Chapitre 9 : Une étude comparative pour la détection de dépendances multiples

Elham Salehi, Jayashree Nyayachavadi, Robin Gras

Université de Windsor
401 Sunset Avenue
N9B 3P4 Windsor, Ontario Canada
rgras@uwindsor.ca
http://cs.uwindsor.ca/faculty/rgras

Résumé. La recherche de dépendances entre variables à partir d'exemples est un problème important en optimisation. De nombreuses méthodes ont été proposées pour résoudre ce problème mais peu d'évaluations à grande échelle ont été effectuées. La plupart de ces méthodes reposent sur des mesures de probabilité conditionnelle. L'ASI proposant un autre point de vue sur les dépendances, il était important de comparer les résultats obtenus grâce à cette approche avec l'une des meilleures méthodes existantes actuellement pour cette tâche : l'heuristique max-min. L'ASI n'étant pas directement utilisable pour traiter ce problème, nous avons conçu une extension à cette mesure spécifiquement adaptée. Nous avons réalisé un grand nombre d'expériences en faisant varier des paramètres tels que le nombre de dépendances, le nombre de variables concernées ou le type de prédiction effectuée pour comparer les deux approches. Les résultats montrent une forte complémentarité des deux méthodes.

1 Introduction

Il existe de nombreuses situations pour lesquelles il est nécessaire de trouver une relation entre les variables d'un domaine. Cela est particulièrement vrai lorsque l'on s'intéresse à des problèmes d'**optimisation**. La notion d'optimisation est une notion très répandue du fait que la résolution dans grand nombre de problèmes concrets nécessite, à un moment ou à un autre du processus de résolution, d'optimiser un ensemble de paramètres (ou variables) vis-à-vis d'une **fonction objectif** F donnée. Dans beaucoup de problèmes réels, en particulier en bioinformatique, la fonction objectif n'est partiellement voire pas du tout connue. On dispose cependant de la possibilité d'évaluer la valeur de la fonction en chaque point de l'espace de recherche. On appelle ce type de problème, des problèmes "boite noire".

De nombreux travaux ont été menés pour étudier la complexité des problèmes d'optimisation combinatoire. Dans un premier temps, ce sont principalement des preuves de NP-complétude qui ont été réalisées pour certains problèmes classiques (sac à dos, le coloriage de graphe, la recherche d'une clique de taille k…) (Garey M.R. and Johnson D.S., 1979; Papadimitriou C.H. and Steiglitz K., 1998). Ce n'est cependant pas un indice de complexité très informatif. On sait que pour une classe de problèmes démontrés NP-complet

il existe dans la plupart des cas des instances de ces problèmes solubles en temps polynomial. Malheureusement, les propriétés des problèmes qui les rendent plus ou moins faciles à résoudre sont souvent mal connues. Cette information est d'autant plus importante à découvrir qu'elle est en général la clé permettant la conception d'un algorithme de résolution efficace. Des notions comme la décomposabilité d'un problème ou le nombre de solutions du problème ont une implication forte sur sa complexité. La principale propriété influant sur la complexité est le degré d'**épistasie** d'un problème, c'est-à-dire le nombre maximum de variables dont dépend, dans le calcul de F, chacune des n variables. Par exemple, un problème d'épistasie zéro est un problème où toutes les variables sont indépendantes. L'épistasie est un concept plus général que celui de décomposabilité puisqu'un problème de degré d'épistasie k n'implique pas que le problème soit décomposable en n/k sous-problèmes complètement indépendants. En effet, les sous-problèmes peuvent contenir des variables communes et on peut, par exemple, tout à fait avoir n sous-problèmes de taille k. k variables dépendantes sont appelées un bloc. Les blocs peuvent être ou non chevauchants s'ils partagent des variables. Dès que k est égal à deux et que le problème n'est pas décomposable en problèmes indépendants (une même variable est contenue dans plusieurs contextes différents), il a été démontré que le problème est NP-complet[1] (Thompson R.K. and Wright A.H., 1996).

Disposer d'un modèle décrivant les dépendances entre les variables de la fonction à optimiser apporte des informations essentielles sur la structure de cette fonction et permet donc de simplifier considérablement le problème d'optimisation. Par exemple, savoir quelle(s) variable(s) influe(nt) sur quelle(s) autre(s) peut être très utile pour le problème de sélection de variables (feature selection), Zeng et Hernandez (2008), Goldebberg et Moore (2004) ainsi que pour la décomposition du problème en sous-problèmes indépendants ; pouvoir prédire la valeur d'une variables en fonction de celles d'autres variables permet de résoudre le problème de la classification ; savoir quelles combinaisons d'instanciation d'un ensemble de variables conduisent à la valeur optimale d'une certaine fonction…

Le modèle utilisé classiquement pour cette tâche de détection de dépendances est le **réseau bayésien**. Ce réseau correspond en fait à la factorisation de la distribution de probabilités représentant un ensemble d'exemples composé de l'instanciation de l'ensemble des variables de F. L'avantage majeur d'un tel modèle, en plus de donner une représentation visuelle des dépendances du problème, est qu'il peut servir de générateur de nouveaux exemples compatibles avec le modèle. On peut ainsi l'utiliser pour explorer l'espace de recherche de F en tenant compte de propriétés précédemment découvertes. Les approches de types **Algorithmes Génétiques** par Construction de Modèles Probabilistes (AGCMP) reposent sur ce principe. A chaque génération, un modèle (réseau bayésien) de l'ensemble d'exemples courants est construit. Ce modèle est ensuite utilisé pour générer un nouvel ensemble d'exemples. Puis F est évaluée sur chacun de ces nouveaux exemples et un nouvel ensemble est constitué par tirage aléatoire biaisé avec remise dans l'ensemble précèdent. Le biais de sélection donne la préférence aux exemples pour lesquels la valeur de F est la plus grande, ce qui conduit à un nouvel ensemble d'exemples dont la valeur moyenne est plus haute que celle du précèdent. En itérant ce procédé, à chaque étape un nouveau modèle, plus représentatif de la fonction F, est construit, permettant lui-même de générer de nouveaux exemples pour lesquels la valeur de F sera plus élevée. Il a été démontré que ce processus converge vers l'optimum global de la fonction.

[1] Le problème peut en effet être transformé en MAX-2-SAT qui est NP-complet.

Cependant, avec cette approche, chaque étape demande la construction d'un réseau bayésien à partir d'un nouvel ensemble d'exemples. Or il est bien connu que la construction d'un réseau bayésien à partir d'exemples est un problème qui est en lui-même NP-difficile. Ainsi, pour résoudre un problème NP-difficile, il faudrait résoudre une succession de problèmes eux aussi NP-difficiles. Il faut toutefois remarquer qu'à chaque étape de cette méthode il n'est pas nécessaire de construire un modèle parfait, une simple approximation peut suffire. Chaque étape permet de découvrir de nouvelles propriétés du problème qui seront transmises à l'étape suivante par l'intermédiaire d'un nouvel ensemble d'exemples représentatif de ces propriétés. Il est donc important de disposer d'algorithmes heuristiques efficaces permettant de construire un réseau bayésien de bonne qualité. La difficulté de cette tâche réside bien évidemment dans la détection des dépendances du problème pour éviter d'avoir à considérer un nombre exponentiellement grand (en fonction du nombre de variables) de réseaux. Toute méthode permettant de découvrir où se situent les dépendances les plus fortes au sein d'un ensemble de variables est donc particulièrement importante. Une tâche connexe à celle là, et toute aussi importante, est de détecter les sous-ensembles de variables de telle sorte que toute variable au sein d'un même sous-ensemble a une relation de dépendance avec toutes les autres variables du même sous-ensemble et d'indépendance avec celles des autres sous-ensembles. Un cas particulier de cette situation est de détecter les variables qui ne dépendent d'aucune autre. Ce genre d'information est essentiel dans l'optique d'une décomposition du problème. En effet, chaque sous-ensemble peut être optimisé indépendamment des autres ce qui réduit considérablement la complexité du problème.

2 L'heuristique max-min

La détection de **dépendances multiples** à partir d'un ensemble d'exemples est un problème difficile. Il est clair que ce problème ne peut être résolu de manière exacte dès que le nombre de variables considérées dépasse cinq ou six et que le nombre maximum de variables dont peut dépendre une variable approche la dizaine. Or, pour certains problèmes, le nombre de variables peut atteindre plusieurs centaines ou plusieurs milliers. Il est donc particulièrement important de disposer de méthodes permettant d'obtenir une solution approchée de bonne qualité. L'approche utilisée généralement est une approche par recherche locale. Dans ce cas, un modèle des dépendances est recherché incrémentalement, chaque étape consistant à ajouter ou retirer une ou plusieurs dépendances dans le modèle. Le choix des dépendances à ajouter ou retirer se fait en utilisant un score qui évalue la qualité du nouveau modèle en fonction de l'ensemble d'exemples. L'espace de recherche de cette approche reste exponentiel en fonction du nombre maximum de variables dont peut dépendre une variable. Il est donc nécessaire de disposer de méthodes permettant d'augmenter les chances de construire un modèle de bonne qualité sans explorer exhaustivement tout l'espace de recherche. Une approche possible est d'utiliser une méthode moins coûteuse en temps de calcul pour déterminer un sous-ensemble de l'espace de recherche prometteur sur lequel on peut appliquer ultérieurement une méthode plus systématique et plus coûteuse.

Un intérêt de cette approche est que l'on peut utiliser une mesure de dépendance pour la première phase permettant la réduction de l'espace de recherche puis une autre mesure pour la construction du modèle final. Le modèle final utilisé le plus fréquemment est le réseau bayésien qui permet de représenter des dépendances correspondant à des probabilités

conditionnelles. Cependant, pour construire ce modèle, il est tout à fait possible d'utiliser des informations provenant d'autres mesures que les probabilités conditionnelles. Dans cette optique, les mesures effectuées lors de la première phase sont utilisées comme filtre pour éliminer les variables indépendantes ou grouper les variables en sous-groupes partageant des dépendances. La deuxième phase utilise ces informations filtrées pour construire un réseau bayésien. Le but de notre étude est de comparer la capacité de deux approches pour la détection des dépendances pour la première phase. Dans cette section nous considérons directement la mesure de probabilité conditionnelle et elle sera comparée dans la section suivante à une mesure basée sur ASI.

2.1 Définitions et notations

Nous considérons un problème composé de n variables $\{v_1, v_{2, ..., } v_n\}$. Chaque variable v_i peut prendre différentes valeurs parmi l'ensemble de modalités $M_i = \{m_{i,1}, m_{i,2}, \ldots m_{i,k}\}$. Pour la détection des dépendances un ensemble de N exemples est disponible. Chaque exemple correspond à l'instanciation de chacune des n variables dans l'une des k modalités possibles.

Soit Par_i, l'ensemble **parent** de v_i, l'ensemble des variables dont dépend la variable v_i. On dira que $v_j \in Par_i$ est un parent de v_i et que v_i est un **enfant** de v_j.

2.2 L'approche grimpeur max-min

Bien que la construction de réseaux bayesiens soit un domaine de recherche suscitant de nombreuses publications et que des algorithmes exacts aient été donnés pour des problèmes pour lesquels le nombre de variables est inférieur à 30, Koivisto et Sood (2004), l'application de ces algorithmes de construction à des problèmes réels, tels que l'étude des réseaux biologiques ou sociaux, doit toujours faire face au problème du grand nombre de dimensions. Ces dernières années, plusieurs algorithmes ont été développés spécifiquement pour résoudre ses problèmes par des approches basées sur la restriction de l'espace de recherche des structures de réseaux possibles en utilisant différentes heuristiques, Friedman et al. (1999), Tsamardinos et al. (2006). L'un de ces algorithmes, le « Spare Candidate », Friedman et al. (1999), a une complexité polynomiale. Le principe de cette méthode est de restreindre les ensembles parents de chaque variable en supposant que, si deux variables sont quasiment indépendantes dans l'ensemble d'exemples, il est très improbable qu'elles soient connectées dans le réseau bayésien. En utilisant ce principe, l'algorithme « Spare Candidate » construit des ensembles parents de taille identique et contenant peu d'éléments pour toutes les variables. Le problème majeur avec cette approche est de choisir quelle taille d'ensemble parent utiliser ce qui nécessite une très bonne connaissance préalable du problème à résoudre. Du plus, cette taille étant la même pour toutes les variables, cela induit une présupposition très contraignante d'uniformité de densité du réseau.

Plus récemment, un autre algorithme, permettant d'obtenir de meilleurs résultats sur un plus large panel de structure de réseau, a été proposé, Tsamardinos et al. (2006). Cet algorithme, appelé **grimpeur max-min** (MMHC pour max-min Hill Climber), permet de découvrir les possibles relations parents-enfants en utilisant une méthode par contrainte et les utilise pour construire un réseau bayésien. La première étape de cet algorithme, celle qui nous intéresse car c'est elle qui détecte les dépendances, est appelée max-min parents-enfants (MMPC pour max-min Parent Children). L'algorithme MMPC utilise une structure de

données appelée ensemble parents-enfants associée à chaque variable v_i qui contient l'ensemble des variables qui sont soit parent soit enfant de v_i en respectant les distributions mesurées dans l'ensemble d'exemples. Cette notion de respect des distributions des exemples est définie dans Tsamardinos et al. (2006), Neapolitan (2003). MMPC utilise pour cela le test statistique G^2, Spirtes et al. (2000), appliqué sur l'ensemble d'exemples pour déterminer les indépendances conditionnelles entre des paires de variables sachant un ensemble d'autres variables. L'algorithme MMPC est composé de deux phases. Dans la première, un ensemble vide de candidat parents-enfants (CPC) est associe à v_i. Il essaie ensuite d'ajouter des variables une par une dans cet ensemble en utilisant l'heuristique max-min. Cette heuristique sélectionne une variable v_j qui maximise l'association minimale de v_i relativement au CPC courant de v_i. L'association minimale de v_j et v_j relativement à l'ensemble de variable CPC est définie par :

$$\text{MinAssoc}(v_i ; v_j | \text{CPC}) = \text{argmin Assoc}(v_i ; v_j | S) \text{ pour tout sous-ensemble S de CPC}$$

$\text{Assoc}(v_i ; v_j | S)$ est une estimation de l'importance de la corrélation entre v_i et v_j sachant le CPC et est égale à zéro si v_i et v_j sont indépendantes conditionnellement sachant le CPC. La fonction Assoc utilise la p-value renvoyée par le test d'indépendance G^2 comme une mesure de corrélation : plus la valeur de la p-value est petite plus la corrélation est forte. La première phase de MMPC s'arrête lorsque toutes les variables restantes sont considérées comme indépendantes de v_i sachant le CPC courant. Cette approche étant gloutonne, une variable ajoutée à une étape de cette première phase peut très bien être en fait inutile après que d'autres variables aient été ajoutées au CPC. La deuxième phase de MMPC essaie de corriger ce problème en effectuant un deuxième passage pour toutes les variables du CPC de v_i et en enlevant toutes celles qui sont indépendantes de v_i sachant un sous-ensemble de CPC.

Ce qui n'apparaît pas clairement dans les publications associées à ces méthodes c'est leur capacité à découvrir tout type de structure et comment différentes distributions de probabilités conditionnelles et structures du réseau du modèle réel (celui que l'on cherche à découvrir) influent sur la qualité des résultats obtenus. Nous présentons dans la section suivante les résultats que nous avons obtenus en utilisant l'algorithme MMPC sur des exemples générés à partir de différents réseaux bayésiens.

2.3 Effet de la structure du model utilisé pour générer les exemples sur l'efficacité de l'heuristique max-min

Dans cette section, nous étudions la capacité de l'algorithme max-min à découvrir les bons ensembles parents-enfants des variables d'un réseau bayésien à partir de données générées par ce réseau.

Un réseau bayésien est un outil permettant de présenter la distribution conjointe d'un ensemble de variables aléatoires. Les propriétés de dépendances de cette distribution sont codées sont la forme d'un graphe acyclique direct (GAD). Les noeuds de ce graphe sont les variables aléatoires et les arcs correspondent aux influences directes entre les variables. Une table de distribution de probabilités conditionnelles (DPC), appelée aussi paramètres locaux d'une variable, est associée à chaque nœud du graphe. Elle représente la distribution de probabilités $P(v_i | \text{Par}_i)$.

Dans les sections 2.3.1-2.3.3, nous utilisons des données artificielles générées à partir de réseaux bayésiens construits aléatoirement. Chacun de ces réseaux comporte A arcs et n =

100 variables divisées en deux ensembles : un ensemble D de variables pour lesquelles il existe des relations de dépendances directes avec au moins une des n-D-1 autres variables ; un ensemble de I de variables ne possédant aucune relation de dépendance avec aucune des n-1 autres variables. La DPC de chaque variable est générée aléatoirement en tenant compte des éventuelles relations de dépendance. Le nombre de modalités de chaque variable est k = 3. Dans notre étude, nous changeons les caractéristiques des réseaux pour analyser les conséquences de ces changements sur l'efficacité de l'algorithme MMPC. Ces changements incluent la distribution des variables indépendantes I, le nombre de variables dépendantes D et le nombre de dépendances parmi les D variables dépendantes (c'est-à-dire le nombre d'arcs A dans le réseau). Les résultats sont présentés dans les tables 1 à 3. Chaque ligne de ces tables représente une moyenne des résultats obtenus pour 10 ensembles d'exemples différents générés à partir de 10 réseaux différents mais possédant les même caractéristiques. Dans chaque expérimentation, nous calculons la moyenne et l'écart type du nombre de Vrais Positifs (VP), Faux Positifs (FP), Faux Négatifs (FN) et du temps de calcul. VP correspond au nombre de relations parent-enfant correctement prédîtes par l'algorithme sans tenir compte du sens de la dépendance. Donc, le nombre de VP peut être au maximum égal au double du nombre d'arcs du réseau. De la même façon, le nombre de FN, c'est-à-dire les arcs existants dans le réseau mais qui n'ont pas été prédits par l'algorithme, peut au maximum être égal au double du nombre d'arcs. La somme VP + FN est donc égale au double du nombre d'arcs du réseau. Le nombre de FP est le nombre de dépendances prédîtes par l'algorithme et qui n'existent pas dans le réseau.

2.3.1 Distribution des variables indépendantes

Dans cette section, nous étudions l'effet de la distribution des variables indépendantes sur l'efficacité de l'algorithme max-min. Les réseaux bayésiens utilisés dans cette section comportent I = 75 variables indépendantes et D = 25 variables dépendantes. La distribution utilisée pour générer les variables indépendantes varie de pratiquement constante à complètement uniforme. Nous représentons la distribution des variables indépendantes comme un triplet tel que (p1, p2 ,p3). Par exemple, (80, 10, 10) signifie que chaque variable aléatoire a une probabilité de 0.8 de prendre comme valeur l'une des trois modalités possibles, et une probabilité de 0.1 pour les deux autres. La modalité ayant une probabilité de 0.8 est choisie aléatoirement parmi les trois pour chaque variable aléatoire. Les résultats sont présentés dans la table 1. Le nombre d'arcs pour tous ces réseaux est A = 40. Chaque ligne du tableau correspond à un résultat moyen obtenu pour 10 réseaux différents mais de mêmes caractéristiques. On peut constater, à partir de ces résultats, que la distribution des variables indépendantes n'a pratiquement aucun effet sur l'efficacité de l'algorithme max-min. L'algorithme max-min, dans ces conditions, permet de découvrir environ 37% des dépendances. On peut remarquer également que le nombre de FP est élevé, ce qui signifie que l'algorithme a tendance à prédire beaucoup plus de dépendances qu'il n'en existe réellement.

Distribution des variables independantes	Moyenne des VP	DS des VP	Moyenne des FP	DS des FP	Moyenne des FN	DS des FN	Temps de calcul (s)	Précision= VP/(VP+FN)
(80, 10, 10)	30	7.29	116	11.33	49	7.28	6.5	37.5%
(50, 25, 25)	30	8.12	113	11.9	49	8.16	6.4	37.5%
(40, 30, 30)	29	7.28	117	11.63	51	7.28	6.7	36.25%
(33, 33, 33)	29	6.76	118	10.54	50	6.78	6.2	36.25%

TAB. 1 – *Efficacité de l'algorithme max-min en fonction de la distribution des variables indépendantes. Chaque ligne comporte la moyenne et la déviation standard (DS) du nombre de VP, de FP, de FN et du temps de calcul pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés. Chaque réseau possède 25 variables dépendantes, 75 variables indépendantes et 40 arcs.*

2.3.2 Proportion de variables dépendantes

Dans cette seconde expérimentation, nous avons fait varier les nombres D et I (n reste égal à 100). Nous avons également fait varier A de façon à conserver pratiquement constant le ratio A/D. Comme l'on peut le constater dans les résultats présentés dans la table 2, lorsque le réseau ne comporte que des variables dépendantes (D = 100), l'algorithme max-min a de bien meilleurs performances : plus de 80% des dépendances sont découvertes. En revanche, comme on peut le voir dans les deux premières lignes de la table, lorsque le nombre de variables dépendantes est égal à 25 ou 50, seules environ 35% des dépendances sont découvertes. Le nombre de FP est aussi très faible lorsque toutes les variables sont dépendantes. Cela semble donc indiquer que cette méthode a des difficultés à déterminer que certaines des variables sont indépendantes. On peut cependant remarquer que le temps de calcul augmente considérablement dans le cas où toutes les variables sont dépendantes. Cela peut s'avérer problématique lorsque le nombre de variables dans le problème est très supérieur a 100.

D	A	Moyenne des VP	DS de VP	Moyenne des FP	DS des FP	Moyenne des FN	DS des FN	Temps de calcul (s)	Précision = VP/(VP + FN)
25	40	29	6.76	118	10.54	50	6.78	0.31	36.2%
50	80	53.4	8.81	99.4	11.35	106.6	8.81	0.34	33.4%
100	150	243.8	8.17	16.6	6.81	56.2	8.17	21.1	81.3%

TAB. 2 – *Efficacité moyenne de l'algorithme max-min en fonction du nombre de variables dépendantes. Chaque ligne comporte la moyenne et la déviation standard (DS) du nombre de VP, de FP, de FN et du temps de calcul pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés. Chaque réseau possède 100 arcs.*

2.3.3 Complexité des réseaux

Dans cette section, nous étudions l'effet de la complexité du réseau sur l'efficacité de l'algorithme max-min. Nous faisons varier le nombre de variables n et le nombre d'arcs A. Toutes les variables sont dépendantes, donc D = n. Comme dans l'étude précédente, puisqu'il n'y a pas de variables indépendantes, le pourcentage de dépendances découvertes est élevé. En revanche, ce pourcentage diminue légèrement avec le nombre d'arcs et le nombre de variables. Il semble que la complexité du réseau ait moins d'importance que la mixité des variables (dépendantes et indépendantes). La complexité du réseau influe tout de même fortement sur le temps de calcul.

D	A	Moyenne des VP	DS des VP	Moyenne des FP	DS des FP	Moyenne des FN	DS des FN	Temps de calcul (s)	Précision = VP/(VP + FN)
25	30	52.4	3.55	2.4	1.96	7.6	3.55	0.31	87.3%
25	40	65.6	4.17	2.2	1.89	14.4	4.17	1.83	82%
25	60	91.8	4.51	3.2	3.37	28.2	4.51	6.63	76.5%
100	120	200.6	5.51	25.2	7.28	39.4	5.52	9.98	83.6%
100	150	243.8	8.17	16.6	6.81	56.2	8.17	21.1	81.3%
100	200	312.4	9.67	10.8	3.37	87.6	9.67	28.3	78.1%

TAB. 3 - *Efficacité moyenne de L'algorithme max-min en fonction de la complexité du réseau bayésien. Chaque ligne comporte la moyenne et la déviation standard (DS) du nombre de VP, de FP, de FN et du temps de calcul pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés.*

2.3.4 Problèmes classiques

Les réseaux que nous avons utilisés dans les expériences précédentes ont tous été générés aléatoirement. Pour évaluer l'efficacité de l'algorithme max-min sur des données provenant de problèmes réels, nous l'avons appliqué sur un ensemble de problèmes classiquement utilisés pour valider les algorithmes de construction de réseaux bayésiens, Binder et al. (1997), Jensen et Jensen (1996), Kristensen et Rasmussen (2002), Beinlich et al. (1989). Les résultats obtenus semblent meilleurs que ceux présentés dans les tables 1 et 2 et de qualité similaire à ceux présentés dans la table 3. Cela peut s'expliquer vraisemblablement par le fait que les problèmes réels comportent plus de régularités que ceux générés aléatoirement et que les dépendances sont de ce fait plus faciles à localiser.

Problèmes	VP	FP	FN	Temps de calcul (s)
insurance	70	2	34	137
halfinder	124	136	8	1200
barley	136	160	32	205
alarm	84	2	8	68

TAB.. 4 – *Résultats obtenus avec l'algorithme max-min pour quatre problèmes classiques. Chaque ligne contient le nombre de VP, de FP, de FN et le temps de calcul.*

2.4 Utilisation de l'algorithme max-min pour la « sélection de variables »

Nous avons évoqué dans l'introduction la possibilité d'utiliser les méthodes de détection des dépendances pour la sélection de variables. L'idée est de décomposer le problème initial en localisant les variables indépendantes (celles pour lesquelles Par_i est vide après application de l'algorithme max-min) pour lesquelles l'optimisation peut être réalisée indépendamment des autres variables. On sélectionne alors uniquement le sous-ensemble des variables pour lesquelles on soupçonne qu'il existe des liens de dépendances avec d'autres variables pour n'appliquer la méthode d'optimisation combinatoire que sur le sous-problème correspondant. L'espace de recherche ayant ainsi été réduit par cette étape préliminaire, les chances de découvrir une solution de bonne qualité s'en trouvent augmentées. Le problème est donc ici légèrement plus facile que celui étudié dans la section 2.3 car on cherche uniquement à déterminer la listes des variables impliquées dans des relations de dépendances sans vouloir découvrir précisément ces dépendances. Nous avons donc réalisé une série d'expériences pour mesurer les capacités de l'algorithme max-min sur ce problème réduit.

Nous avons utilisé des réseaux en utilisant la méthode présentée dans la section 2.3.1 pour la distribution des variables indépendantes. Pour la distribution (33, 33, 33) nous avons également fait varier la complexité du réseau en changeant le nombre d'arcs. Nous donnons les résultats de ces expérimentations dans la table 5. Chaque ligne représente la moyenne obtenue pour 10 réseaux différents mais possédant les mêmes propriétés. Il apparaît que bien que la méthode max-min puisse découvrir plus de 90% des variables dépendantes (les Vrai Positif de la table 5), elle ne découvre qu'environ 17% des variables indépendantes (les Vrai Négatifs de la table 5). Une méthode basique qui prédirait toutes les variables comme indépendantes aurait des résultats à peine moins bons. Cela signifie que cette méthode a tendance à surestimer fortement le nombre de dépendances. Les résultats sont peu affectés par les différentes distributions de variables indépendantes et par la complexité du réseau. Il semble donc que cette méthode ne puisse pas être utilisée pour le problème de la sélection de variables car pratiquement toutes les variables sont sélectionnées.

Distribution	D	A	VP	VN
(33, 33, 33)	25	60	24.2	11.4
(33, 33, 33)	25	40	23.2	12.4
(33, 33, 33)	25	30	24	10.6
(80, 10, 10)	25	40	23.8	12.6
(50, 25, 25)	25	40	23.8	13.6
(40, 30, 30)	25	40	23.8	13.8

TAB.. 5 – *Résultats obtenus par l'algorithme max-min pour le problème de la sélection de variables. Chaque ligne comporte la moyenne du nombre de VP et de VN pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés.*

3 Approche basée sur l'analyse statistique implicative

Nous nous intéressons aux capacités de la méthode d'Analyse Statistique Implicative pour la détection de dépendances multiples. Nous voulons étudier plus particulièrement si les spécificités de l'ASI, Gras et Kuntz (2008), Gras et al. (2004), permettant de prendre en considération les contre-exemples à une hypothèse d'implication, peuvent être utiles pour découvrir des dépendances dans des situations qui sont difficiles pour les méthodes classiques se basant sur des mesures de probabilités conditionnelles. Par exemple, une situation dans laquelle deux variables sont indépendantes mais prennent toutes les deux très souvent la même modalité dans un grand nombre d'exemples. Dans cette situation, une mesure basée sur la probabilité conditionnelle détectera une dépendance. Or si cette modalité est bien présente fréquemment simultanément pour les deux variables, il n'en reste pas moins que les occurrences des ces modalités sont décorrélées, puisque les variables sont indépendantes, donc la prise en compte du nombre de contre-exemples, même peu nombreux, pourrait permettre de réfuter l'hypothèse de dépendance.

Cependant, pour pouvoir utiliser l'ASI dans des situations très générales, certaines modifications sont nécessaires. En effet, nous ne voulons a priori pas nous limiter aux modalités binaires ni présupposer une relation d'ordre entre les différentes modalités d'une variable. Nous voulons également pouvoir détecter une situation où une conjonction de variables implique une autre variable et, dans ce cas, en tenir compte dans une mesure globale. C'est-à-dire que nous voulons pouvoir mesurer qu'une ou plusieurs combinaisons de modalités des variables parents impliquent une ou plusieurs modalités de la variable enfant. Par exemple, soit les variables A, B et C $\in \{0, 1, 2\}$. Nous voulons définir une mesure capable de détecter une dépendance de A vis-à-vis de B et C parce que : quand $B=0 \wedge C=2$ alors 'souvent' $A = 1$ et quand $B = 0 \wedge C = 0$ alors 'souvent' $A = 0$. La version actuelle de l'ASI ne permet pas de prendre en compte simultanément toutes les combinaisons des modalités des variables considérées pour produire un score unique déterminant le niveau de dépendance global de A vis-à-vis de B et C. Nous avons donc défini une extension à l'ASI pour prendre en compte ces différentes situations.

3.1 Définitions et notations

Ces définitions et notations s'ajoutent à celles présentées dans la section 2.1. Soit Card($m_{i,j}$) le nombre de fois où la variable v_i prend la valeur $m_{i,j}$ dans les N exemples. On note Card($m^-_{i,j}$) le nombre de fois où la variable v_i prend une valeur différente de $m_{i,j}$ et Card($m_{i1,j1} \wedge m_{i2,j2}$) le nombre de fois où la variable v_{i1} prend la valeur $m_{i1,j1}$ et la variable v_{i2} prend la valeur $m_{i2,j2}$ dans les N exemples.

On note π_i une instanciation de chacune des variables parents de v_i choisie dans la liste de modalités que chacune d'elles peut prendre et Π_i l'ensemble de toutes les combinaisons d'instanciations des variables parents de v_i. Par exemple, en reprenant l'exemple précédent avec les variables A, B et C, Π_A = {(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)}. Si on note k = $|M_i|$ alors pour tout $v_j \in Par_i$:

$$\left|\Pi_i\right| = k^{\left|Par_i\right|}.$$

Soit Card(π_i) le nombre de fois où l'ensemble des variables parents de v_i prend la valeur π_i dans les N exemples. La mesure q de l'ASI est alors donnée par :

$$\boldsymbol{q(\pi_i, m_{i,j})} = \frac{Card(\pi_i \wedge m^-_{i,j}) - \frac{Card(\pi_i) \times Card(m^-_{i,j})}{N}}{\sqrt{\frac{Card(\pi_i) \times Card(m^-_{i,j})}{N}}}$$

De la même façon, nous calculons la mesure de l'ASI $\tau(\pi_i, m_{i,j})$. Puis le score que nous cherchons à maximiser est donne par :

$$s(\pi_i, m_{i,j}) = -i(\pi_i, m_{i,j}) \times q(\pi_i, m_{i,j})$$

où i(.,.) est l'indice d'inclusion défini dans la partie 1, chap. 1, § 8.2.

3.2 Extension de l'ASI

Nous avons expérimenté plusieurs mesures permettant de prendre en considération les différents critères que nous avons énoncés plus haut. Pour une variable v_i nous voulons une mesure unique qui représente globalement son degré de dépendance vis-à-vis de son ensemble parent. Nous devons donc tenir compte de l'ensemble des combinaisons possibles $\prod_i$ et déterminer, en utilisant les mesures $s(\pi_i, m_{i,j})$, comment elles impliquent l'ensemble M_i des modalités de v_i. Nous construisons donc une table T_i, contenant l'ensemble $\prod_{si}$ des mesures de s pour toutes les combinaisons de π_i et de M_i et de taille :

$$k \times \left|\pi_i\right| = k^{\left|Par(v_i)\right|+1}$$

Nous avons essayé différentes méthodes pour combiner toute l'information provenant de cette table en une seule mesure. La méthode la plus simple pour cela est de considérer uniquement le maximum des $\prod_{si}$. D'autres possibilités sont de considérer la moyenne des $\prod_{qi}$, ou la moyenne des x% plus hauts scores de $\prod_{si}$. Nous avons réalisé de nombreux tests avec ces différentes approches et aucun n'a donné de résultats satisfaisants. Dans cette première série de mesures nous avons considéré indépendamment les scores obtenus pour

une même valeur de π_i mais différentes valeurs de M_i. Or, ce que l'on souhaite détecter c'est qu'une valeur de π_i, c'est-à-dire une instanciation particulière des variables parents de v_i, implique une instanciation particulière de v_i, et que ça soit le cas pour plusieurs instanciations différentes de π_i. Il faut donc une mesure qui permette de détecter que q est fort pour un couple (π_i, $m_{i,j}$) avec $m_{i,j} \in M_i$ et faible pour tous les autre $m_{i,j'} \in M_i$ et que cela soit vrai pour plusieurs π_i. Nous avons donc défini un score qui tient en compte, pour un π_i donné, de la valeur de s maximum $Sup_{\pi i}$ pour tout $m_{i,j} \in M_i$ et de l'entropie $E_{\pi i}$ de s pour toutes les valeurs $m_{i,j} \in M_i$.

Soit $$Sup_{\pi i} = \max_{1 \le j \le k}(s(\pi_i, m_{i,j}))$$

Et $$E_{\pi i} = -\sum_{j=1}^{k} \frac{p(s(\pi_i, m_{i,j}))\log(p(s(\pi_i, m_{i,j})))}{\log(k)}$$

Avec $$p(s(\pi_i, m_{i,j})) = \frac{q(\pi_i, m_{i,j})}{\sum_{j'=1}^{k} q(\pi_i, m_{i,j'})}$$

Pour calculer la mesure associée à une table T_i, on considère l'ensemble H, de taille h, des π_i correspondant aux x% des valeurs de $Sup_{\pi i}$ les plus élevées de la table. Le score de la table est alors :

$$S_{i,Par_i} = \frac{\sum_{\pi_i \in H} Sup_{\pi_i}}{\sum_{\pi_i \in H} E_{\pi_i}}$$

C'est cette mesure que l'on cherche finalement à maximiser.

En reprenant l'exemple avec les variables A, B et C, la table T_A correspond à :

B	C	A = 0	A = 1	A = 2	Sup	E
0	0	0	1.3	0.6	1.3	0.272
0	1	0	0	0	0	0
0	2	2.1	0	0.2	2.1	0.129
1	0	0	0	0	0	0
1	1	0.4	0.2	0.5	0.5	.45
1	2	1.1	0	0	1.1	0
2	0	0	0	0	0	0
2	1	0	0	0	0	0
2	2	0	0	0	0	0

TAB. 6 – *Exemple de calcul de la table T_i avec A, B et C $\in$ {0, 1, 2} et Π_A = {(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)}.*

Si on sélectionne les 20% Sup les plus élevés, seules les lignes 1 et 3 seront sélectionnées et c'est avec les valeurs de Sup et de E correspondantes qu'est calculée S_A qui est égale à

8.48 dans cet exemple. Plus la valeur de S est élevée plus on considère que la dépendance est forte. Dans la section suivant nous donnons un algorithme qui va utiliser cette mesure pour déterminer quelles sont les dépendances principales d'un problème.

3.3 Algorithme de détection des dépendances

Nous avons défini la mesure S_i, associée à chaque variable v_i. Il faut cependant connaître l'ensemble parent de v_i pour pouvoir la calculer. Pour déterminer les dépendances d'un problème il faut examiner différentes configurations possibles d'ensembles parents pour toutes les variables et choisir la configuration qui mène à un score total maximal. Cependant le nombre de configurations à examiner est exponentiel avec le nombre de variables. Il faut donc définir une heuristique pour construire les ensembles parents. Nous avons choisi une approche incrémentale pour cette heuristique. L'algorithme associé commence avec des ensembles parents vides pour chaque variable, puis à chaque étape une nouvelle variable est ajoutée dans l'un des ensembles parents. C'est la mesure S qui est utilisée pour choisir à chaque étape quelle variable ajouter dans quel ensemble parent. Ce processus est appliqué jusqu'à ce qu'un nombre total maxVariable, fixé à l'avance, de variables ait été ajouté. Le calcul de la table T_i étant lui aussi exponentiel en fonction du nombre de variables parents de v_i, nous utilisons des contraintes supplémentaires, limitant à quatre le nombre maximal de variables que peut comporter chaque ensemble parent et le nombre total de variables ajoutées maxVariable. L'algorithme utilisé est également glouton.

Le choix de la prochaine variable à ajouter dans un ensemble parent se fait en comparant le meilleur score de quatre différentes tables. L'algorithme 1, que nous avons défini permet d'éviter de calculer le score pour toutes les combinaisons ayant 2, 3 et 4 variables dans l'ensemble parent. Seules les combinaisons qui ont été sélectionnées avec x variables parents conduiront au calcul du score avec x + 1 variables parents. Dans cet algorithme, la variable structMax contient trois informations : le score de dépendance de la variable enfant vis-à-vis de ses variables parents, le numéro de la variable enfant et le numéro de la variable parent candidate à être ajoutée à l'ensemble parent. Après initialisation, la table Max_1 contient la liste triée de façon décroissante de tous les scores de toutes les combinaisons comportant une variable enfant et une variable parent. Il y a donc n^2 scores dans la table initialement. Les tables Max_2, Max_3 et Max_4 sont initialement vides. Elles servent à mémoriser les scores des combinaisons enfant-parents dans les cas respectifs où il y a 2, 3 et 4 variables dans l'ensemble parent. Ainsi, à chaque étape de l'algorithme, la variable à ajouter dans l'ensemble parent d'une autre variable sera déterminée en sélectionnant le score le plus élevé des 4 tables. Si c'est la table Max_i qui est sélectionnée, l'ensemble parent de la variable associée au score maximal de cette table passera de i-1 à i variables. Le score est alors retiré de la table max_i et un nouveau score est calculé et inséré dans la table max_{i+1}. Les quatre tables sont maintenues triées par ordre décroissant de telle manière que la valeur maximale de chaque table se trouve toujours en position 0.

```
Pour tout v_i
        Par_i = {∅}
    structMax = {0, 0, 0}
    max_1 = ∅
    pour tout v_i {
        pour tout v_j {
            si ( S_{i,Par_i +v_j} > structMax.score) {
                sturctMax.score = S_{i,Par_i +v_j}
                structMax.enfant = i
                structMax.parent = j
            }
        }
        max_1 = max_1 + structMax
    }
    TrieDecroissant(max_1)
    max_2 = ∅, max_3 = ∅, max_4 = ∅
    nbVariable = 0
    tant que (nbVariable < maxVariable) {
        k = maximum(max_1[0].score, max_2[0].score, max_3[0].score, max_4[0].score)
        enf = max_k[0].enfant
        par_enf = par_enf + max_k[0].parent
        si (k < 4) {
            structMax = {0, 0, 0}
            pour tout v_j ∉ par_enf {
                si ( S_{i,Par_i +v_j} > structMax.score) {
                    sturctMax.score = S_{i,Par_i +v_j}
                    structMax.enfant = i
                    structMax.parent = j
                }
            }
            max_{k+1} = max_{k+1} + structMax
            TrieDecroissant(max_{k+1})
            max_k[0] = {0, ∅, ∅}
            TrieDecroissant(max_k)
            nbVariable = nbVariables + 1
    }
```

Algorithme 1 – *Détermination des maxVariable ensembles parents par heuristique gloutonne.*

3.4 Evaluation de l'algorithme de détection des dépendances basé sur l'ASI

Pour réaliser la comparaison la plus honnête possible, nous avons réutilisé les mêmes expériences que celles de la section 2 pour évaluer notre algorithme de détection de dépendances basé sur l'ASI. Il faut cependant remarquer que cette procédure défavorise l'ASI. En effet, les données ont été générées à partir de modèles, des réseaux bayésiens, basés sur la mesure de probabilité conditionnelle qui est celle qui est utilisée également par l'approche max-min. Or, l'approche ASI utilise une autre mesure qui n'a pas les mêmes propriétés. En particulier, une différence très importante est que le modèle du réseau bayésien est non transitif alors que celui de l'ASI l'est. Mais une comparaison totalement équitable n'étant pas possible et en tenant compte de ces différences dans notre analyse, cela nous a paru être la meilleure façon de procéder.

3.4.1 Distribution des variables indépendantes

Nous avons repris les mêmes données que dans la section 2.3.1. Notre algorithme utilise plusieurs paramètres : le pourcentage x% des meilleurs Sup de chaque table T_i et le nombre maximal maxVariable de variables ajoutées dans les ensembles parents. Pour chacun d'eux nous avons expérimenté de nombreuses valeurs. Celles qui nous ont semblé les plus pertinentes et que nous présentons ici sont 10% et 50% pour x% et 35, 50 et 150 variables pour max variables. Pour ce dernier paramètre nous pouvons évaluer ainsi trois configurations importantes correspondant à une situation réelle dans laquelle on ne sait pas à l'avance combien il y a de dépendances dans le problème. Avec 35 variables on recherche moins de variables qu'il n'en existe réellement, avec 50 variables on en recherche légèrement plus et avec 150 on en recherche nettement plus qu'il n'en existe réellement.

Les résultats présentés dans les tables 7 et 8 confirment que notre algorithme ne découvre que peu de dépendances. La mesure utilisée semble toutefois plus sensible à la distribution utilisée pour générer les variables indépendantes. Les résultats obtenus avec la valeur x% = 10% sont cependant légèrement meilleurs. Le temps de calcul est également plus élevé qu'avec l'algorithme max-min mais notre programme n'a pas encore été optimisé pour cela.

X% = 10%	Moyenne des VP	Moyenne des FP	Moyenne des FN	Précision =VP/(VP+FN)	Temps de calcul (s)
(80, 10, 10) 35 var	0.9	34.1	39.1	2.25%	37
(50, 25 ,25) 35 var	6.7	28.3	33.3	16.7%	61.7
(40, 30, 30) 35 var	7.8	27.2	32.2	19.5%	69.3
(33, 33, 33) 35 var	0.6	34.4	39.4	1.5%	44.6
(80, 10, 10) 50 var	1	49	39	2.25%	46.1
(50, 25 ,25) 50 var	8.4	41.6	31.6	21%	76.4
(40, 30, 30) 50 var	11	39	29	27.5%	89.8
(33, 33, 33) 50 var	1.2	48.8	38.8	3%	57.4
(80, 10, 10) 150 var	1.2	148.8	38.8	3%	63.6
(50, 25 ,25) 150 var	12.3	137.7	27.7	30.7%	179
(40, 30, 30) 150 var	15.3	134.7	24.7	38.2%	184
(33, 33, 33) 150 var	4.9	145.1	36.1	12.2%	194

TAB. 7- *Efficacité de notre algorithme en fonction de la distribution des variables indépendantes lorsque x% = 10%. Le paramètre maxVariable prend successivement les valeurs 35, 50 et 150 variables. Chaque ligne comporte la moyenne du nombre de VP, de FP, de FN et du temps de calcul pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés. Chaque réseau possède 25 variables dépendantes, 75 variables indépendantes et 40 arcs.*

X% = 50%	Moyenne des VP	Moyenne des FP	Moyenne des FN	Précision = VP/(VP + FN)	Temps de calcul (s)
(80, 10, 10) 35 var	0.2	34.8	39.8	0.5%	33.8
(50, 25 ,25) 35 var	7.7	27.3	32.3	19.25%	59.7
(40, 30, 30) 35 var	5.1	29.9	34.9	12.7%	66.4
(33, 33, 33) 35 var	0.4	34.6	39.6	1%	36.9
(80, 10, 10) 50 var	0.2	49.8	39.8	0.5%	42.3
(50, 25 ,25) 50 var	6.2	43.8	33.8	15.5%	70.2
(40, 30, 30) 50 var	7.1	42.9	32.9	17.7%	80.9
(33, 33, 33) 50 var	0.5	49.5	39.5	1.25%	41.3
(80, 10, 10) 150 var	0.3	149.7	39.7	0.75%	55.5
(50, 25 ,25) 150 var	6.6	143.3	33.4	16.5%	164.2
(40, 30, 30) 150 var	8	142	32	20%	177.9
(33, 33, 33) 150 var	4.2	145.8	36.8	10.5%	140.9

TAB. 8 - *Efficacité de notre algorithme en fonction de la distribution des variables indépendantes lorsque x% = 50%.. Le paramètre maxVariable prend successivement les valeurs 35, 50 et 150 variables. Chaque ligne indique la moyenne du nombre de VP, de FP, de FN et du temps de calcul pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés. Chaque réseau possède 25 variables dépendantes, 75 variables indépendantes et 40 arcs.*

3.4.2 Problème de la « sélection de variables »

Nous avons réutilisé les mêmes jeux de données pour tester la capacité de notre algorithme à résoudre le problème de la sélection de variables. Les résultats présentés dans les tables 9 et 10 montrent un très fort potentiel de notre algorithme pour ce problème. Les résultats obtenus sont en effet bien meilleurs que ceux obtenus avec l'algorithme max-min. Si le nombre de VP est légèrement plus faible, le nombre de VN est considérablement plus élevé. Ce qui est particulièrement important c'est le constat que le niveau de prédiction est ici bien meilleur à ce que l'on pourrait attendre du hasard. Comme le ratio du nombre de variables dépendantes sur le nombre de variables indépendantes est de 1/3 dans le modèle utilisé pour générer les données, une prédiction aléatoire donnerait le même ratio de VP/FP (c'est-à-dire donc ce cas VP/(75-VN)). Nous présentons dans la colonne VP/(0.33xFP) des tables 9 et 10, le gain par rapport à une sélection aléatoire des variables dépendantes. Dans les cas de distributions des variables indépendantes (40, 30, 30) et (50, 25, 25) le gain est très important allant jusqu'à 16.1. A titre de comparaison, les résultats de l'algorithme max-min montre plus de stabilité mais un gain qui ne dépasse jamais 1.18. Notre algorithme semble avoir plus de difficulté lorsque les variables indépendantes ont des distributions extrêmes (33, 33, 33) ou (80, 10, 10). Avec x%=10% et dans le cas où l'on recherche moins de dépendances qu'il en existe (35 variables) le gain est toujours d'au moins 1. Bien que ce soit une première version, notre algorithme semble donc avoir un très fort potentiel pour détecter les variables dépendantes et donc résoudre le problème de la sélection de variables. Nous avons également testé notre algorithme sur les jeux de données présentées en section 2.3.2 dans lesquels D = 50, I = 50 et A = 80 (résultats non présentés ici). Les résultats montrent qu'avec la configuration x% = 10%, le gain est compris entre 1.28 et 1.82.

X% = 10%	Moyenne des VP	Moyenne des VN	VP/(0.33xFP)
(80, 10, 10) 35 var	6.7	55.5	1.03
(50, 25 ,25) 35 var	15.5	71.8	14.7
(40, 30, 30) 35 var	15.4	72.1	16.1
(33, 33, 33) 35 var	3.4	64.7	1
(80, 10, 10) 50 var	6.8	46.4	0.73
(50, 25 ,25) 50 var	18.3	68.7	8.79
(40, 30, 30) 50 var	18.3	69.7	10.5
(33, 33, 33) 50 var	6.1	61.3	1.36
(80, 10, 10) 150 var	9.2	13	0.45
(50, 25 ,25) 150 var	22.8	31.8	1.6
(40, 30, 30) 150 var	21.8	46.7	2.33
(33, 33, 33) 150 var	17.3	32.6	1.24

TAB. 9 - *Résultats obtenus par notre algorithme pour le problème de la sélection de variables avec le paramètre x% = 10%. Chaque ligne comporte la moyenne du nombre de VP et de VN pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés. Les résultats sont présentés pour trois valeurs du paramètre maxVariable : 35, 50 et 150 variables.*

X% = 50%	Moyenne des VP	Moyenne des VN	VP/FP
(80, 10, 10) 35 var	7.6	60.4	1.58
(50, 25 ,25) 35 var	16.8	66.2	5.79
(40, 30, 30) 35 var	14.8	67	1.85
(33, 33, 33) 35 var	4	57	0.67
(80, 10, 10) 50 var	8	53.5	1.13
(50, 25 ,25) 50 var	18.3	59.9	3.67
(40, 30, 30) 50 var	18.1	63.7	4.85
(33, 33, 33) 50 var	5.5	50	0.67
(80, 10, 10) 150 var	11.8	6.7	0.52
(50, 25 ,25) 150 var	22.1	22.8	1.28
(40, 30, 30) 150 var	22	41.2	1.97
(33, 33, 33) 150 var	22	16.5	1.14

TAB. 10 - *Résultats obtenus par notre algorithme pour le problème de la sélection de variables avec le paramètre x% = 50%. Chaque ligne comporte la moyenne du nombre de VP et de VN pour dix exécutions de l'algorithme sur des données générées à partir de dix réseaux différents mais présentant les mêmes propriétés. Les résultats sont présentés pour trois valeurs du paramètre maxVariable : 35, 50 et 150 variables.*

4 Conclusion

Nous avons réalise une étude sur la capacité de découverte des dépendances d'un problème de deux méthodes basées sur des mesures différentes. La première, l'algorithme max-min repose sur le test de dépendance conditionnelle G^2. La deuxième est un algorithme que nous avons conçu basé sur une extension de la mesure ASI. Nous avons appliqué ces algorithmes à de nombreux jeux de données en faisant varier les paramètres du problème tels que la distribution des variables indépendantes, le nombre de variables dépendantes et le nombre de dépendances. Nous avons également considéré deux problèmes différents : déterminer quelles sont les dépendances et déterminer quelles sont les variables impliquées dans des relations de dépendance. Bien entendu être capable de résoudre le premier problème permet de résoudre également le second. Il n'est cependant généralement pas possible de résoudre directement et intégralement ce problème. Avoir la possibilité de découvrir dans un premier temps uniquement quel est le sous-ensemble des variables concernées par des relations de dépendance permet de réduire la complexité du premier problème et donc de fournir une solution de meilleure qualité.

Nos résultats ont montré une bonne efficacité pour l'algorithme max-min pour découvrir les dépendances lorsque toutes les variables du problème sont impliquées dans des relations de dépendance. L'algorithme semble peu affecté par la variation de complexité du modèle et par les différentes distributions des variables indépendantes. Il a cependant d'importantes limitations pour détecter les dépendances lorsqu'une partie des variables sont indépendantes. L'algorithme max-min ne semble pas non plus être efficace pour le deuxième problème : la sélection de variables. Notre algorithme, basé sur l'ASI, en revanche ne semble pas capable de détecter directement les dépendances et ce quelles que soient les configurations. Il paraît cependant très efficace pour déterminer quelles sont les variables indépendantes et quelles

sont les variables dépendantes. Il a toutefois plus de difficulté dans les situations où les variables indépendantes ont des distributions extrêmes (80, 10, 10) ou (33, 33, 33).

Les deux approches paraissent donc complémentaires et prometteuses. Il serait très intéressant de développer une méthode combinant ces deux approches. Dans une première phase notre algorithme, utilisant la version étendue de l'ASI, permettrait de sélectionner un sous-ensemble des variables pour lesquelles il y a une forte présomption de dépendances. Puis, dans une deuxième phase, l'approche max-min serait appliquée à ce sous-ensemble pour déterminer plus précisément où sont ces dépendances. L'ensemble de ces informations permettrait alors de construire un réseau bayésien modélisant bien l'ensemble des exemples disponibles et donc pouvant être utilisé pour résoudre le problème d'optimisation associé. Il serait intéressant également d'étudier d'autres variantes de l'extension de la mesure ASI pour voir dans quelle mesure il serait possible d'améliorer les résultats obtenus pour la détection directe de dépendances.

Références

Beinlich I.A., H.J. Suermondt, R.M. Chavez, G.F. Cooper (1989) The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Second European Conference in Artificial Intelligence in Medicine,* 247-256

Binder J., D. Koller, S. Russell, K. Kanazawa (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning,* 29(2-3):213-244

Friedman N., I. Nachman et D. Peèr (1999). Learning bayes network structure from massive datasets: The "sparse candidate" algorithm. *15th Conference on Uncertainty in Artificial Intelligence* 206-215.

Garey M. R. et D. S. Johnson (1979). *Computers and Intractability: a Guide to the Theory of NP-Completeness.* Freeman.

Goldebberg A. et A. Moore (2004). Tractable learning of large Bayes net structures from sparse data. *21ft International Conference on Machine Learning* 44-51

Gras R., R. Couturier, J. Blanchard, H. Briand, P. Kuntz, P. Peter (2004). Quelques critères pour une mesure de qualité de règles d'association. Un exemple : l'implication statistique, *Mesures de qualité pour la fouille de données, RNTI-E-1, Cepaduès – Editions,* 3-32

Gras R. et P. Kuntz (2008). An overview of the Statistical Implicative, *Statistical Implicative Analysis,* R.Gras, E. Suzuki, F.Guillet and F.Spagnolo, Eds, Springer-Verlag.

Jensen A.L., F.V. Jensen (1996). MIDAS: An Influence Diagram for Management of Mildew in Winter Wheat. In *Proceedings of 12th Annual Conference on Uncertainty in Artificial Intelligence,* 349-356

Koivisto M. et K. Sood (2004). Exact Bayesian structure discovery in Bayesian networks, *Journal of Machine Learning Research* 5:549-573

Kristensen K., I.A. Rasmussen (2002). The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture,* 33: 197–217

Larranaga P. et J.A. Lozano (2002). *Estimation of Distribution Algorithms: A new tool for evolutionary computing*: Kluwer Academic Publishers.

Muhlenbein H. et T. Mahnig (2001). Evolutionary Algorithms: From Recombination to Search Distributions. *Theoretical Aspects of Evolutionary Computing*: Springer Verlag, 135-173.

Neapolitan, R. (2003) *Learning Bayesian networks*. Prentice Hall

Papadimitriou C. H. et K. Steiglitz (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover.

Spirtes P., C. Glymour et R. Scheines (2000). *Causation, prediction, and search*. The MIT Press, second Edition

Thompson R. K. et A. H. Wright (1996). *Additively decomposable fitness functions.* University of Montana, Computer Science department.

Tsamardinos I., L. E. Brown et C. F. Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78.

Zeng Y. et C. Hernandez (2008). A Decomposition Algorithm for Learning Bayesian Network Structures from Data. PAKDD 2008, *Lecture Notes in Artificial Intelligence* 5012:441–453.

Summary

Searching for dependencies between variables using a set of examples is an important problem in combinatorial optimization. Numerous methods have been proposed to solve this problem but few large scale evaluations have been performed. Most of these methods rely on conditional probability measures. The SIA propose another point of view on dependency measurement. It is therefore important to compare the results obtained using this approach with the ones obtained with one of the currently most efficient method for this task: the max-min heuristic. However, the SIA can not be used directly for this problem. We have therefore defined a dedicated extension to this measure. We have performed a large number of experiments varying several parameters like the number of dependencies, the number of variables or the goal of the prediction task. The results show a strong synergy of the two methods.