 .

Une indépendance entre deux variables notée dans la matrice d'indépendances se traduit par l'absence d'arcs entre les deux noeuds correspondants. une dépendance conditionnelle se traduit par une V-structure.

Une des premières méthodes de recherche d'indépendances conditionnelles efficace proposée fût celle de l'algorithme PC (*Peter and Clark*). Elle a été introduite par [36]. Elle utilise le test statistique χ^2 pour évaluer s'il y a indépendance conditionnelle entre deux variables. L'ensemble des relations d'indépendances conditionnelles découvertes permettent alors de reconstruire la structure du réseau bayésien. Cet algorithme prend comme initialisation un graphe complètement connecté, et lorsqu'une indépendance conditionnelle est détectée à l'aide d'un test statistique, l'arc correspondant est retiré de ce graphe. Les tests statistiques sont effectués suivant un ordre donné préalablement aux variables. [7] a alors montré que cet ordre avait une grande influence sur le résultat rendu par cette méthode.

L'algorithme IC (*Inductive Causation*) est de principe similaire à celui de PC. Il a été introduit à la même époque par [31]. La principale différence entre ces deux algorithmes est que l'algorithme PC est initialisé avec un graphe complètement connecté et utilise l'ensemble des indépendances découvertes pour retirer des arcs, tandis que l'algorithme IC est initialisé avec un graphe vide et utilise l'ensemble de dépendances pour ajouter des arcs.

Puisque la phase de recherche des indépendances conditionnelles est la phase la plus coûteuse, plusieurs simplifications ont été proposées par *Spirtes et al* pour réduire la complexité exponentielle de l'algorithme PC. Ces simplifications ont permis la naissance de l'algorithme PC* et l'accélération de PC.

La variante IC* de l'algorithme IC, permet de détecter si éventuellement il peut exister des variables latentes (voir [30] pour plus de détails). De même, [37] a introduit une version augmentée de PC qui détecte l'existence de variables latentes et elle a été nommée FCI (*Fast Causal Inference*). Ces algorithmes ont alors une complexité exponentielle en nombre de tests d'indépendance.

A.2.2 Notion de score

Comme nous l'avons vu précédemment, le nombre de tests statistiques à effectuer croît exponentiellement avec le nombre de noeuds. Les méthodes de recherche d'indépendances conditionnelles ont donc une complexité élevée. D'autres approches procèdent différemment et se basent sur des fonctions de score pour la détermination de structure des RB à partir de données. Ces approches auront donc à chercher la structure qui maximise ce score. Nous présentons dans la suite quelques notions de base qui seront utiles, les fonctions de score les plus utilisées ainsi que leur caractéristiques.

Soit $X = (X_1, \dots, X_n)$ un vecteur aléatoire composé de n variables aléatoires. r_i la taille du noeud X_i du réseau bayésien B , et $pa(X_i)$ les parents de X_i . Le nombre de paramètres nécessaire pour décrire la distribution de probabilité $P(X_i | pa(X_i) = x_j)$ est égal à $r_i - 1$.

Pour représenter $P(X_i | pa(X_i))$, il faudra alors $Dim(X_i, B)$ paramètres, avec :

$$Dim(X_i, B) = (r_i - 1) \cdot q_i \tag{A.2}$$

où q_i est le nombre de configurations possibles pour les parents de X_i , c'est-à-dire

$$q_i = \prod_{X_j \in pa(X_i)} r_j \tag{A.3}$$

Le nombre de paramètres nécessaire pour décrire toutes les distributions de probabilité du réseau B est appelé **dimension du réseau bayésien** ($Dim(B)$) et défini par la formule A.4.

$$Dim(B) = \sum_{i=1}^n Dim(X_i, B) = \sum_{i=1}^n (r_i - 1) \cdot q_i \tag{A.4}$$

La dimension d'un réseau bayésien est une grandeur qui sera utile pour les fonctions de score qui font intervenir un terme de pénalité. Cette grandeur représente le nombre de paramètres indépendants que possède le réseau.

Il est judicieux de pénaliser les réseaux trop complexes. Les réseaux fortement connectés sont difficilement interprétables. De plus, ils contiennent un très grand nombre de paramètres ce qui nécessitera un nombre énorme de données pour pouvoir faire l'apprentissage. Sinon, l'apprentissage de paramètres ne sera pas très précis. Il faut alors trouver un bon compromis *expressivité/complexité* du réseau. Pour cela, la

plupart des scores existants dans la littérature appliquent le principe de parcimonie du *rasoir d'Occam* : "trouver le modèle qui correspond le mieux aux données mais qui soit le plus simple possible". Ce qui explique le fait que ces scores sont souvent décomposables en deux termes :

- la vraisemblance $L(D|\theta, B)$: correspondances aux données
- la dimension du réseau bayésien $Dim(B)$: nombre de paramètres

Différents scores ont été proposés, citons :

- **Le score bayésien : BD** (*Bayesian Dirichlet*)

Le score bayésien mesure la probabilité des données conditionnellement à la structure. Ce score est décomposable et il est donnée par la formule A.5.

Soit D l'ensemble des données, et G la structure du réseau.

$$ScoreBD(B, D) = P(B) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})} \quad (A.5)$$

où Γ est la fonction *Gamma*

- **Le score BDe** (*Bayesian Dirichlet Equivalent*)

Le score BD est simple à utiliser, mais il possède le désavantage de ne pas être un score équivalent (deux structures équivalentes n'ont pas le même score). Des adaptations ont été faites sur ce score pour en obtenir des nouveaux, très proches, et qui possèdent cette propriété d'équivalence. Dans [15], on a adapté ce score en utilisant une distribution a priori sur les paramètres définie par la formule A.6.

$$\alpha_{ijk} = N' * P(X_i = x_k, pa(X_i) = x_j | B_c) \quad (A.6)$$

où B_c est le graphe complètement connecté et où N' est un nombre d'exemples équivalents défini par l'utilisateur.

- **Le score $BDeu$**

Le score $BDeu$ a le même principe que celui du score BDe , sauf qu'il utilise un a priori uniforme donnée par la formule A.7.

$$\alpha_{ijk} = \frac{N'}{r_i q_i} \quad (A.7)$$

- **Le score $BD\gamma$** (*Bayesian Dirichlet généralisé*)

L'introduction d'un hyperparamètre γ permet de généraliser le score bayésien.

$$ScoreBD\gamma(B, D) = P(B) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\gamma N_{ij} + \alpha_{ij})}{\Gamma((\gamma + 1) N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma((\gamma + 1) N_{ijk} + \alpha_{ijk})}{\Gamma(\gamma N_{ijk} + \alpha_{ijk})} \quad (A.8)$$

La fonction de score de l'équation A.8 a été introduite par [2]. Elle permet de passer du score bayésien avec $\gamma = 0$ à l'entropie conditionnelle quand γ tend vers $+\infty$.

– **Le score *AIC*** (*Akaike Information Criterion*)

$$ScoreAIC(B, D) = \log L(D|\theta^{MV}, B) - Dim(B) \quad (A.9)$$

– **Le score *BIC***

$$ScoreBIC(B, D) = \log L(D|\theta^{MV}, B) - \frac{1}{2}Dim(B) \log N \quad (A.10)$$

– **La Longueur de Description Minimale (*MDL*)**

$$ScoreMDL(B, D) = \log L(D|\theta^{MV}, B) - |A_B| \log N - c.Dim(B) \quad (A.11)$$

où $|A_B|$ est le nombre d'arcs dans le graphe B et c le nombre de bits utilisé pour stocker chaque paramètre numérique.

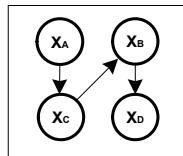
La recherche gloutonne

La méthode de recherche gloutonne GS (Greedy Search) est la plus générale puisqu'elle recherche dans l'espace complet des DAG. Elle prend un graphe de départ, définit un voisinage de ce graphe, puis associe un score à chaque graphe du voisinage. Le meilleur graphe est alors choisi comme point de départ de l'itération suivante. Le voisinage d'un DAG est défini par l'ensemble des DAG obtenus en ajoutant, en supprimant ou en inversant un arc du graphe courant (voir exemple de la Figure A.1 issu de [25]).

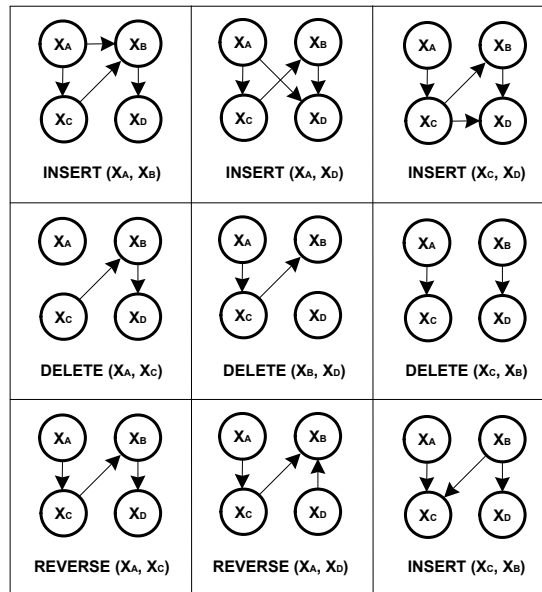
L'algorithme itère jusqu'à l'obtention d'un graphe qui maximise la fonction du score utilisé. Ce maximum peut être local. Pour éviter la convergence vers un optimum local, une idée est de répéter la recherche gloutonne plusieurs fois à partir d'initialisations différentes tirées aléatoirement. Cette méthode (iterated hill climbing ou random restart) permet converger à chaque fois vers un optimum local qui peut être meilleur que les précédents, ainsi une forte chance de converger, à la fin, vers la solution optimale.

La figure A.1 montre que pour un exemple de petite taille, le voisinage comprend un nombre plus important. Pour choisir dans ce voisinage celui qui sera le graphe de départ pour l'itération suivante, il faut calculer le score de chacun de ces graphes. Pour des structures complexes (grand nombre de noeuds), la taille du voisinage devient plus importante ce qui nécessite de ne plus recalculer le score entièrement pour chaque graphe du voisinage, mais il sera seulement nécessaire de mettre à jour le score en fonction de l'opération effectuée (insertion, suppression ou inversion d'arc),

Considérons le graphe B suivant, ainsi qu'un voisinage défini par les trois opérateurs ajout (*INSERT*), suppression (*DELETE*) et retournement (*REVERSE*) d'arc. Remarquons que les graphes résultants ne sont retenus que s'ils sont sans circuit.



Génération du voisinage de B :

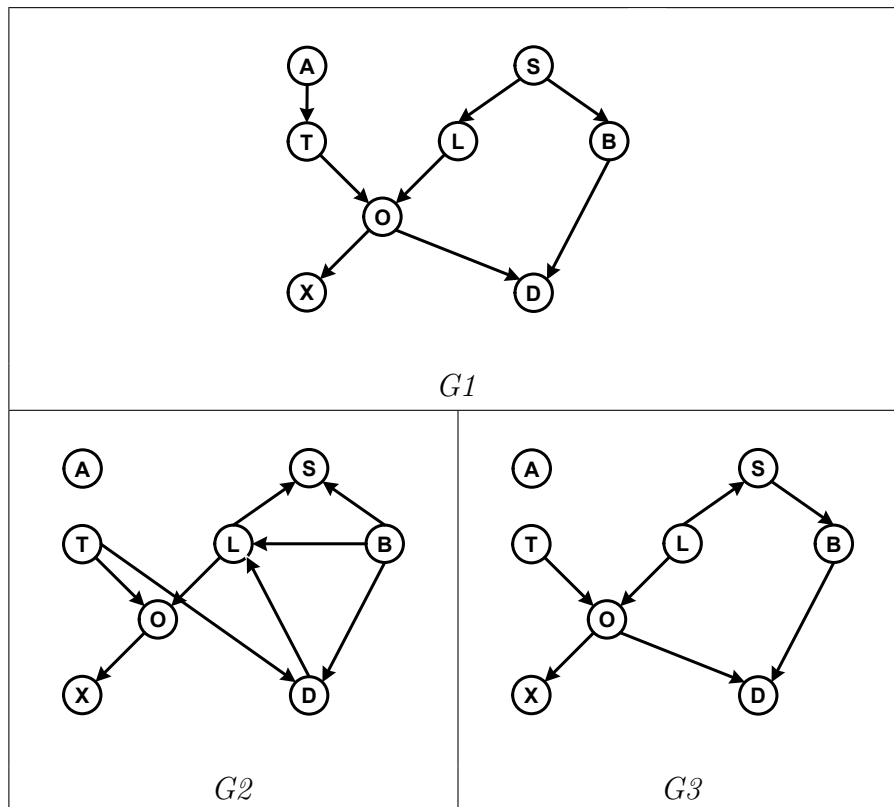


TAB. A.1 – Exemple de voisinage GS

c'est la notion de score décomposable. Il est judicieux aussi de recourir à un système de cache pour ne pas recalculer plusieurs fois chaque score local.

La recherche gloutonne étant une méthode itérative, il faut alors choisir une initialisation.

En testant la recherche gloutonne avec différentes initialisations nous obtenons des résultats de convergence différents. Il est ainsi important de choisir une bonne initialisation. Il est possible de recourir aux experts pour définir un graphe de départ, des test dans [11] ont montré que l'initialisation de la recherche gloutonne par le graphe issu de l'algorithme MWST permet assez souvent d'obtenir une structure meilleure que celle obtenue avec une initialisation aléatoire ou vide. L'exemple A.2 montre l'avantage de l'initialisation avec le graphe de poids maximal (MWST).



TAB. A.2 – Résultats de l'algorithme GS avec différentes initialisations

Avec :

- G_1 est le réseau "ASIA" théorique avec lequel les données ont été générées.
- G_2 est le graphe obtenu avec les données ASIA et le score BIC en partant d'un graphe d'initialisation vide.
- G_3 est le graphe obtenu avec les mêmes données en partant de l'arbre fourni par l'algorithme MWST.

Recherche dans l'espace des arbres

Cette méthode consiste à rechercher un arbre (forme particulière de graphe) dit arbre de poids maximal qui relie la totalité des noeuds en maximisant un critère. Elle utilise la notion de recherche de l'arbre de recouvrement maximal (MWST) en recherche opérationnelle.

Cette méthode s'applique à la recherche de structure d'un réseau bayésien en fixant un poids à chaque arête potentielle A–B de l'arbre. Ce poids peut être par exemple l'information mutuelle entre les variables A et B comme proposé dans [4] (voir formule A.12), ou encore la variation du score local lorsqu'on choisit B comme parent de A comme proposé dans [16] (voir formule A.13).

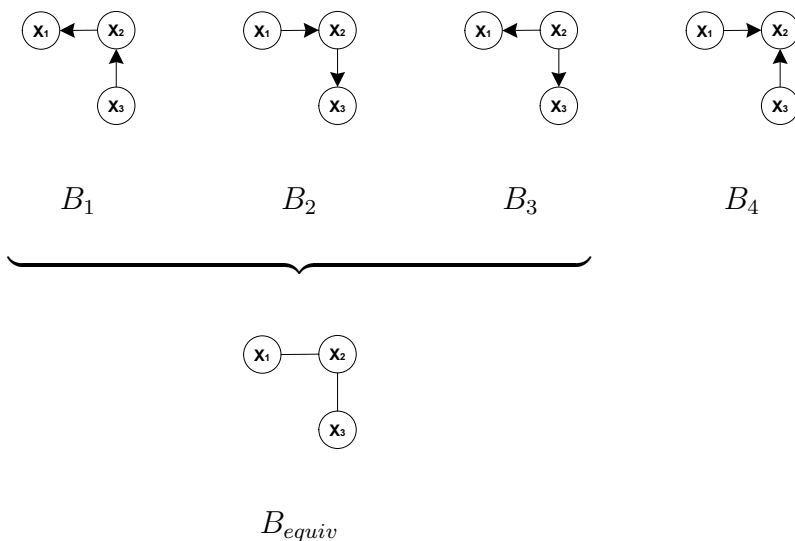
$$\begin{aligned} W_{CL}(X_A, X_B) &= \sum_{x_a, x_b} P(X_A = x_a, X_B = x_b) \log \frac{P(X_A = x_a, X_B = x_b)}{P(X_A = x_a) P(X_B = x_b)} \\ &= \sum_{a,b} \frac{N_{ab}}{N} \log \frac{N_{ab} N}{N_a N_b} \end{aligned} \quad (\text{A.12})$$

$$W(X_A, X_B) = \text{score}(X_A, X_B) - \text{score}(X_A, \emptyset) \quad (\text{A.13})$$

où $\text{score}(X_A, X_B)$ est le score local en X_A en supposant qu'il a X_B comme parent, et $\text{score}(X_A, \emptyset)$ est le score local en X_A en supposant qu'il n'a aucun parent. A partir de ce calcul de score, nous calculons une matrice W tel que : $W_{ij} = W(X_i, X_j)$ Une fois cette matrice de poids définie, il suffit d'utiliser un des algorithmes standards de résolution du problème de l'arbre de poids maximal comme l'algorithme de *Kruskal*. L'arbre de poids maximal est un arbre non dirigé reliant toutes les variables, donc il doit par la suite être dirigé en choisissant une racine puis en parcourant l'arbre par une recherche en profondeur. La racine peut être choisie soit aléatoirement, soit à l'aide de connaissance a priori.

Recherche dans l'espace des représentants des classes d'équivalence de Markov

Nous commençons cette partie par définir la notion d'équivalence de Markov. Deux réseaux bayésiens B_1 et B_2 sont dits équivalents au sens de Markov s'ils représentent les mêmes relations d'indépendance conditionnelle. Il est simple de montrer que les structures B_1 , B_2 et B_3 sont équivalentes, tandis que la structure B_4 n'est équivalente à aucune de ces structures puisqu'elle présente une V-structure. Donc les trois structures B_1 , B_2 et B_3 appartiennent à la même classe d'équivalence qui peut être représenté par le graphe sans circuit partiellement dirigé représenté dans la figure B_{equiv} .



Algorithme K2

Cooper & Hersovits [6] ont proposé une méthode largement utilisée dans l'apprentissage de structure des RB. Cette méthode est appelée K2. L'algorithme K2 présente un inconvénient de demander un ordre d'énumération des variables en paramètre d'entrée. Le but de cette méthode est donc de maximiser la probabilité de la structure sachant les données dans l'espace des DAG (espace des graphes acycliques dirigés) en respectant la contrainte de l'ordre d'énumération. Cette contrainte permet de restreindre l'espace de recherche et de le limiter par la recherche seulement des arcs intéressants. En effet, l'algorithme K2 teste l'ajout de parents en respectant cet ordre de la manière suivante : si X_i est le premier sur la liste, alors il ne pourra avoir de parents. sinon, si X_i est cité avant X_j alors il ne pourra y avoir d'arc de X_j vers X_i . Autrement dit, l'ensemble des parents choisis pour un noeud est le sous-ensemble de noeuds qui augmente le plus le score parmi l'ensemble des noeuds le **précédant** dans l'ordre d'énumération.

Ainsi, l'espace de recherche est réduit à l'ensemble des DAG respectant cet ordre.

Cette technique utilise originellement le score *Bayesian Dirichlet*.

A.3 Apprentissage de structure à partir de données incomplètes

En réalité, plusieurs bases de données présentent des manques de mesures. Ce manque de données revient à plusieurs raisons. Par exemple, un instrument de mesure tombe en panne pour une période ce qui se traduit par un manque de données

pour toutes les séries de mesures durant cette période, le système entre dans un état particulier imprévu alors il répond aléatoirement par l'une des réponses possibles ou il se bloque ,ou encore un responsable de la prise de mesures oublie de le faire pour une cause ou une autre (fatigue, sommeil, mesures variant très rapidement ...), ou que la donnée reportée ne soit pas lisible.

Quand nous voulons modéliser le comportement de notre système à partir de cette base de données, nous aurons un problème causé par ces manques de mesures. La première idée qui nous vient à la tête est de filtrer les données et ne prendre que les exemples complètement observés de la base, sauf que nous ne disposons plus de la même quantité de données pour faire l'apprentissage et nous risquons de ne plus avoir qu'un nombre très petit de données dans la base lorsque le taux de valeurs manquantes est élevé. L'idée la plus pertinente est de compléter les données manquantes en utilisant une technique de substitution par la moyenne ou encore la méthode de maximum de vraisemblance (*algorithme EM*).

Afin de faire l'apprentissage de structure de réseau bayésien, les deux approches proposées dans le cas de données complètes (approche statistique et approche basée sur un score) sont aussi valables dans le cas de données incomplètes mais avec des rectifications.

Nous détaillons dans la suite le principe de l'algorithme EM ainsi que les rectifications aux fonctions de scores pour les rendre adaptés aux données incomplètes. Nous finirons par présenter les méthodes proposées dans la littérature pour l'apprentissage de structure des réseaux bayésiens à partir de données incomplètes.

A.3.1 Algorithme Expectation-Maximisation

L'algorithme EM est une méthode itérative pour évaluer le maximum de vraisemblance des paramètres de modèles probabilistes en présence de données incomplètes [8]. Cet algorithme consiste en l'alternance de deux étapes :

- Évaluation de l'espérance (E), où l'on calcule l'espérance de la vraisemblance en tenant compte des dernières variables observées.
- Maximisation (M), où l'on estime le maximum de vraisemblance des paramètres en maximisant la vraisemblance trouvée à l'étape E.

On utilise ensuite les paramètres trouvés en M comme point de départ d'une nouvelle phase d'évaluation de l'espérance, et l'on itère ainsi jusqu'à convergence. La convergence de l'algorithme EM a été prouvée dans [8].

Principe de fonctionnement de EM :

Soit D l'ensemble de données de la base, D_O la partie observées mais incomplète de D et D_m la partie manquante. Le but de l'algorithme EM est de maximiser la

vraisemblance $P(D|\theta)$ des paramètres θ d'un modèle donné sachant les données D . Puisque la fonction logarithmique (\log) est strictement croissante la valeur qui maximise $P(D|\theta)$ maximise également $L(\theta)$, avec : $L(\theta) = \log(P(D|\theta))$.

A partir d'un modèle de référence θ^* , il est possible d'estimer $P(D_m|\theta^*)$, et par la suite calculer l'espérance de la log-vraisemblance précédente :

$$Q(\theta : \theta^*) = E_{\theta^*}[\log P(D_0, D_m|\theta)] \quad (\text{A.14})$$

$Q(\theta : \theta^*)$ étant alors l'espérance de la vraisemblance des paramètres θ quelconques. Cette espérance est calculée en utilisant des données manquantes.

L'étape suivante consiste en la recherche des paramètres θ_1^* qui maximise la vraisemblance.

$$\theta_1^* = \arg \max_{\theta} L(\theta) = \arg \max_{\theta} Q(\theta : \theta^*) \quad (\text{A.15})$$

Donc le principe de l'algorithme EM se résume en deux étapes, soient $\theta^{(t)}$ les paramètres du réseau bayésien à l'itération t .

Etape E : calculer l'espérance $Q(\theta : \theta^{(t)})$ à partir des paramètres de référence $\theta^{(t)}$.

Etape M : choisir la meilleure valeur des paramètres $\theta^{(t+1)}$ en maximisant Q .

$\theta^{(t+1)}$ sera alors les paramètres de référence à la place de $\theta^{(t)}$, et on itère ces deux étapes E et M tant que la valeur de Q augmente.

De nombreuses variantes de l'algorithme EM ont été proposées vu le succès de cette méthode. Nous citons l'algorithme *EM généralisé* qui propose de ne pas chercher les paramètres qui maximisent Q à l'étape M, mais se contente de trouver une valeur meilleure que celle courante. Il existe aussi l'algorithme *EM incrémental* [5] appelé *Voting EM* permettant un apprentissage de paramètres de réseaux bayésiens de manière incrémentale, l'algorithme *Monte-Carlo EM* [38] qui utilise une méthode de Monte-Carlo pour évaluer les intégrales et les sommes de très grande taille, l'algorithme *EM variationnel* [26] [1] qui propose d'utiliser des méthodes variationnelles pour effectuer une estimation lors du calcul des intégrales, comme il existe un algorithme adapté pour la classification (*CEM*) [3].

A.3.2 Méthodes basées sur un score avec données incomplètes

Pour faire de l'apprentissage de structure d'un réseau bayésien à partir de données incomplètes, il est indispensable de résoudre les deux problèmes rencontrés, l'un lors de l'apprentissage de paramètres d'une structure à partir de données incomplètes, l'autre lors de l'apprentissage de structure à partir de données incomplètes.

Un algorithme SEM [13] (Structural EM) a été proposé. Cet algorithme se base sur l'algorithme EM pour faire de l'apprentissage de structure. En effet, le principe de cet algorithme est de se donner une structure initiale B^0 ayant des paramètres Θ^0 (aléatoirement choisis), faire de l'apprentissage des paramètres de cette structure à partir de la base de données incomplète, ensuite rechercher dans l'espace $\{B, \Theta\}$ la meilleure structure pour les paramètres appris Θ^1 . Pour aboutir à une structure bien vraisemblable aux données, SEM propose d'itérer ces deux phases. L'algorithme SEM est alors une adaptation de l'algorithme EM. Son principe est décrit dans l'algorithme A.3.

Algorithme EM Structurel générique
<ul style="list-style-type: none"> • Init : $i = 0$ • Initialisation du graphe B^0 (Choix aléatoire ou utilisant une heuristique pour sélectionner le réseau bayésien initial B^0) • Initialisation des paramètres Θ^0 • Répéter : <ul style="list-style-type: none"> • $i = i + 1$ • $(B^i, \Theta^i) = \arg \max_{B, \Theta} Q(B, \Theta : B^{i-1}, \Theta^{i-1})$ <p>Tant que : $Q(B^i, \Theta^i : B^{i-1}, \Theta^{i-1}) - Q(B^{i-1}, \Theta^{i-1} : B^{i-1}, \Theta^{i-1}) > \epsilon$</p>
<p>Notations :</p> <p>$Q(B, \Theta : B^*, \Theta^*)$: Espérance de la vraisemblance d'un réseau bayésien $\langle B, \Theta \rangle$ calculée à partir de la distribution de probabilité des données manquantes $P(D_m B^*, \Theta^*)$</p>

TAB. A.3 – Algorithme EM Structurel générique

L'étape de maximisation est relative en pratique à deux maximisations, l'une dans l'espace des DAG (équation A.16), l'autre dans l'espace des paramètres (équation A.17) :

$$B^i = \arg \max_B Q(B, \bullet : B^{i-1}, \Theta^{i-1}) \quad (\text{A.16})$$

$$\Theta^i = \arg \max_{\Theta} Q(B^i, \Theta : B^{i-1}, \Theta^{i-1}) \quad (\text{A.17})$$

où la notation $Q(B, \bullet : B^{i-1}, \Theta^{i-1})$ correspond à l'espérance $E_{\Theta} [Q(B, \Theta : B^{i-1}, \Theta^{i-1})]$ pour une approche bayésienne, ou $Q(B, \Theta^{MV}, : B^{i-1}, \Theta^{i-1})$ pour une approche par maximum de vraisemblance.

Dans l'équation A.16, la maximisation dans l'espace des DAG (espace de taille super-exponentielle) peut être coûteuse en temps de calcul. Cette maximisation peut être remplacée par une recherche d'une structure meilleure que la structure actuelle (principe de l'algorithme EM généralisé) au lieu de parcourir tout l'espace. La structure meilleure peut être recherchée dans le voisinage V_B de la structure actuelle (principe de la recherche gloutonne). V_B est défini comme étant l'ensemble des DAG qui diffèrent de la structure courante B par l'ajout, la suppression ou l'inversion d'un arc.

La recherche dans l'espace des paramètres se fait en utilisant l'algorithme EM paramétrique (répétition de l'opération plusieurs fois en utilisant des initialisations intelligentes). Les paramètres optimaux de l'itération précédente sont utilisés pour initialiser les paramètres de la structure maximisant le score utilisé. Ce score est l'espérance du score BIC ou MDL.

Le problème de manque de certaines mesures se pose aussi lors du calcul des fonctions de score à partir de données incomplètes. Il est possible de faire le calcul de ces fonctions à partir des exemples complets seulement, mais cette méthode n'est justifiée que lorsque le taux de manque est très faible. Une seconde possibilité propose de faire le calcul en tenant compte seulement des exemples disponibles pour chaque terme. Par exemple, l'équation du score BIC pour des données complètes garde sa structure sauf que les N_{ijk} seront évalués à partir de la base de données incomplètes de la manière suivante : pour calculer les N_{ijk} , le comptage ne se fait que sur les exemples de la base où X_i et $Pa(X_i)$ sont complètement observés.

Il est possible d'adapter le score BIC pour les données incomplètes. La fonction de ce score Q^{BIC} s'écrit alors :

$$Q^{BIC}(B, \Theta : B^*, \Theta^*) = E_{B^*, \Theta^*}[\log P(D_0, D_m | B, \Theta)] - \frac{1}{2} \text{Dim}(B) \log N \quad (\text{A.18})$$

Q^{BIC} est décomposable puisque le score BIC l'est :

$$Q^{BIC}(B, \Theta : B^*, \Theta^*) = \sum_{i=1}^n Q^{bic}(X_i, P_i, \Theta_{X_i|P_i} : B^*, \Theta^*) \quad (\text{A.19})$$

où Q^{bic} est le score local :

$$Q^{bic}(X_i, P_i, \Theta_{X_i|P_i} : B^*, \Theta^*) = \sum_{X_i=x_k} \sum_{P_i=pa_j} N_{ijk}^* \log \theta_{ijk} - \frac{\log N}{2} \text{Dim}(X_i, B) \quad (\text{A.20})$$

avec $N_{ijk}^* = E_{B^*, \Theta^*}[N_{ijk}] = N * P(X_i = x_k, P_i = pa_j | B^*, \Theta^*)$ obtenu par inférence dans le réseau $\{B^*, \Theta^*\}$ si $\{X_i, P_i\}$ ne sont pas complètement mesurées, ou calculé classiquement sinon.

Un deuxième algorithme a été proposé dans [10]. Cet algorithme, MWST-EM, a pour principe de rechercher la structure arborescente optimale reliant toutes les variables. Cet algorithme est similaire à celui SEM (algorithme A.3), mais il ne propose plus de choisir un meilleur graphe dans un voisinage, mais la structure arborescente optimale. Nous détaillerons le principe de cet algorithme plus tard.

A.4 Conclusion

L'objet de cette annexe était de présenter la notion d'apprentissage de structure des réseaux bayésiens à partir de bases d'observations pouvant être complètes ou incomplètes. Nous avons détaillé également les différentes méthodes adoptées pour chacun des cas.