Sommaire

Chapitre II : Réseaux bayésiens

Liste des figures

Liste des tableaux

Résumé

L'objectif de ce projet est d'améliorer la maintenance des systèmes OO, et d'intervenir plus précisément dans la tâche de l'analyse et la prédiction de l'impact du changement. Parmi plusieurs modèles de représentation, les Réseaux Bayésiens (RBs) constituent une approche quantitative particulière qui peut intégrer l'incertitude dans le raisonnement offrant ainsi des explications proches à la réalité. De plus, avec les RBs, il est aussi possible d'exploiter les jugements des experts pour anticiper les prédictions, dans notre cas sur l'impact de changement. Nous nous intéressons en premier lieu à l'aspect inférence puis éventuellement a l'aspect apprentissage par les RBs

Nous proposons une approche probabiliste afin de déterminer l'impact des changements dans les systèmes à objets. Cette approche sert à prédire, pour un changement donné dans un système à objet, l'ensemble des effets de ce changement donnée sous la forme de probabilité.

Un des principaux avantages liés à l'utilisation des réseaux bayésiens est l'inférence qui permet de simuler le comportement du système en fonction du jeu de données en entrée. Dans le cadre de cette étude, nous avons implémenté l'algorithme d'inférence « Arbre de jonction», puis nous l'avons utilisé afin d'améliorer l'efficacité de l'impact de changement.

Mots-clés : Analyse d'impact de changement, maintenance, modèle probabiliste, réseaux bayésiens, inférence bayésienne.

Abstract

The objective of this project is to improve the maintenance of OO systems and intervene precisely in the analysis and prediction of change impact. Among several models of representation, Bayesian Networks constitute a particular quantitative approach which can integrate uncertainty in the reasoning so offering explanations close to reality. Furthermore, with bayesian networks it's also possible to exploit expert judgments in order to anticipate the prediction; in our case the impact of change .We are interested first of all in inference aspect then possibly in apprenticeship aspect by the bayesian networks.

We suggest a probability approach in order to determine the impact of changes in object-oriented systems. This approach is used to predict for a change given in an object system, the overall effects of this change is given under the shape of a probability.

One of the main advantages of using bayesian networks is the inference that can simulate the behavior of the system depending on the input data set. In this study, we have implemented the inference algorithm "junction tree", then we used to improve the effectiveness of the impact of change.

Keywords: Impact of change analysis, maintenance, probability model, Bayesian networks, bayesian inference.

Introduction Générale

Introduction générale

Avec le temps les logiciels informatiques sont devenus plus complexes et leur maintenance nécessite plus de temps et de coûts, c'est pour cela que l'industrie de l'informatique doit élaborer des logiciels plus efficaces et de très bonne qualité afin de diminuer les coûts de maintenance qui sont devenus au fil du temps plus couteux en raison de la grande importance de cette phase dans le cycle de vie du logiciel, car en effet la maintenance est la phase la plus longue et elle ne s'achève qu'avec la fin de vie du logiciel.

Dans l'industrie, le coût de maintenance d'un logiciel (par rapport à son budget total) est passé de 40 à 60% dans les années 80, à plus de 75% ou même 80% au début de l'an 2000 [19]. On estime également que plus de la moitié de cette maintenance est consacrée à la compréhension du programme lui-même. Ce coût élevé de la maintenance et surtout de la compréhension de programmes, pousse les chercheurs et les industriels à se focaliser sur cette phase du cycle de vie d'un système afin d'essayer de comprendre les facteurs qui influent ce coût.

D'autre part, la programmation OO a été définie de manière à modéliser des entités du monde réel, c'est-à-dire, la définition d'objets correspondants à des entités (contrairement à la programmation procédurale qui définit une suite de fonctions censées représenter des traitements sur des données). Chaque objet est considéré comme une entité qui possède ses propres caractéristiques, et qui a un comportement spécifique décrit par la liste des méthodes qu'il utilise.

La modification des systèmes est une tâche à la fois difficile et porteuse de conséquence sur la suite de l'évolution de ces systèmes. Les effets des changements subis par le système doivent donc être pris en considération. En effet, un petit changement peut avoir des effets considérables et inattendus sur le reste des éléments du système. Le danger encouru lors de la modification réside dans cette conséquence de l'impact d'un changement donné. La modularité en conception objet, adéquatement utilisée, limite les effets relatifs aux changements.

L'objectif de ce mémoire est d'améliorer la maintenance des systèmes à objets et d'intervenir plus précisément dans la tâche de l'analyse et de la prédiction de l'impact du changement. En

identifiant l'impact potentiel d'une modification, on réduit le risque d'entamer des changements coûteux et imprévisibles. Pour cela, nous essayons de donner plus d'explications sur les facteurs réels et responsables de cet impact du changement ainsi que son évolution. Il existe plusieurs modèles de représentation et les réseaux bayésiens (RBs) constituent une approche quantitative particulière qui peut intégrer l'incertitude dans le raisonnement (Naïm et al.2004), offrant ainsi des explications proches de la réalité. De plus, avec les RBs, il est aussi possible d'exploiter les jugements des experts pour anticiper les prédictions, qui dans notre cas portent sur l'impact du changement.

Le reste de ce mémoire est organisé comme suit; le premier chapitre montre une idée générale sur l'impact de changement, le deuxième chapitre est consacré aux réseaux bayésiens et aux différents algorithmes bayésiens d'inférence et d'apprentissage, puis on va montrer quelques travaux de recherche d'impact de changement et les outils bayésiens les plus connus et utilisés. Le troisième et quatrième chapitre donnent la conception détaillée du système et sa réalisation, et on conclu enfin par une conclusion générale et quelques perspectives majeures.

Chapitre I :

Impact de changement

I.1 Introduction

Le terme génie logiciel a été introduit en 1968 pour nommer une conférence traitant des problèmes liés au développement de logiciels.
Dans les années 70, elle s'est popularisée. Jusqu'à ce que les logiciels étaient essentiellement développés de manière "artisanale" (phase 1 du modèle). Des problèmes de production (phase 2) ont mis à jour le besoin de nouvelles solutions et la nécessité de s'organiser pour mettre en place une réelle industrie du logiciel (phase 3).

En 1976 Boehm [63] propose une définition de Génie logiciel dont une traduction pourrait être : "*appliquer des connaissances scientifiques à la production de programmes et de la documentation nécessaire à leur développement, mise en œuvre et maintenance*". Cette définition est cohérente avec le terme "Ingénierie". Cependant Boehm faisait immédiatement remarquer que peu de connaissances scientifiques étaient disponibles. Bien que des progrès très importants aient été faits depuis (l'utilisation de méthodes formelles par exemple), la construction de logiciel reste un savoir-faire. *Actuellement le terme ingénierie ne correspond pas à la réalité industrielle du logiciel*. [1]

Les efforts de développement de logiciel aboutissent à la livraison d'un produit logiciel qui satisfait les exigences des utilisateurs. Par la suite, le produit logiciel doit changer et évoluer. Une fois en opération, des défauts sont découverts, les environnements d'opération changent, et de nouvelles exigences de la part des utilisateurs font surface. La phase de maintenance du cycle de vie débute après une période de garantie ou de support après livraison.

Les plus activités coûteuses dans la maintenance de logiciels sont la compréhension du problème qui est généralement liée à la compréhension du logiciel maintenu, et la maîtrise de la totalité des effets de propagation des changements proposés. En effet, un petit changement peut avoir des effets considérables et inattendus sur le reste du système.

La modification des systèmes est une tâche à la fois difficile et porteuse de conséquence sur la suite de l'évolution de ces systèmes. La maintenance du logiciel soutient le produit logiciel tout au long de son cycle de vie opérationnel. Les demandes de modification sont enregistrées et suivies, l'impact des changements proposés est déterminé, le code est modifié, les essais sont

conduits, et une nouvelle version du produit logiciel est mise en production. Et aussi, la formation et le support quotidien sont fournis aux utilisateurs. Pfleeger [2] énonce que « la maintenance a une portée plus large, avec plus à suivre et à contrôler ». Le danger encouru lors de la modification réside dans cette conséquence de l'impact d'un changement donné. La modularité en conception objet, adéquatement utilisée, limite les effets relatifs aux changements. Néanmoins, en général, ces effets sont subtils et difficiles à découvrir. Pour toutes ces raisons, les concepteurs ont besoin de mécanismes pour analyser les changements et connaître leurs impacts sur le reste du système. [3]

I.2 Définitions et terminologie

I.2.1 Définition 1

La maintenance du logiciel est définie dans la norme IEEE pour la maintenance du logiciel, IEEE 1219, comme la modification d'un logiciel après la livraison pour corriger les fautes, pour améliorer la performance ou les autres attributs ou pour adapter le produit à un environnement modifié. [65]

I.2.2 Définition 2

La norme ISO/IEC 12207 pour les processus de cycles de vie décrit essentiellement la maintenance comme l'un des processus primaires du cycle de vie, et décrit la maintenance comme le processus d'un produit logiciel subissant « une modification du code et de la documentation associée due à un problème ou au besoin d'amélioration. L'objectif est de modifier le produit logiciel existant tout en préservant son intégrité » [66].

Un survol des définitions proposées pour la maintenance du logiciel est présenté au tableau suivant :

Définition - Interprétation	Auteurs des références	Année
« Les changements qui doivent être effectués à un logiciel après sa livraison à l'utilisateur. »	Martin et McClure [Mar83]	1983
« L'ensemble des activités requises afin de garder le logiciel en état d'opération suite à sa livraison opérationnelle. »	FIPS [FIP84]	1984
« La maintenance couvre le cycle de vie du logiciel à partir de son installation jusqu'à sa mise à la retraite. »	Von Mayrhauser [Von90]	1990
« La modification d'un logiciel, après sa livraison, afin de corriger des défaillances, d'améliorer sa performance ou d'autres attributs ou de l'adapter suite à des changements d'environnements. »	IEEE 610.12 [IEEE90]	1993
« Les changements au logiciel et à sa documentation causés par un problème ou le besoin de l'améliorer. »	ISO12207 [ISO95]	1995
« La totalité des activités qui sont requises afin de procurer un support, au meilleur coût possible, d'un logiciel. Certaines activités débutent avant la livraison du logiciel, donc pendant sa conception initiale, mais la majorité des activités ont lieu après sa livraison finale (l'équipe de développement ayant maintenant terminé son travail et étant affectée à d'autres travaux). »	SWEBOK [Abr05, s3.1.1]	2005

Tab I.1 Quelques définitions de la maintenance du logiciel

I.3 Le besoin de maintenance

La maintenance est nécessaire pour assurer que le logiciel continue à satisfaire les exigences des utilisateurs. La maintenance est applicable à des logiciels développés en utilisant n'importe quel modèles de cycle de vie (par exemple : le modèle en spirale). Le système change à cause des actions logicielles correctives ou non correctives.

La maintenance doit être effectuée dans le but de :

- Corriger les défauts ;
- Corriger les failles dans les exigences et la conception ;
- Améliorer la conception ;
- Implanter les améliorations ;
- Interfacer avec d'autres systèmes ;
- Adapter les programmes de façon à ce que du matériel, du logiciel, des fonctionnalités du système, et des possibilités de communications différentes puissent être utilisées ;
- Faire migrer des systèmes hérités ;
- Mettre le système à la retraite.

Il y a quatre activités clefs pour le mainteneur, selon Pfleeger [02] :

- Maintenir le contrôle sur les fonctions du logiciel au jour le jour ;
- Maintenir le contrôle sur les modifications du logiciel ;

- Perfectionner les fonctions existantes ;
- Empêcher les performances du système de se dégrader jusqu'à des niveaux inacceptables.

1.4 Les catégories de maintenance

Lientz & Swanson [64], identifièrent initialement trois catégories de maintenance : corrective, adaptative et perfective. Elles ont été mises à jour, et l'Organisation Internationale pour la Standardisation (ISO) a défini une nouvelle catégorie dans la norme pour le génie logiciel/-la maintenance du logiciel, Les catégories de maintenance définies par ISO/IEC sont les suivantes

1.4.1 Maintenance corrective : modification réactive d'un produit logiciel effectuée après la livraison pour corriger un problème découvert.

1.4.2 Maintenance adaptative : modification d'un produit logiciel faite après la livraison pour maintenir un produit logiciel utilisable dans un environnement changé ou changeant.

1.4.3 Maintenance perfective : modification d'un produit logiciel après sa livraison pour améliorer la performance ou la maintenabilité.

1.4.4 Maintenance préventive : modification d'un produit logiciel après sa livraison pour détecter ou corriger des défauts latents dans le produit logiciel avant qu'ils ne deviennent des défaillances.

ISO 14764, [67] classe la maintenance adaptative et perfective comme des améliorations. Elle regroupe aussi ensemble les catégories de la maintenance corrective et préventive en une catégorie 'correction', tel que montré dans le Tableau I.2. La maintenance préventive, la plus nouvelle des catégories, est faite le plus souvent sur des produits logiciels lorsque la sûreté est critique.

Correction	Amélioration
Préventive	Perfective
Corrective	Adaptive

Tab I.2 Catégories de maintenance du logiciel de l'ISO14764-00 [69]

1.5 Le processus de la maintenance

Le processus de maintenance est mis en œuvre lorsque le système subit des modifications relatives aux codes et à la documentation correspondante. Ces modifications peuvent être dues à

des erreurs, des défauts, des problèmes, des spécifications des besoins trop sommaires ou incomplètes ou encore à des besoins d'amélioration ou d'adaptation du système.L'objectif est de modifier un système existant tout en préservant son intégrité. Ce processus inclut la migration et le retrait du logiciel (dernière étape). Les activités liées à ce processus sont les suivantes :

- Indication du mode et des conditions de détection du problème.

Le développeur doit reproduire et vérifier l'existence du problème lorsqu'il s'agit d'une correction du logiciel.

- A partir de cette analyse, les actions seront menées pour la mise en œuvre des modifications.
- Identification des constituants logiciels touchés par les modifications (programme de base, données, structures, éléments de configuration).
- Mise en œuvre du processus de maintenance (développement, documentation, mise en œuvre, établissement des procédures permettant de recevoir et suivre les rapports d'anomalies et les demandes de modification des utilisateurs, enregistrement des problèmes pour des actions correctives, gestion de configuration, documents de suivi).
- Analyse des problèmes et des modifications : impact sur l'organisme et le système ou les interfaces existants (étendue des modifications, coût impliqué, délais de modification), type de modification (correction, amélioration, prévention, adaptation à un nouvel environnement), criticité (impact sur les performances, sûreté, sécurité).
- Le problème ou la demande de modification, les résultats de l'analyse, les options de mise en œuvre doivent être documentés.
- L'option de modification choisie devra être approuvée après détermination des codes et de la documentation à modifier.
- Mise en œuvre des modifications après analyse détaillée : le processus de développement doit être respecté, les critères de test et d'évaluation des éléments modifiés et non modifiées doivent être définis et documentés (composants logiciels, unités logicielles, éléments de configuration), la mise en œuvre complète et correcte des exigences nouvelles et modifiées doit être garantie (les exigences d'origine non modifiées ne doivent pas être affectées), les tests doivent être documentés.

- Des revues doivent être organisées afin de garantir que le système modifié est conforme et pourra être validé.

- Un plan de migration sera développé (analyse des exigences et définition de la migration, développement des outils de migration, conversion des logiciels et des données, exécution de la migration, vérification de la migration, support de l'ancien environnement dans le futur).
- Les utilisateurs du système doivent être avisés des plans et des activités de migration (raison de la migration, description du nouvel environnement et date de disponibilité, description des autres options de support disponibles, une fois que le support aura été retiré).
- Exploitation parallèle dans les environnements anciens et nouveaux en vue d'une transition douce, avec une formation utilisateur.
- Au moment de la migration, une notification est envoyée à toutes les personnes concernées, la documentation et les anciens codes et enregistrements sont mis en archive.
- Une revue après migration permettra d'évaluer l'impact des changements (les résultats seront envoyés aux autorités concernées).
- Les données utilisées ou associées à l'ancien environnement resteront accessibles (protection des données, audit applicable aux données).
- Le logiciel sera retiré à la demande du propriétaire, un plan de retrait sera fourni pour arrêter l'assistance effective (exploitation, maintenance) : interruption partielle ou totale de l'assistance après une période déterminée, archivage du logiciel et de sa documentation, responsabilité concernant les futurs problèmes résiduels d'assistance, transition éventuelle vers le nouveau projet. Il est souhaitable de préparer un plan de maintenance (approuvé par le client et le fournisseur) qui :

- précise le champ d'application de la maintenance,
- identifie l'état initial du produit,
- met en place une organisation de soutien (identification des installations et ressources, gestion des problèmes imprévus, priorités), indique les procédures de modification (les mêmes que celles utilisées pendant le développement),

- permet d'enregistrer toutes les activités de maintenance (liste des incidents, demandes d'assistance, actions correctives, responsabilités, priorités, résultats',
- définit les procédures applicables aux nouvelles versions (remise à jour complète incidence des nouvelles versions sur l'exploitation, annonce des mises à jour de nouvelles versions, sites et produits mis à jour, etc.) [4].

I.6 Mesure de la maintenance du logiciel

Grady et Caswell [68] discutent la mise en place d'un programme corporatif de la mesure du logiciel, dans lequel sont décrits les formulaires et la collecte des données pour la mesure de la maintenance du logiciel. Le projet « Practical Software and SystemsMeasurement(PSM) » [68] décrit un processus orienté vers la résolution de problématiques, lequel processus est utilisé par plusieurs organisations et est orienté vers la pratique.

Il y a des mesures du logiciel qui sont communes à tous les domaines, incluant les catégories suivantes identifiées par le Software Engineering Institute - SEI: la taille, l'effort, l'échéancier et la qualité. Ces mesures constituent un bon point de départ pour le mainteneur.

I.6.1. Mesures spécifiques

Abran [5] présente des techniques d'étalonnage interne pour comparer différentes organisations internes de maintenance. Le mainteneur doit déterminer quelles mesures sont appropriées pour l'organisation en question.

Il existe des mesures suggérées qui sont plus spécifiques aux programmes de prise de mesures sur la maintenance du logiciel. Cette liste inclue un nombre de mesures standards pour chacune des quatre sous-catégories de maintenabilité :

Analysabilité : mesure de l'effort du mainteneur ou des ressources utilisées pour essayer de diagnostiquer les déficiences ou les causes des pannes ou pour identifier les parties à modifier.

Changeabilité : mesure de l'effort du mainteneur associé à l'implantation d'une modification spécifiée.

Stabilité: mesure les comportements inattendus du logiciel, incluant ceux rencontrés lors des essais.

Essaiabilité: mesure de l'effort du mainteneur et de l'utilisateur pour essayer de faire des essais sur le logiciel modifié. La prise de mesures de la maintenabilité d'un logiciel peut être effectuée en utilisant des outils commerciaux [6].

I.7 Analyse d'impact du changement

L'analyse d'impact décrit comment mener, à un coût efficace, une analyse complète d'impact d'un changement dans un logiciel existant. Les mainteneurs doivent posséder une connaissance intime de la structure et du contenu du logiciel. Ils utilisent ce savoir pour faire l'analyse d'impact, qui identifie tous les produits des systèmes et des logiciels affectés par une demande de modification de logiciel et établir une estimation des ressources nécessaires pour accomplir le

changement. De plus, le risque à faire le changement est déterminé. La demande de changement, parfois appelée demande de modification (MR) et aussi appelée rapport de problème (PR), doit d'abords être analysée et traduite en termes logiciels. Elle est faite après qu'une demande de modification soit entrée dans le processus de gestion de configuration.

Les objectifs d'analyse de l'impact sont : [54]

- Détermination de la portée d'un changement pour établir un plan et implanter le travail;
- Développement d'estimations justes des ressources nécessaire pour effectuer le travail;
- L'analyse de coûts/bénéfices du changement demandé;

La sévérité d'un problème est aussi souvent utilisé pour décider comment et quand un problème sera corrigé. Le mainteneur identifie ensuite les composants affectés.
Plusieurs solutions potentielles sont fournies et alors une recommandation est faite sur la meilleure voie d'action. Les logiciels conçus avec la maintenabilité à l'esprit facilitent grandement le processus d'analyse d'impact.

L'analyse d'impact du changement est effectuée principalement lors de la maintenance des programmes, en vue d'évaluer les effets du changement sur le reste du système et de réduire le risque de s'embarquer dans des dépenses coûteuses. Cette évaluation englobe l'estimation des ressources humaines, financières, temps, effort et plannings nécessaires pour accomplir le changement.

I.7.1 Processus du changement

Une méthode formelle pour gérer le changement est essentielle pour deux raisons. Elle fournit (i) un canal de communication commun entre le personnel de maintenance, les utilisateurs, le chef de projet et les opérateurs, (ii) un annuaire des changements du système, pour la gestion de projet, l'audit et le contrôle de qualité [7].

Madhavji définit les étapes du processus de changement comme suit : [55]

- Identifier le besoin d'apporter le changement à un élément (item) de l'environnement;
- Acquérir le changement adéquat relatif à l'élément;
- Évaluer ou estimer l'impact du changement sur les autres éléments de l'environnement;
- Sélectionner ou construire une méthode pour le processus de changement;

- Faire les changements nécessaires pour tous les éléments, et résoudre les interdépendances de manière satisfaisante;
- Enregistrer les détails des changements pour de futures références;
- Remettre l'élément changé de nouveau dans l'environnement.

Pour bien mener les changements dans un environnement, il est nécessaire de connaître tous les facteurs affectant le changement en question et les conséquences de ce changement.

Bohner décrit un processus plus en détail de changement du logiciel qui incorpore l'analyse d'impact [8]. Ce modèle illustre où les impacts du changement peuvent être détectés durant la plupart des activités de changement du logiciel.

Ces étapes se présentent comme suit :

1. Gérer le changement logiciel : au cours de laquelle les objectifs de qualité sont établis, les risques des changements sont déterminés et le suivi de la progression du changement, de l'attribution des ressources et de la planification du release sont faites;

2. Comprendre le changement et déterminer l'impact : permet d'identifier les impacts du changement, de clarifier la requête du changement, d'enregistrer les impacts du changement et de déterminer la stabilité du logiciel. L'identification de l'impact représente l'activité centrale qui supporte la plupart des activités du processus de changement. Cette étape a pour objectif de déterminer les impacts du changement du logiciel, de classifier les changements et d'explorer les changements similaires. De plus, elle permet d'identifier les impacts des exigences et de conception, d'analyser les impacts du code source et de déterminer les tests de régression candidats;
3. Spécifier et concevoir les changements : au cours de laquelle se fait l'analyse des exigences du changement, l'examen des changements de l'architecture du logiciel, la dérivation des exigences du changement et la conception du programme de changement;
4. Implémenter les changements : c'est une activité itérative qui englobe la détermination des modules à changer, l'identification des expressions du programme à changer, l'application des changements au programme et le test unitaire du logiciel modifié;
5. Retester le logiciel affecté : consiste à générer les cas de tests pour les nouvelles fonctionnalités ou celles modifiées, à mettre à jour la suite de tests, à effectuer les tests d'intégration, à mener le test du système et à réaliser les tests d'acceptation. Elle a pour objectif

d'assurer que les modifications effectuées répondent aux nouveaux besoins, et tout le système satisfait les besoins existants.

D'une façon générale, une bonne méthodologie et une parfaite compréhension du système et du changement contribuent largement dans la réduction du temps entre la proposition du changement, son analyse, son implémentation et sa délivrance. Par conséquent, moins d'effort requis pour effectuer le changement. L'implémentation des modifications doit se faire de manière organisée et par priorité. D'une part, les changements qui chevauchent doivent être rassemblés de même que les changements similaires afin d'éviter la duplication du travail et l'augmentation de la complexité. D'autre part, dans le but de cerner les effets de bord du changement et son effet de propagation, un seul changement doit être traité à la fois sinon on risque de rendre le système indisponible.

I.7.2 Avantages de l'analyse d'impact

Quand on est en face d'une nécessité de modification dans un système, il est nécessaire de savoir l'impact ou les effets secondaires qui peuvent résulter de cette modification sur le reste du système. Sans une bonne analyse d'impact du changement, les ingénieurs peuvent faire de petits changements qui peuvent involontairement causer des problèmes majeurs ou avoir des répercussions sur tout le système. [56]

En génie logiciel, cerner les spécifications et les objectifs du projet avant son développement diminue le risque de refaire le travail, de retarder les plannings et de dépasser le budget. Le même principe se présente en analyse d'impact du changement. L'identification des impacts du changement, la planification du changement et l'estimation des ressources requises avant d'implémenter le changement, permettent de diminuer le risque de s'embarquer dans des dépenses coûteuses, inutiles ou imprévisibles.

Lorsqu'un changement se présente, plusieurs solutions sont envisageables pour résoudre le même problème. L'analyse d'impact du changement permet aux gestionnaires de qualifier leurs choix à base d'informations recueillies par l'équipe de maintenance sur la faisabilité technique du changement. L'analyse d'impact est donc un outil d'aide à la décision. En effet, si un changement peut affecter un ensemble de sections disjointes du programme, les gestionnaires peuvent choisir de ne pas l'implémenter ou de l'examiner une autre fois pour une implémentation alternative plus sûre. L'analyse d'impact du changement permet de diriger les tests de régression. Ces derniers jouent un rôle intégral dans la maintenance du logiciel. Les tests de régression appliqués au logiciel modifié, permettent d'avoir l'assurance que le code modifié

continue à satisfaire les spécifications actuelles et qu'il ne compromet pas le comportement du code non modifié.

Suite à une modification, il faut retester toutes les parties du code qui ont été affectées directement ou indirectement par ce changement. Pour cela, il faut recenser toutes les classes qui doivent être retestées pour que le système ne perde pas ses caractéristiques initiales et continues à satisfaire les besoins des utilisateurs actuels et futurs. Sans ce recensement, on doit retester tout le logiciel, ce qui est très coûteux et très long. Ne pas tester suffisamment peut avoir une influence sur la qualité du logiciel.

I.7.3 Difficultés de l'analyse d'impact

L'analyse d'impact est l'une des parties les plus complexes du processus de changement d'un logiciel. La plupart du temps elle est effectuée quand on commence à se soucier du budget nécessaire pour accomplir le changement. Les langages orientés objets ont été considérés comme des cas difficiles pour l'analyse d'impact [09]. Par exemple, dans une application orientée objet, l'implémentation des méthodes de classes tend à être courte et se limite à une simple instruction d'invocation des autres méthodes. Plus encore, les caractéristiques de l'approche orientée objet comme le polymorphisme et la liaison dynamique amplifient et compliquent les dépendances entre les méthodes. Les relations complexes entre les classes d'objets rendent difficiles l'anticipation et l'identification de la propagation des effets de changement [08].

Par ailleurs, l'analyse d'impact repose sur des informations statiques connues à la compilation. Le polymorphisme et la liaison dynamique impliquent que les objets peuvent prendre plus d'une forme, qui n'est connue qu'au moment de l'exécution. L'analyse statique est donc insuffisante pour identifier les composants affectés. Faire intervenir le programmeur permet d'apporter plus d'efficacité au processus d'analyse d'impact et d'obtenir des résultats bien précis sur les conséquences du changement. Dans notre recherche, on se base sur l'équipe de développement et son expérience afin d'estimer l'effort dépensé pour faire le changement. Puisque ces personnes sont fortement expérimentées, elles peuvent nous fournir des grandeurs précises.

I.8 Modèle d'impact de Changement [69]

I.8.1 Objectifs

La conception d'un système peut être décrite comme un ensemble d'artefacts logiciels (classes) qui sont en interaction. Les liens d'inter-classe sont assumées avoir une plus grande influence sur

la maintenabilité que les liens d'intra-classe**.** Ainsi, nous nous concentrons sur comment les liens divers entre des classes influencent en fait l'impact de changement.

Quand un changement à un système est considéré, il est nécessaire d'identifier les composants du système qui seront impactés suite à ce changement. Cela permet de s'assurer que le système s'exécutera toujours correctement après que le changement soit mis en œuvre. Notre intérêt est concentré sur comment le système réagit à ce changement et à d'autres changements en général. Notons qu'un système absorbe facilement un changement si le nombre de composants impactés est petit.

I.8.2 Modèle Conceptuel

Un système est vu comme un ensemble de classes connectées par différents liens. Une classe est définie comme un groupe de méthodes qui servent comme interface publique ou pour des opérations internes, et une section de variables qui définissent l'état des instances de la classe.

I.8.2.1 Changements

Nous définissons un changement à un système comme un changement qui peut s'appliquer à un composant. Un composant se réfère à une classe, une méthode, ou bien une variable. Comme exemples de changement, on peut avoir la suppression d'une variable, le changement de la portée d'une méthode, de "public" à "protected", ou le déplacement du lien entre une classe et son parent.

Le tableau. I.3 présente les principaux changements aux systèmes O.O au niveau conception. Ils sont définis puis classifiés selon le composant qu'ils affectent et un total de 13 changements est identifié.

Composant	***Description de Changement***
Variable	**Changement de type de variable**
	Changement de portée de variable
	Ajout de variable
	Suppression de variable
Méthode	**Changement de type de retour de méthode**
	Changement d'implémentation de méthode
	Changement de signature de méthode
	Changement de portée de méthode
	Ajout de méthode
	Suppression de méthode

Classe	**Changement de structure d'héritage de classe**
	Ajout de classe
	Suppression de classe

Tab. I.3 Principaux changements au niveau conceptuel

I.8.2.2 Liens

Une fois qu'un composant donné est soumis à un changement, nous sommes intéressés par savoir quelles autres parties (classes) dans le reste du système seront affectées par ce changement. Une partie spécifique peut être affectée, dans le cas où elle est liée au composant changé via quelques liens entre eux. Ces liens sont parmi les quatre types suivants:

S (association): une classe fait référence aux variables d'une autre classe,

G (agrégation): la définition d'une classe implique des objets d'une autre classe,

H (héritage) : une classe hérite les particularités définies dans une autre classe (parente),

I (invocation) : les méthodes d'une classe invoquent des méthodes définies dans une autre classe.

Nous considérons aussi une notation spéciale généralement utilisée dans l'algèbre booléenne :

L'absence d'un opérateur entre 2 liens signifie une ***intersection***.

L'opérateur "+"signifie une ***union***.

L'opérateur "~" avant un lien signifie la ***négation***, *c.à.d l'*ensemble des classes non associées par ce lien spécial, par exemple, **~G** signifie l'ensemble des classes qui ne sont pas liées à la classe indiquée par le lien d'agrégation.

Les liens sont indépendants les uns des autres et nous pouvons trouver n'importe quel nombre et type de liens entre deux classes. Un changement d'une classe peut aussi avoir un impact dans la même classe. Le pseudo lien **L** (local) est introduit pour exprimer ceci.

I.8.2.3 Impact

Nous appelons impact d'un changement l'ensemble de classes qui exigent une correction suite à ce changement. Il dépend de deux facteurs : l'un est le type de changement. Par exemple, un changement de type de variable a un impact sur toutes les classes faisant référence à cette

variable tandis que l'ajout d'une variable n'a aucun impact sur ces classes. Étant donné un type de changement, l'autre facteur est la nature des liens impliqués. Si, par exemple, la portée d'une méthode est changée de "public" à "protected", les classes qui invoquent la méthode seront impactées, à l'exception des classes dérivées. Nous notons que plus qu'un type de lien entre la classe changée et une classe impactée peuvent être impliqués dans le calcul de l'impact.

Ainsi, pour un changement donné ch_i dans la classe cl_j, l'ensemble des classes impactées est exprimé par une expression booléenne dans laquelle les variables représentent les liens. Par exemple, la formule d'impact pour un changement hypothétique peut être donnée par : Impact (cl_j, ch_i) = **S~H+G**

Le tableau I.4 suivante présente un exemple de changement pour chaque type de composant avec son expression d'impact : (Voir la suite du tableau dans L'annexe A)

Composant	***Description de changement***	***Expression d'impact*** **Impact (cl_j, ch_i)**
Variable	*Changement de type*	**S+L**
Méthode	*Changement de portée de "public" à "protected"*	**I~H**
Classe	***Suppression***	**H+G+S+I**

Tab. I.4 Exemples de Changements

Ce modèle d'impact permet de prédire quelles classes seraient impactées si un changement a été réellement fait. Un changement donné est caractérisé par une transformation du code quelque part dans le système. Si le système est recompilé avec succès, alors il n'y a aucun impact. Sinon, nous sommes face à un impact, c'est-à-dire, des modifications du code qui doivent être faites ailleurs dans le système pour obtenir un code syntaxiquement correct qui se recompilera. Les considérations sémantiques relatives à la transformation du code ne sont pas considérées dans ce cas car elles ne peuvent pas être inférées du code source seulement. Puisque l'intérêt est centré seulement sur l'impact syntaxique d'un changement, les mesures appropriées que nous devons appliquer sont basées sur l'impact qui dépend seulement de la nature statique du code source. Ainsi, l'impact qui peut survenir pendant le temps d'exécution dû au polymorphisme est approvisionné.

Notons que Le modèle défini au niveau conceptuel (Fig. I.1**)** été déjà adapté en C++ [47], et [70], le travail de [71] vise les systèmes logiciels codés en Java, une opération d'adaptation de ce modèle à ce langage a été nécessaire. A cet effet l'auteur dans [71] a examiné l'adaptation déjà faite dans, [47] et [48]. Il a remarqué qu'il y a certains changements qui sont communs aux deux langages, dans le sens où ils peuvent être appliqués aux deux langages. A titre d'exemples, le changement de type de variable, changement de signature de méthode, changement de la structure d'héritage d'une classe, etc. Par contre, il y a d'autres changements qui sont propres au langage C++, principalement les concepts de "virtual" (méthode virtuelle ou classe virtuelle) et " friendship" (classe amie).Les résultats de ce raffinements est présenté dans l'annexe A.

I.9 Travaux de recherche sur l'analyse d'impact de changement des systèmes OO

Plusieurs études ont été menées sous différentes formes à propos de l'analyse d'impact. Han a développé une approche pour calculer l'impact de changement sur les documents de conception et d'implémentation [45]. Kung et al se sont intéressés à l'impact de changement pour les tests de régression [46]. Ils ont classifié les changements et l'impact résultant en se basant sur les relations d'héritage, d'association et d'agrégation. Des algorithmes formels ont été réalisés pour calculer l'ensemble des classes impactées tout en incluant l'effet de propagation. Une méthode a été proposée pour identifier les classes affectées suite aux changements de structures d'une librairie de classe. Pour chaque structure de changement, l'impact sur les classes résiduelles est calculé pour déterminer si ces classes sont affectées ou non. Afin de calculer l'impact sur tout le système, le concept de firewall est utilisé.

Chaumun et Kabaili se sont intéressés à un des aspects de la maintenance qui est la changeabilité [47]. Leur approche consiste à calculer l'impact du changement apporté aux classes du système en utilisant le modèle d'impact défini. Ce modèle consiste à spécifier un ensemble de changements pouvant intervenir dans une classe, et pour chaque classe son impact sur les classes directement connectées à la classe changée. Par ailleurs, ce modèle d'impact a été étendu par Kabaili pour tenir compte de l'effet de propagation et des tests de régression afin d'obtenir une meilleure évaluation de la changeabilité du système [48]. Pour déterminer les classes à retester, elle a utilisé une approche inspirée du concept du firewall et basée sur les relations de dépendances entre les classes.

Li et Offut analysent l'impact des changements apportés au logiciel OO en prenant en considération l'encapsulation, l'héritage et le polymorphisme [49]. Cette analyse accorde une grande importance aux tests de régression en suggérant les classes et les méthodes qui ont besoin

d'être retestées. Ils ont recensé un ensemble de types de changements, analysé leurs caractéristiques et déterminé leurs influences sur les autres parties du système. Un ensemble d'algorithmes est proposé ayant pour objectifs l'impact à l'intérieur de la classe changée, l'impact parmi les classes clientes et l'impact parmi les classes dérivées. Combinés, ils permettent d'analyser l'effet de propagation à travers le système. Ces algorithmes calculent la fermeture transitive pour chaque classe susceptible d'être affectée par le changement d'un composant.

Lindvall, dans son approche de recherche, vise à comprendre les changements les plus communs en C++ pour conduire l'analyse d'impact le plus exactement possible [50]. L'approche est répartie en deux étapes. En premier, une étude empirique a été menée afin de détecter et d'analyser les changements effectués à un code source après tous les changements effectués et le système livré. La deuxième étude est complémentaire où les développeurs indiquent leurs perceptions sur l'occurrence de certains types de changements. Les résultats de l'étude sont confirmés par les résultats de l'enquête à quelques différences prés. En somme, ces résultats permettent de bien comprendre quels sont les modèles qui sont stables et ne changent pas, et quels sont les modèles qui peuvent changer et changent fréquemment.

Antoniol et al proposent une approche pour prédire la taille des changements des systèmes OO en évolution, basée sur l'analyse des classes impactées par la requête du changement [51]. Ils prédisent la taille du changement en termes de nombre de lignes de code (LOC) ajoutées et modifiées. L'approche a été évaluée empiriquement par l'analyse de la relation entre le nombre de LOCs ajoutées/modifiées et le nombre de classes ajoutées/modifiées sur 31 versions d'un système. L'analyse des codes sources a montré que le LOCs a augmenté considérablement de la première version à la dernière version. En outre, le nombre de LOCs supprimées est généralement faible, comparé au nombre de LOCs ajoutées et modifiées et le nombre de LOCs ajoutées est plus grand que le nombre de LOCs modifiées.

Pfleeger et Bohner considèrent l'analyse d'impact comme l'activité primaire dans la maintenance du logiciel. Ils ont utilisé l'analyse d'impact pour mesurer la stabilité de tout le logiciel avec sa documentation en proposant un nombre de métriques logicielles [52]. Le modèle est basé sur le graphe de traçabilité. Un tel graphe montre les relations dans le code source, les cas de tests, les documents de conception et les spécifications. Pour chaque workproduct (exigences, conception, code, plans de test), la traçabilité verticale exprime les relations parmi les parties du workproduct, et la traçabilité horizontale gère les relations de ses composants par

paire de workproduct. La traçabilité (verticale et horizontale) a été représentée en utilisant un graphe orienté.

Enfin, **Arnold et Bohner** définissent un modèle conceptuel composé de trois parties pour caractériser et comparer les différentes approches d'analyse d'impact, et ressortir les forces et les faiblesses de chacune d'elles [53]. La première partie du modèle d'analyse d'impact (Impact Analysis application) examine la méthode utilisée par une approche, pour accomplir l'analyse d'impact. La deuxième partie (Impact Analysis Parts) s'intéresse au fonctionnement de l'approche, à savoir, ce qu'elle fait, comment elle le fait et les outils utilisés. La dernière (Impact Analysis effectiveness) s'intéresse à l'efficacité de l'approche d'analyse d'impact.

I. 10 Analyse d'impact du changement et réseaux bayésiens

Tang et al. [74] ont utilisé des réseaux bayésiens pour l'analyse de l'impact des changements au niveau architectural. En effet, les auteurs présentent le modèle AREL (Architecture Rationale and Element Linkage) qui est utilisé pour présenter la relation entre les décisions à apporter et les éléments de conception qui dépendent de ces choix de modification d'architecture. Par la suite ce modèle fut transposé en des réseaux bayésiens où les nœuds sont les éléments du AREL et les arcs sont les relations de dépendances entre ces nœuds. Les réseaux bayésiens ont fait l'objet d'une étude de cas qui consiste en une application du domaine bancaire. Les besoins fonctionnels ont été définis et traduits en AREL et ensuite en réseaux bayésiens dont les tables de probabilités (TPs) sont définies par des spécialistes. L'aspect prédiction consiste à estimer les effets qu'un changement de besoins ou dans l'environnement pourrait avoir sur les éléments de l'architecture en identifiant les relations de causalité dans le réseau bayésien.

Abdi, al. [3] se sont intéressés à la compréhension des facteurs réels qui sont responsables de l'impact de changement et de son évolution. Des métriques de conception et d'implémentation sont étudiées afin de comprendre leurs effets sur les systèmes. Les auteurs ont proposé une approche probabiliste qui utilise les réseaux bayésiens pour déterminer l'impact des changements.

Mirarab [75] a proposé une approche qui utilise deux sources d'information: les métriques de dépendances et les données historiques. Les métriques de dépendances reflètent le degré de liaison entre les paquetages du système exprimé en nombre d'appels entre les classes de chaque paquetage. Ces dépendances peuvent correspondre à des appels méthodes ou à des utilisations de variables. Pour la détection de ces dépendances, les auteurs utilisent des techniques statiques. Le deuxième type d'information qui est les données historiques, représente des données relatives

aux co-changements dans le passé. Bien que cette information ne présente pas la propagation des changements, les auteurs font l'hypothèse que les éléments changés en même temps dans le passé ont de fortes chances de changer de la même façon dans les versions futures. Une fois ces données collectées, la deuxième étape consiste à construire des réseaux bayésiens pour prédire l'impact des changements. Les réseaux bayésiens utilisés dans cette approche sont de trois types :

- BDM (Bayesien Dependency Model)
- BHM (Bayesian History Model)
- BDHM (Bayesian Dependency and History Model).

Zhou et al. [76] présentent un modèle de réseaux bayésiens qui prédit la probabilité des co-changements entre les entités du système. Les informations servant de point d'entrée aux réseaux sont extraites à partir des données historiques et des dépendances dans le code et sont essentiellement : l'âge du changement, son auteur, la fréquence des changements, l'objectif des changements et d'autres informations qui servent à prédire les classes potentiellement affectées par un changement. Le processus de cette approche est divisé en trois étapes. La première consiste à collecter les données à partir des dépôts de contrôle de versions. Ces données sont transformées en une base de données de changements. La deuxième étape consiste à extraire les facteurs qui peuvent influencer les changements détectés tels que l'auteur qui les a effectués, l'entité du système qui a été modifiée ou la date du changement vu que les auteurs adhèrent à l'idée que les entités changées depuis longtemps deviennent plus stables. Ces facteurs forment ensuite les nœuds des réseaux bayésiens qui vont être, durant la dernière phase du processus, entraînés pour pouvoir prédire l'impact des changements.

L'étude de l'impact de changement est une activité fondamentale dans le génie logiciel car elle peut servir à planifier des changements, à les mettre en place et à prévoir ou détecter leurs effets sur le système et essayer de les réduire.

Diverses méthodes ont été présentées dans la littérature pour ce volet de la maintenance. La première partie de ce chapitre présente un aperçu des travaux relatifs à l'analyse de l'impact des changements. On a décidé de les classer en deux catégories : les travaux qui utilisent la traçabilité entre les artefacts comme les diagrammes d'UML ou de l'architecture et ceux qui se basent sur les dépendances extraites à partir du code source. Les différentes contributions relatives à cet aspect de maintenance peuvent être classées selon leurs objectifs. En effet, les

travaux présentés dans ce chapitre ont deux objectifs : la prédiction de l'impact et l'analyse de l'impact résumée dans les travaux cités dans la section précédente. Et le tableau suivant résume quelques approches de prédiction de l'impact :

Approche	Techniques	Artefact logiciel utilisé
Briand [9]	Métriques de couplage	Code source + données historiques
Tang [74]	AREL+ réseaux bayésiens	Architecture du système
Abdi [3]	Métriques de couplage+ réseaux bayésiens	Code source
Mirarab[75]	Métriques de couplage + réseaux bayésiens	Code source + données historiques
Zhou [76]	Réseaux bayésiens	Données historiques

Tab.I.5 Travaux sur la prédiction de l'impact

I.11 Conclusion

La maintenance est une phase importante dans le cycle de vie d'un logiciel. Diverse techniques comme l'analyse de l'impact des changements sont déployées lors de cette phase pour garantir le bon fonctionnement du système et assurer son évolution.

L'étude de l'impact de changement est une activité fondamentale dans le génie logiciel car elle peut servir à planifier des changements, à les mettre en place et à prévoir ou détecter leurs effets sur le système et essayer de les réduire.

Ce chapitre présente un aperçu sur les différent notions sur l'impact de changement et sa propre utilité dans pour faciliter la phase de la maintenance des logiciels, il les existe plusieurs méthodes consacrés sur l'analyse de l'impact des changements, pour notre travail on va faire la prédiction d'impacts des changements dans les systèmes à objets par les réseaux bayésiens.

Chapitre II :

Réseaux bayésiens

II.1 Introduction

Les réseaux bayésiens, qui doivent leur nom aux travaux de Thomas Bayes au dix-huitième siècle sur “la probabilité des causes”, sont le résultat de recherches effectuées dans les années 80. Les réseaux bayésiens sont un outil de modélisation à la jonction de la théorie des probabilités et de la théorie des graphes, permettant de décrire les relations régissant un ensemble de variables aléatoires et d'effectuer un raisonnement probabiliste sur celles-ci. Il constitue à la fois un formalisme de représentation des connaissances, ainsi qu'un outil permettant d'expliquer et de prédire l'état de variables d'intérêt d'un domaine de connaissances en fonction de l'état de variables observées.

De part le développement d'algorithmes d'inférences efficaces, les réseaux bayésiens ont été récemment mis en application dans de nombres domaines, tels que l'aide au diagnostic médical et industriel, la surveillance de réseaux de télécommunications, la classification automatique de documents structurés ou encore l'analyse d'images.

II.2. Pourquoi les réseaux bayésiens

Une des grandes problématiques de notre époque est de traiter la grande quantité des données qui est mise à notre disposition (notamment grâce à l’informatique) pour en extraire de l’information. Il serait donc intéressant d’avoir un (ou plusieurs) modèle(s) effectuant le lien entre les observations et la réalité pour un objectif précis, et cela, même lorsque les observations sont incomplètes et/ou imprécises [10].

II.3 Un réseau bayésien comme un système expert

Un système expert à base de règles est souvent défini comme une application capable d’effectuer des raisonnements logiques comparables à ceux que feraient des experts humains du domaine considéré. Il s’appuie sur des bases de données de faits et de connaissances, ainsi que sur un moteur d’inférence, lui permettant de réaliser des déductions logiques (chaînage avant et arrière).

En pratique, un système expert modélise le mode de raisonnement de l’expert puis essaye de reproduire ce raisonnement sur de nouvelles requêtes (mode de raisonnement causal).Or, un modèle probabiliste ne modélise pas le mode de raisonnement de l’expert mais la connaissance qualitative que l’expert a des phénomènes physiques influençant le système (mode de raisonnement fondé). Un tel modèle n’est donc pas un système expert au sens où est

habituellement utilisé ce terme, les raisonnements effectués n'étant, par ailleurs, pas logiques, mais probabilistes.

Les réseaux probabilistes sont également une représentation du savoir incertain plus flexible que les systèmes à base de règles. Par exemple, en médecine, une même combinaison de symptômes peut être observée pour différentes maladies. Il n'y a donc pas toujours de règles strictes pour obtenir un diagnostic. Pour un système complexe, un expert humain est capable de porter un jugement même lorsque toutes les données nécessaires ne sont pas observées. Un système expert ne peut pas faire cela alors qu'un modèle probabiliste le permet.

De plus, les réseaux bayésiens sont plus facilement adaptés et mis à jour en fonction du contexte que les systèmes à base de règles. L'expérience montre qu'il est alors plus simple et rapide de créer des modèles graphiques. Ceux-ci étant très intuitifs, la communication avec les experts devient plus simple.

Les réseaux bayésiens permettent de modéliser la connaissance subjective d'un expert, mais ne modélisent pas le mode de raisonnement qu'effectue un expert.

Pour résumer :

- L'aspect graphique des modèles graphiques permet de représenter les relations entre les attributs clairement et intuitivement.
- Leurs orientations (si elles existent) peuvent représenter des relations de cause à effet.
- Les modèles probabilistes sont capables de gérer l'incertain et l'imprécis.

Le rôle des graphes dans les modèles probabilistes est triple :

- Fournir un moyen simple et efficace d'exprimer des hypothèses,
- Donner une représentation économique des fonctions de probabilité jointe,
- Faciliter l'inférence à partir d'observations.

Ces modèles deviennent incontournables lorsque nous avons affaire à un problème sujet à l'incertain. Les modèles probabilistes sont alors utiles pour :

- L'extraction de connaissance probabiliste : c'est-à-dire trouvé quelles variables sont corrélées, dépendantes ou conditionnellement indépendantes,
- Le diagnostic : l'évaluation de P(causes|symptômes),
- La prédiction : l'évaluation de P(symptômes|causes),
- La classification : le calcul de $MAX_{classes}$ P(classe | observations) [10].

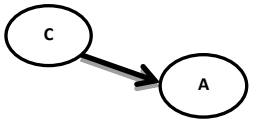
II.4 Les réseaux bayésiens

II.4.1 Définition1 [11]

Un réseau bayésien est un graphe dont les arcs sont dirigés et les nœuds représentent des variables aléatoires. Les arcs du réseau modélisent les relations causales entre ces variables, tandis que les valeurs numériques attribuées à chacun d'eux modélisent l'intensité des probabilités conditionnelles associées à ces relations causales. À tout moment, l'état global d'un réseau bayésiens est défini par l'union des états (des valeurs prises) par l'ensemble des variables qu'il contient. Dans ce cas, l'utilité d'un tel système est de pouvoir en tout temps :

⇒ calculer la probabilité de son nouvel état étant donné qu'un ensemble d'évènements a été observé (certaines variables ont été observées ou ont pris des valeurs). Il s'agit de l'inférence des probabilités postérieures des variables du réseau qui n'ont pas été observées (*Belief Updating*). Pour cela, Pearl (1988) propose l'application des formules

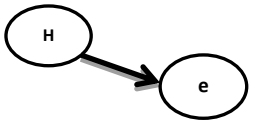
$$P(A/C) = \frac{P(A,c)}{P(C)}$$



Déduction bayesienne

Probabilité conditionnelle de A

$$P(H/e) = \frac{P(e/H).P(H)}{P(e)}$$



Abduction bayesienne:

Vraisemblance de H sachant e

Pour chaque variable *Vi* du réseau, selon que l'évidence observée corresponde à un nœud qui est une conséquence ou une cause de la valeur de *Vi*.

⇒ trouver l'état du réseau qui explique le mieux un ensemble d'observations. Il s'agit de l'état du réseau qui maximise la probabilité des évènements observés ("*BeliefRevision*"). Pour cela, Pearl (1988) utilise la formule de la règle en chaîne pour matérialiser ces algorithmes. Dans cette formule Xi est un nœud du réseau et Π_{Xi} est l'ensemble des nœuds parents de Xi. Ainsi, en tout temps, la probabilité de l'état d'un réseau bayésiens de *n* variables est le produit des probabilités postérieures de chacune de ces variables.

$$P(X_1,X_2,...,X_n) = \prod_i^n (\boldsymbol{P(Xi/\Pi\ Xi)})$$

II.4.2 Définition2

Un *réseau bayésien* est défini par :

• un graphe orienté sans circuit (*DAG*) G = (V, E), où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G;

• un *espace probabilisé* fini (Ω,Z, p) ;

• un ensemble de *variables aléatoires* associées aux nœuds du graphe et définies sur (Ω,Z, p), tel que :

$$P(V_1,V_2,.....,V_n) = \Pi_{i=1..n} [\, p(V_i \mid C(V_i)) \,]$$

Où $C(V_i)$ est l'ensemble des causes (parents) de Vi dans le graphe G [12].

II.4.3 Propriétés

Un réseau bayésien est donc un graphe causal auquel on a associé une représentation probabiliste sous-jacente. Comme on l'a vu, cette représentation permet de rendre quantitatifs les raisonnements sur les causalités que l'on peut faire à l'intérieur du graphe. Nous avons également évoqué très rapidement le lien entre *d-séparation* et *indépendance «Annexe B »* . En réalité un résultat très important existe, qui affirme que « si X et Y sont d-séparés par Z, alors X et Y sont indépendants sachant Z ». Ce résultat, démontré par Verma et Pearl [13], constitue la propriété fondamentale des réseaux bayésiens, dont nous parlerons plus précisément dans la partie suivante **:** $< X \mid Z \mid Y > \Rightarrow p(X \mid Y,Z) = p(X \mid Z)$

Ce résultat est très important, car il permet de limiter les calculs de probabilités grâce à des propriétés du graphe. Supposons que X et Y soient d-séparés par Z, et que Z soit connu.
Supposons, par ailleurs, qu'on vienne de calculer p(X | Z). Si une nouvelle information sur Y est alors connue, ce résultat nous permet de conserver notre calcul de p (X|Z) comme valeur de p(X | Z, Y) [13].

II.4.4 La représentation graphique

La représentation graphique la plus intuitive de l'influence d'un événement, d'un fait, ou d'une variable sur une autre, est probablement de représenter la *causalité* en reliant la cause à l'effet par une flèche orientée.

Fig. II.1 Représentation graphique de la causalité

Supposons que A et B soient des événements, qui peuvent être observés ou non, vrais ou faux, du point de vue du sens commun, le graphe ci-dessus peut se lire comme ceci : la connaissance que nous avons de A détermine la connaissance que nous avons de B.

Cette détermination peut être stricte, c'est-à-dire que, sachant avec certitude que A est vrai, en déduire B avec certitude. Il peut aussi s'agir d'une simple influence. Dans ce cas, cela signifie que, si nous connaissons A avec certitude, notre opinion sur B est modifiée, sans que nous puissions toute fois affirmer si B est vrai ou faux.

Avant d'aller plus loin, il est important de comprendre que, bien que la flèche soit orientée de A vers B, elle peut cependant fonctionner dans les deux sens, et ce même si la *relation causale* est stricte [12].

Par exemple, que la relation causale soit l'implication logique A ⇒B, cette relation signifie que si A est vrai, B l'est également. Si A est faux, B peut être vrai ou faux.

X	Y
V	V
F	V
F	F

Tab II.1.Table de vérité de l'implication

La table ci-dessus représente les configurations possibles de A et B dans le cas où la relation causale A => B est vraie. Cette table nous permet d'affirmer que, si B est faux, A l'est également. Du point de vue de la logique, il s'agit simplement de la contraposée de A => B. Du point de vue de la causalité, cela montre qu'une relation causale, donc orientée, est réversible de l'effet vers la cause, même si elle ne l'est que partiellement. En d'autres termes :

S'il existe une relation causale de A *vers* B*, toute information sur* A *peut modifier la connaissance que nous avons de* B*, et, réciproquement, toute information sur* B *peut modifier la connaissance que nous avons de* A.

En présence d'un graphe plus complexe, il est donc essentiel de conserver à l'esprit que l'information ne circule pas seulement dans le sens des flèches.

II.4.5 Circulation de l'information

Pour étudier de plus près comment l'information circule au sein d'un graphe causal.

On fait référence à un exemple classique dans la littérature sur les réseaux bayésiens :

« Ce matin-là, alors que le temps est clair et sec, M. Holmes sort de sa maison. Il s'aperçoit que la pelouse de son jardin est humide. Il se demande alors s'il a plu pendant la nuit, ou s'il a simplement oublié de débrancher son arroseur automatique. Il jette alors un coup d'œil à la pelouse de son voisin, M. Watson, et s'aperçoit qu'elle est également humide. Il en déduit alors qu'il a probablement plu, et il décide de partir au travail sans vérifier son arroseur automatique ».

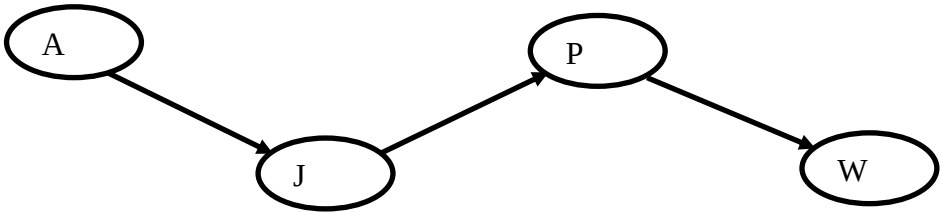


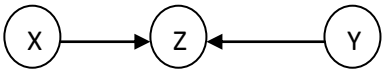
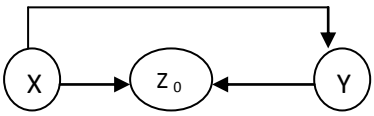
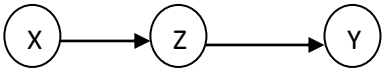
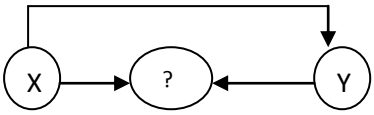
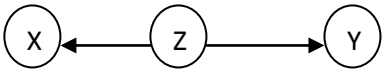
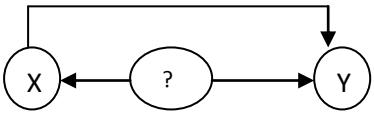
Fig II.2. Représentation graphique du modèle causal utilisé par M. Holmes

Dans ce graphe le nœud A représente l'événement que M. Holmes a oublié de débrancher son arroseur automatique, le nœud P représente l'événement qu'il a plu cette nuit, le nœud J représente l'événement que l'herbe de son jardin est humide, le nœud W représente l'événement que l'herbe du jardin de M. Watson est humide.

Comment l'information peut circuler dans ce graphe ? Évidemment l'information peut circuler depuis les causes vers les effets, d'ailleurs on remarque que *M. Holmes a décidé d'aller au bureau sans vérifier son arroseur après avoir su que l'herbe du jardin de M. Watson est humide aussi*. C'est-à-dire la connaissance de W peut modifier la connaissance de P, ou autrement dit l'information peut circuler dans la direction inverse.

II.4.6 Les chemins de circulation d'information

Le tableau suivant détermine les différents chemins possibles de circulation de l'information dans un graphe causal.

Graphe	Propriété	Exemple	
	*L'information ne peut circuler de* X *à* Y *que si* Z *est connu.* 	X = *tremblement de terre* Y = *cambriolage* Z = *alarme* Le fait qu'il y ait eu un tremblement de terre dans le voisinage (X) n'a aucun lien *a priori* avec le fait que ma maison ait été cambriolée (Y). En revanche, si mon alarme s'est déclenchée (Z), j'ai tendance à croire que je viens d'être cambriolé (Y). Si maintenant j'apprends qu'il vient d'y avoir un tremblement de terre (X) dans le voisinage, je suis rassuré sur l'éventualité d'un cambriolage (Y).	Connexion Convergente
	L'information ne peut circuler de X à Y que si Z n'est pas connu. 	X = *ensoleillement* Y = *prix du blé* Z = *récolte* Si la saison a été ensoleillée (X), la récolte sera abondante (Z). Si la récolte est abondante, le prix du blé est bas (Y). Si je sais déjà que la récolte a été abondante (Z), le fait de connaître l'ensoleillement (X) ne m'apprend plus rien sur le prix du blé (Y).	Connexion en série
	*L'information ne peut circuler de* X *à* Y *que si* Z *n'est pas connu.* 	X = *la pelouse de mon jardin est humide* Y = *la pelouse de mon voisin est humide* Z = *il a plu cette nuit* Si la pelouse de mon jardin est humide (X), j'ai tendance à croire qu'il a plu cette nuit (Z), et donc que la pelouse de mon voisin sera aussi humide (Y). Si en revanche je sais qu'il a plu cette nuit (Z), je peux affirmer que la pelouse du jardin de mon voisin sera humide (Y), et l'information que je peux avoir sur l'état de ma propre pelouse (X) n'y change rien.	Connexion divergente

Tab II.2. Circulation de l'information dans le graphe causal [12].

II.5 La probabilité conditionnelle et le théorème de Bayes

Le premier élément du langage de la probabilité est l'événement, un exemple classique d'événement est le lancement d'une pièce de monnaie en l'air. Sachant l'événement *e* on appelle *e* l'événement que *e* n'a pas lieu. La probabilité d'un événement *e* représente la fréquence que l'événement *e* a lieu. Il existe une autre représentation de la probabilité: la probabilité d'un événement *e* représente le degré de certitude d'une personne que l'événement *e* a lieu dans un seul essai. Si une personne assigne la valeur 1 à l'événement *e*, alors il est sur que *e* aie lieu. Par contre s'il assigne la valeur 0 à l'événement *e*, il croit que *e* n'a jamais lieu. Et s'il assigne la valeur entre 0 et 1 à l'événement *e*, peut-être il ne sait pas si *e* a lieu. On écrit *p(e)* pour montrer la probabilité de l'événement *e*. Alors on a la règle de somme:

$$p(e) + p(\bar{e}) = 1 \qquad (1)$$

Soient 2 événements *a* et *b* on suppose que *b* s'est produit, s'il existe un lien entre *a* et *b* cette information va modifier la probabilité de *a*. Alors on a le concept de la probabilité

$$P(a/b) = p(a, b) / p(b) \qquad (2)$$

Dans la formule (2), $p(a,b)$ représente la probabilité que tous les deux événements *a* et *b* ont lieu, $p(a|b)$ est la probabilité de *a* sachant *b*. À partir de la formule (2), on peut tirer le théorème de Bayes suivant:

$$p(a/b) = p(b/a).p(a)/p(b) \qquad (3)$$

La connaissance de *b* n'apporte rien sur celle de *a* et réciproquement.

On dit que 2 événements *a* et *b* sont indépendants si et seulement si: $p(a,b) = p(a).p(b)$

En utilisant la définition de la probabilité conditionnelle on a:

$$p(a \mid b) = p(a) \text{ et } p(b \mid a) = p(b)$$

Pour représenter les événements d'un essai, on utilise les variables aléatoires. Une variable aléatoire accepte une valeur d'un ensemble complet, mutuellement exclusif des états, chaque état est correspondant à un ou plusieurs événements. Une variable peut être discrète ou continuelle.

Sachant que la variable Y peut accepter un des états $y_1..y_n$, on a le théorème des probabilités totales:

$$p(X) = \sum_{y} \boldsymbol{p}(\mathbf{X}/\mathbf{y})\, \boldsymbol{p}(\mathbf{y}) \qquad (4)$$

À partir de la formule (3) et la formule (4), finalement on a le théorème de Bayes complet suivant:

$$P(y/x)= p(y) .p(x/y)/ \sum_y \boldsymbol{p}(\mathbf{X}/\mathbf{y})\, \boldsymbol{p}(\mathbf{y}) \qquad (5)$$

II.6 Table de probabilités conditionnelles

La paramétrisation d'un réseau bayésien nécessite à la fois un graphe et un ensemble de distributions de probabilités conditionnelles. Pour ce travail, nous allons nous limiter aux variables aléatoires discrètes. Les distributions conditionnelles seront alors représentées par des matrices telles que la somme des éléments de chaque 'ligne' soit égale à1, matrice que nous appellerons tables de probabilités conditionnelles.

Les réseaux bayésiens sont également capables de modéliser des distributions gaussiennes, des mélanges de gaussiennes, des mélanges d'exponentielles tronquées. Dans le cas des modèles conditionnels gaussiens, il est possible d'effectuer une inférence exacte tant qu'un nœud discret n'est pas enfant d'un nœud continu.

Une fois que nous possédons un modèle, l'idéal est de pouvoir en faire quelque chose. En présence d'un réseau bayésien, nous pouvons extraire un certain nombre d'informations. En premier lieu, nous avons accès à la structure du réseau, celle-ci nous permet de savoir quels attributs sont dépendants ou non. Pour cela nous utilisons la d-séparation et, par exemple, l'algorithme dit du Bayes-Ball (Shachter) [61].qui permet de vérifier si une relation d'indépendance est représentée, ou non, dans la structure. Ensuite, nous avons accès aux tables de probabilités conditionnelles, qui nous permettent de retrouver de la connaissance statistique.
Néanmoins, la majeure partie des probabilités auxquelles nous voudrions avoir accès ne sont pas inscrites dans ces tables. Passons en revue les principales méthodes existantes pour les évaluer.

II.7 Inférence

II.7.1 Définition 1

L'inférence dans un réseau bayésien concerne le calcul de la probabilité de n'importe quelle variable ou sous ensemble de variables à partir des autres variables observées. Il s'agit donc de déterminer les probabilités conditionnelles d'événements reliés par des relations d'influence [14].

II.7.3 Définition 2

L'inférence dans un réseau bayésien se résume à un calcul de probabilités *a posteriori*. Connaissant les états de certaines variables (appelées *variables d'observation*), on détermine les probabilités des états de certaines autres variables (appelées *variables cibles*) conditionnellement aux observations [23].

II.7.4 Définition 3

L'inférence est le fait de propager les informations des nouvelles évidences du réseau pour pouvoir calculer de quelle manière elles influent sur les autres variables du système et ainsi nous permettre de connaître avec moins d'a priori l'état du système observé [18].

L'inférence bayésienne est définie par Naim et al comme « le processus de propager une ou plusieurs informations certaines au sein d'un réseau pour en déduire comment sont modifiées les croyances concernant les autres nœuds ». En d'autres termes, l'inférence sert à calculer la probabilité d'une hypothèse suite à l'observation des évidences. Les évidences correspondent aux nœuds d'entrée et les hypothèses sont les différents états des nœuds de sortie du réseau. L'injection des probabilités des nœuds d'entrée va modifier récursivement les probabilités des nœuds enfants jusqu'aux nœuds de sortie. Le calcul des probabilités utilise à la fois les tables de probabilités et le théorème de Bayes [19].

II.8 Comment calculer l'inférence

II.8.1 Exemple explicatif

Prenons comme exemple l'alarme d'un magasin. Ce que l'on voudrait par exemple savoir c'est si on sait que l'alarme s'est déclenchée, quelle est la probabilité que ce soit par un camion ou par un voleur. On connait P(C), P(V) et P(A|C, V) pour toutes valeurs de C = Camion, V = Voleur

et A = Alarme et l'on voudrait connaître P(V = V |A = V) et P(C = V |A = V).

Selon le théorème de Bayes, l'on peut dire que la probabilité qu'il y ait un voleur quand l'alarme s'est déclenchée :

$$
\begin{aligned}
P(V=V|A=V) &= \frac{P(A=V|V=V)\cdot P(V=V)}{P(A=V)} \\
&= \frac{\sum_{i=V}^{F} P(A=V|C=i,V=V)\cdot P(C=i)\cdot P(V=V)}{\sum_{i=V}^{F}\sum_{j=V}^{F} P(A=V|C=i,V=j)\cdot P(C=i)\cdot P(V=j)} \\
&= \frac{98\cdot 1.2\cdot 0.3 + 97\cdot 98.8\cdot 0.3}{98\cdot 1.2\cdot 0.3 + 2\cdot 1.2\cdot 99.7 + 97\cdot 98.8\cdot 0.3 + 0\cdot 98.8\cdot 99.7} \\
&= 92.4\%
\end{aligned}
$$

De même pour la probabilité qu'il y ait un camion quand l'alarme s'est déclenché : [19]

$$
\begin{aligned}
P(V=V|C=V) &= \frac{P(C=V|V=V)\cdot P(V=V)}{P(C=V)} \\
&= \frac{\sum_{i=V}^{F} P(A=V|C=V,V=i)\cdot P(C=V)\cdot P(V=i)}{\sum_{i=V}^{F}\sum_{j=V}^{F} P(A=V|C=i,V=j)\cdot P(C=i)\cdot P(V=j)} \\
&= \frac{98\cdot 1.2\cdot 0.3 + 2\cdot 1.2\cdot 99.7}{98\cdot 1.2\cdot 0.3 + 2\cdot 1.2\cdot 99.7 + 97\cdot 98.8\cdot 0.3 + 0\cdot 98.8\cdot 99.7} \\
&= 8.7\%
\end{aligned}
$$

II.8.2 Les algorithmes d'inférence

Il existe plusieurs algorithmes d'inférence dans les réseaux bayésiens classés en deux groupes. D'un côté nous avons les méthodes d'inférence exactes qui exploitent les indépendances conditionnelles contenues dans les réseaux et donnent à chaque inférence les probabilités *a posteriori* exactes. Par exemple l'algorithme Clustering [28] effectue l'inférence en transformant le réseau en un arbre pour lequel chaque nœud regroupe plusieurs nœuds du réseau initial. De l'autre côté nous avons les méthodes approchées qui estiment les probabilités *a posteriori*. Pour ces méthodes, deux exécutions d'une inférence peuvent donner des probabilités *a posteriori* différentes [20].

Likelihoodweighting [15], Backwardsampling[16], Self importance[17] et Heuristic importance [17]) qui estiment les probabilités en effectuant plusieurs tirages dans l'ensemble des combinaisons possibles des états des variables du réseau.

II.8.2.1 Méthodes d'inférence exactes

1) Messages locaux

La première méthode d'inférence, est celle des messages locaux, plus connue sous le nom « polytree algorithm ». Elle consiste en une actualisation, à tout moment, des probabilités marginales, par transmission de messages entre variables voisines dans le graphe d'indépendance. Cette méthode ne fonctionne de manière exacte que lorsque le réseau bayésien possède une forme d'arbre (ou polytree en anglais), elle est donc à recommander dans ce cas [21].

Par ailleurs, il existe des adaptations de cette méthode ce sont les méthodes mentionnées dans l'approche exacte (2 et 5) et de la l'inférence approchée dans la 4 éme méthode qui permettent d'utiliser les messages locaux même lorsque nous ne sommes pas en présence d'une structure arborescente.

2) Ensemble de coupe

L'algorithme Loop Cutset Conditioning a été introduit très tôt par Pearl Dans cette méthode, la connectivité du réseau est changée en instantiant un certain sous ensemble de variables appelé l'ensemble de coupe (loop cutset). Dans le réseau résultant, l'inférence est effectuée en utilisant l'algorithme des messages locaux. Puis les résultats de toutes les instanciations sont combinés par leurs probabilités a priori. La complexité de cet algorithme augmente donc exponentiellement en fonction de la taille de l'ensemble de coupe [22].

3) Arbre de jonction

La méthode de l'arbre de jonction (aussi appelée clustering ou clique-tree propagation algorithm) a été introduite par Lauritzen & Spiegelhalter [23] et Jensen, Lauritzen& Olesen [24]. Elle est aussi appelée méthode JLO (pour Jensen, Lauritzen, Olessen). Elle est applicable pour toute structure de DAG contrairement à la méthode des messages locaux. Néanmoins, s'il y a peu de circuits dans le graphe, il peut être préférable d'utiliser une méthode basée sur un ensemble de coupe. Cette méthode est divisée en cinq étapes qui sont :

- Moralisation du graphe,
- Triangulation du graphe moral,
- Construction de l'arbre de jonction,
- Inférence dans l'arbre de jonction en utilisant l'algorithme des messages locaux,
- Transformation des potentiels de clique en lois conditionnelles mises à jour.

➢ **Moralisation :**

La moralisation se décompose suivant les étapes suivantes :

- Le mariage des parents : pour les nœuds possédant plusieurs parents, liaison des parents deux à deux avec des arcs supplémentaires.
- La récupération du squelette du graphe ainsi obtenu, Nous obtenons alors un graphe non dirigé dit graphe moralisé.

Définition (graphe moral)

Si G = (X, ε) est un DAG, alors G' = (X, ε') est u graphe moral associé à G si G' est un graphe non orienté tel que ε' >ε et si [ε (X_{i1} ,X_j) =1 et ε (X_{i2} , Xj) = 1] alors

ε'(X_{i1} , X_{i2}) = 1 .

La moralisation est une réécriture de$P(X_1, \cdots ,X_n)) = \pi i P(X_i, Pa(X_i)))$ par

$P(X_1, \cdots ,X_n)) = \pi i$ ***fct***$(X_i, Pa(X_i)))$

fct$(X_i, Pa(X_i)))$ telle qu'il est toujours possible de retrouver la première expression. Après avoir retiré les orientations, nous pourrions croire que de l'information a été perdue, mais cela n'est pas le cas.

➢ **Triangulation**

Pour que les potentiels de toutes les lois conditionnelles soient associés à des sous graphes complets, il suffit de procéder à la triangulation du graphe moral en y ajoutant des arêtes créant des raccourcis dans tout cycle de longueur 4 ou plus, nous obtiendrons alors un graphe moral dit triangulé.

5) Elimination de variables

L'élimination de variables est décrite dans Zhang & Poole [25]. Cet algorithme supprime les variables une par une après avoir sommé sur celles-ci. Cette méthode a été généralisée dans Dechter [26] par l'algorithme Bucket Elimination. Un ordre des variables doit être donné en entrée et sera alors l'ordre d'élimination des variables. Le nombre de calculs dépend alors de cet ordre puisqu'il influe sur la taille des facteurs futurs. Trouver le meilleur ordre équivaut au problème de trouver l'arbre de plus petite largeur dans le réseau ce qui est un problème NP-dur.

Cette méthode est avantageuse lorsqu'un ordre d'élimination des variables est déjà connu ou si le réseau est peu dense mais avec de nombreux circuits.

6) Explication la plus probable

La méthode de l'explication la plus probable (MPE pour Most probable explanation) n'est pas réellement une technique d'inférence mais plutôt un problème d'inférence.

Ce problème consiste en l'identification de l'état le plus probable. Il est possible d'adapter différentes méthodes d'apprentissage pour répondre à cette question. La technique la plus commune (et exacte, Lauritzen & Spiegelhalter [27] pour effectuer cette inférence consiste en le remplacement des signes sommes par des max et les signes produits par des min dans les formules de l'inférence classique. Il est possible d'adapter cette méthode pour trouver le deuxième cas le plus probable ou, plus généralement, le n-ième cas le plus probable.

Comme pour les autres problèmes d'inférence, il existe également des algorithmes approchés pour résoudre ce problème : par exemple, Guo, Boddhireddy& Hsu (2004)[28] propose une méthode à base de colonies de fourmis.

7) Méthodes symboliques

L'inférence probabiliste symbolique (SPI pour Symbolic Probabilistic Inference) a été introduite dans Shachter, D'Ambrosio, & DelFabero [29] et Li & D'Ambrosio [30].

Cette méthode est orientée par un but : n'effectuer que les calculs nécessaires pour répondre à la requête. Des expressions symboliques peuvent être obtenues en remettant à plus tard l'évaluation des expressions, et en les gardant sous forme symbolique.

Par ailleurs, Castillo, Gutiérrez, & Hadi [31] ont proposé une autre technique d'inférence symbolique en modifiant les méthodes existantes d'inférences numériques et en remplaçant les paramètres initiaux par des paramètres symboliques.
Ces méthodes ont le désavantage qu'il est difficile de calculer et de simplifier automatiquement des expressions symboliques mais, l'avantage qu'elles orientent les calculs. Citons deux implémentations de ces idées.

8) Méthodes différentielles

Les méthodes différentielles transforment un réseau bayésien en polynôme multivarié (Darwiche) [32]. Elles calculent ensuite les dérivées partielles de ce polynôme. Il est alors possible d'utiliser ces dérivées pour calculer les réponses à de nombreuses requêtes, et cela en

temps constant. Cette méthode est très utile lorsque nous effectuons régulièrement les mêmes requêtes car maintenant nous pourrons y répondre en temps constant.

II.8.2.2 Méthodes d'inférence approchées

Effectuer une inférence exacte est un problème NP-difficile (Cooper) [33]. Ceci n'est pas étonnant, car il est possible de simuler un problème de satisfaction de contraintes à l'aide d'un réseau bayésien (en effet, en utilisant que des probabilités 0 et 1, les nœuds d'un réseau bayesien deviennent des portes logiques).

Lorsque la dimension du réseau bayésien augmente, il est nécessaire d'utiliser de plus en plus de temps de calcul. Or, si les tables de probabilités conditionnelles ne se pas exactes (car évaluées à partir d'une base de cas peu représentative par exemple), l'intérêt d'effectuer une inférence exacte avec ces valeurs approximatives n'est plus probant. Dans ce cas, il peut être intéressant d'effectuer une inférence approchée pour économiser du temps de calcul.

Par ailleurs, pour certains réseaux bayésiens particuliers (par exemple, qui contiennent des nœuds discrets et des nœuds continus), il n'existe pas d'algorithme d'inférence exact, le seul recours pour évaluer des probabilités est alors d'utiliser des algorithmes d'inférence approchés.

1) Simulation stochastique par Chaîne de Monte-Carlo

Les méthodes MCMC (pour Markov Chains Monte Carlo) décrites dans Gilks, Richardson & Spiegelhalter [34] permettent d'échantillonner des variables aléatoires en construisant une chaîne de Markov. Les deux algorithmes à base de MCMC pour faire de l'estimation de densité les plus répandus, sont l'algorithme de Metropolis-Hastings et l'échantillonneur de Gibbs.

Ces méthodes statistiques approchées ne calculent pas exactement les lois marginales, mais en donnent une estimation. Elles permettent donc de traiter des applications de grande taille en temps raisonnable, ce qui n'est pas le cas des méthodes exactes. Les méthodes statistiques basées sur les principes de Monte-Carlo sont décrites en détail dans Robert & Castela [35].

Le principe ici est d'effectuer un certain nombre de tirages aléatoires compatibles avec une loi. Pour cela, la décomposition de la loi jointe en produit de lois conditionnelles permet de mener les calculs. En pratique, chaque variable dont tous les parents sont connus est tirée aléatoirement, jusqu'à ce que toutes les variables aient été simulées. Pour que l'estimation soit fine, il faut alors effectuer un très grand nombre de simulations.

Par ailleurs, ces méthodes sont inefficaces quand certaines probabilités sont très faibles mais elles possèdent l'avantage d'être aisément implémentables et de ne pas être fermées : plus

l'algorithme est arrêté tard, plus de cas seront simulés et plus l'évaluation des résultats sera précise.

Algorithme de Metropolis-Hastings :

L'algorithme de Metropolis-Hastings permet de simuler une loi de densité π qui n'est connue qu'à un facteur près, c'est-à-dire que l'on ne connaît que$\frac{\pi(x)}{\pi(y)}$. La transition de l'état x^{t-1}à x suivant la loi $q(x|\, x^{t-1})$est alors acceptée avec la probabilité $\alpha(x, x^{t-1})$définie par

$$\alpha(x, x^{t-1}) = \min\left(1, \frac{\pi(x)q(x|x^{t-1})}{\pi(x^{t-1})q(x^{t-1}|x)}\right)$$

Nous sommes alors en présence d'une chaîne de Markov de loi de transition, p(x, y) = q(x|y) α (x, y) avec y ≠ x. Pour simuler des exemples générés par π, il faut choisir la loi q. La vitesse de convergence dépendra principalement de ce point de départ. Remarquons que cette méthode ne génère pas d'échantillon, car la probabilité d'acceptation dépend de $x^{t-1.}$ Il suffit ensuite d'effectuer un comptage dans cette base d'exemples pour obtenir une approximation des probabilités recherchées.

2) Méthodes variationnelles

Dans l'état actuel des choses, l'inférence dans les modèles graphiques probabilistes n'est possible que lorsque les distributions de probabilités des variables continues sont gaussiennes et n'ont pas de variables filles discrètes. Lorsque ce cas se présente, il faut avoir recours à des méthodes d'inférence approchée. Une technique pour cela serait de discrétiser toutes les variables, ou encore d'utiliser les méthodes variationnelles. Néanmoins, même lorsque nous ne sommes pas dans ce cas, ces méthodes ont l'avantage d'être peu complexes, et peuvent alors se révélés utiles.

Les méthodes variationnelles ont été introduites pour approcher l'apprentissage par maximum de vraisemblance en présence d'une base d'exemples incomplète. Elles sont alors utilisées pour approcher les intégrales nécessaires pour l'inférence bayésienne ou encore l'apprentissage bayésien comme proposé par Jordan, et Wainwright & Jordan [36].
L'approche variationnelle consiste en l'application de l'inégalité de Jensen introduisant une distribution approchée *Q*.

$$\ln \mathbb{P}(X|\Theta) = \ln \sum_H \mathbb{P}(H, X|\Theta)$$
$$= \ln \sum_H Q(H|X) \frac{\mathbb{P}(H, X|\Theta)}{Q(H|X)}$$

3) Méthodes de recherche de masse

Ces méthodes supposent qu'une petite partie de l'espace de définition des variables contient une grande partie de la masse de probabilité. Elles recherchent alors les instanciations de haute probabilité et les utilisent pour obtenir une bonne approximation (Henrion [37] et Pool [38]). Cette méthode est à utiliser lorsque nous voulons négliger les évènements de probabilités faibles.

4) Loopybelief propagation

L'algorithme des messages locaux de Pearl [39] ne fonctionne de manière exacte que sur des structures en forme d'arbre. Cependant cette méthode a été généralisée dans Pearl [40] pour effectuer de l'inférence approchée avec une structure de DAG quelconque. Murphy,Weiss & J [41] proposent une étude empirique de la convergence de cet algorithme en présence de cycles, et parfois, certains résultats très mauvais peuvent être obtenus avec cette méthode.

5) Simplification du réseau

Pour effectuer l'inférence plus rapidement, il peut être envisageable de simplifier les paramètres, par exemple en mettant à zéro les probabilités trop faibles, ou encore de simplifier la structure, puis de faire de l'inférence exacte dans ce nouveau réseau.

Kjærulff [42] a été le premier à donner ce type de méthode en proposant de retirer du graphe les liens faibles au sens de la dépendance causale. De nombreuses autres techniques ont ensuite été proposées : retirant des arcs, retirant des nœuds ou encore recherchant l'arbre le plus proche du réseau, il est possible de trouver de nombreuses références dans Guo & Hsu [43].

Les méthodes d'inférence par simplification du réseau sont principalement à construire et à utiliser dans des buts particuliers, par exemple dans le cas de l'utilisation d'une simplification pertinente pour le problème traité.

II.8.3 Les propriétés d'un calcul d'inférence

Les propriétés pour lesquelles ont été établis les profils de performance sont le nombre d'échantillons, le nombre de nœuds du réseau, le nombre de variables d'observations, le nombre de variables cibles et l'espace d'évaluation. Le nombre d'échantillons définit le nombre de tirages que les algorithmes d'échantillonnage stochastique devront effectuer pour pouvoir estimer les probabilités *a posteriori*. *L'espace d'évaluation est défini* comme étant l'ensemble des variables du réseau dont la valeur doit être évaluée pour mettre à jour les probabilités sur les variables cibles en fonction des variables d'observation.

D'une façon générale, l'espace d'évaluation n'est pas simple à déterminer. L'*Ancêtre C, O* l'ensemble des nœuds contenant les nœuds d'observation, les nœuds cibles, les ancêtres des nœuds d'observation et les ancêtres des nœuds cibles.

Les auteurs de la bibliothèque SMILE décrivent dans [44] ,les caractéristiques des variables qu'ils excluent du processus de calcul des probabilités *a posteriori*. Ces variables sont (1) les nœuds n'appartenant pas à l'ensemble *Ancêtre C, O*, (2) les nœuds d-séparés des nœuds cibles par les nœuds d'observation, (3) les nœuds n'appartenant à aucun chemin entre un nœud cible et un nœud d'observation. L'espace d'évaluation utilisé dans les algorithmes de la bibliothèque SMILE est donc inclus dans l'ensemble *AncêtreC, O*. Pour simplifier, les auteurs de la bibliothèque SMILE ont choisi de sélectionner les variables d'observations et les variables cibles de telle sorte que l'espace d'évaluation soit égal à l'ensemble *Ancêtre C, O*. Pour cela, ils ont choisi des variables cibles sans parent et appartenant aux ancêtres des variables d'observation, ou inversement, des variables d'observation sans parent et appartenant aux ancêtres des variables cibles. Pour chacune de ces cinq propriétés, nous avons construit un ensemble de calculs d'inférence où seule la propriété concernée varie, alors que les autres propriétés gardent une valeur constante.

II.8.4 Apprentissage des réseaux bayésiens

L'apprentissage d'un réseau bayésien consiste à définir la structure graphique et associer des tables de probabilités conditionnelles à chaque variable du problème à modéliser. Il s'agit donc d'apprentissage de structures et de paramètres.

L'apprentissage d'un réseau bayésien doit donc répondre aux deux questions suivantes :

- Comment estimer les lois de probabilités conditionnelles ?

- Comment trouver la structure du réseau bayésien ?

Donc on va séparer le problème de l'apprentissage en deux parties :

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

II.8.4.1 Apprentissage des paramètres

1) Apprentissage à partir de données complètes

L'estimation de distributions de probabilités, paramétriques ou non, est un sujet très vaste et complexe. On décrira ici les méthodes les plus utilisées dans le cadre des réseaux bayésiens, selon que les données à notre disposition sont complètes ou non.

Dans le cas où toutes les variables sont observées, la méthode la plus simple et la plus utilisée est l'estimation statistique qui consiste à estimer la probabilité d'un événement par la fréquence d'apparition de l'événement dans la base de données. Cette approche, appelée maximum de vraisemblance (MV), nous donne alors :

$$P(X_i = x_k \mid parent(X_i) = c_j) = \theta_{i,j,k} = \frac{N_{i,j,k}}{\sum_k N_{i,j,k}}$$

Dans la formule précédente $N_{i,j,k}$ est le nombre d'événements dans la base de données pour lesquels la variable X_i est dans l'état x_k et ses parents sont dans la configuration c_j. [12]

Notons que :

Il faut déterminer chaque $P(X_i = x_k|pa(X_i) = x_j) = \Theta_{i,j,k}$

Avec $\Theta = \{ \Theta_{i,j,k}\}_{i=1:n, j=1:qi, k=1:ri}$

2) Apprentissage à partir de données incomplètes

Dans les applications pratiques, les données sont très souvent incomplètes. Certaines variables ne sont observées que partiellement ou même jamais. La méthode d'estimation de paramètres avec des données incomplètes la plus couramment utilisée est fondée sur l'algorithme itératif EM (Expectation- Maximisation) proposé par Dempster [73].

$Xv=\{Xv^{(l)}\}\ l = 1..N$: l'ensemble des données observées (visibles).

$\Theta^{x(t)} = \{\Theta^{(t)}i, j, k\}$: Les paramètres du réseau bayésien à l'itération *t*.

L'algorithme EM :

Initialiser $\Theta^{(0)}$, t=0 initialiser les paramètres $\Theta^{(0)}$

Répéter

t=t+1

calculer N*i,j,k

calculer $\Theta^{(t)}$- i ,j,k

Tant que $|\Theta^{(t)} - \Theta^{(t-1)}| \geq \varepsilon$

Fin

L'abréviation l'algorithme EM :

E : Estimer les valeurs manquantes à partir des paramètres actuels $\Theta^{(t)}$

- Calculer $P(X_{manquant}|X_{mesurés})$ dans le RB actuel
- Faire des inférences

M : ré-estimer les paramètres $\Theta^{(t+1)}$ à partir des données complétées, en utilisant MV : Max de Vraisemblance, MAP : Max A Posteriori, ou EAP : Espérance A Posteriori de $\Theta_{i,j,k}$.

II.8.4.2 Apprentissage de la structure

On supposant que la structure de ce réseau est déjà connue. Se pose maintenant le problème de l'apprentissage de cette structure : comment trouver la structure qui représentera le mieux notre problème. Une première approche consiste à rechercher les différentes relations causales qui existent entre les variables. Les autres approches essaient de quantifier l'adéquation d'un réseau bayésien au problème à résoudre, c'est-à-dire d'associer un score à chaque réseau bayésien. Puis elles recherchent la structure qui donnera le meilleur score dans l'espace des graphes acycliques dirigés. Une approche exhaustive est impossible en pratique en raison de la taille de l'espace de recherche. On a prouvé que le nombre de structures possibles à partir de n nœuds est super exponentiel. Pour résoudre ce problème, ont été proposées un certain nombre d'heuristiques de recherche dans l'espace des graphes, qui restreignent cet espace à l'espace des arbres, ordonnent les nœuds pour limiter la recherche des parents possibles pour chaque variable, ou effectuent une recherche gloutonne dans l'espace.

II.8.5 La construction d'un réseau bayésien

La construction d'un réseau bayésien s'effectue en trois étapes essentielles, qui sont présentées sur la figure suivante : [12].

Chacune des trois étapes peut impliquer un recueil d'expertise, au moyen de questionnaires écrits, d'entretiens individuels ou encore de séances de *brainstorming*[1]. Préconiser, dans un cadre général, l'une ou l'autre de ces approches serait pour le moins hasardeux.

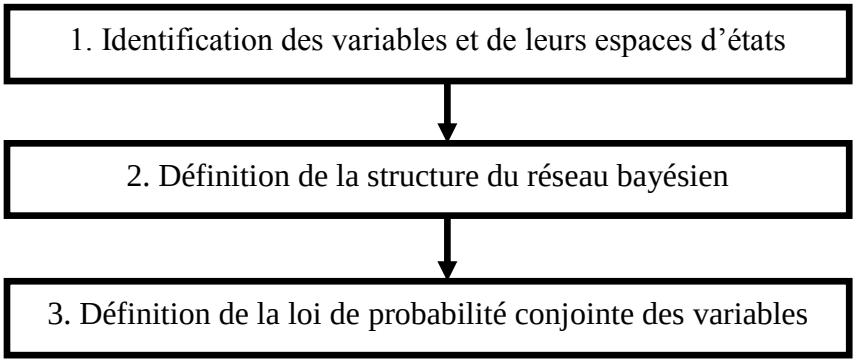


Fig. II.3 Étapes de construction d'un réseau bayésien [12].

II.8.5 1. Identification des variables et de leurs espaces d'états

La première étape de construction du réseau bayésien est la seule pour laquelle l'intervention humaine est absolument indispensable. Il s'agit de déterminer l'ensemble des variables X_i, catégorielles ou numériques, qui caractérisent le système. Comme dans tout travail de modélisation, un compromis entre la précision de la représentation et la maniabilité du modèle doit être trouvé, au moyen d'une discussion entre les experts et le modélisateur.

Lorsque les variables sont identifiées, il est ensuite nécessaire de préciser l'espace d'états de chaque variable X_i, c'est-à-dire l'ensemble de ses valeurs possibles.

II.8.5 2. Définition de la structure du réseau bayésien

La deuxième étape consiste à identifier les liens entre variables, c'est à- dire à répondre à la question : pour quels couples (i, j) la variable X_i influence-t-elle la variable X_j ?

Dans la plupart des applications, cette étape s'effectue par l'interrogation d'experts. Dans ce cas, des itérations sont souvent nécessaires pour aboutir à une description consensuelle des interactions entre les variables X_i.

L'expérience montre cependant que la représentation graphique du réseau bayésien est dans cette étape un support de dialogue extrêmement précieux. Un réseau bayésien ne doit pas comporter de circuit orienté ou boucle.

[1] : une méthode de réunion de groupe soigneusement préparée , puis tout aussi soigneusement exploitée pour trouver un nombre important d'idées *publicitaires* et *promotionnelles* pour les clients et les clients potentiels de l'agence

II.8.5 3 Définition de la loi de probabilité conjointe des variables

La dernière étape de construction du réseau bayésien consiste à renseigner les tables de probabilités associées aux différentes variables.

Dans un premier temps, la connaissance des experts concernant les lois de probabilité des variables est intégrée au modèle.

Concrètement, deux cas se présentent selon la position d'une variable Xi dans le réseau bayésien :

- La variable X_i n'a pas de variable parente : les experts doivent préciser la loi de probabilité marginale de X_i.
- La variable X_i possède des variables parentes : les experts doivent exprimer la dépendance de X_i en fonction des variables parentes, soit au moyen de probabilités conditionnelles, soit par une équation déterministe (que le logiciel convertira ensuite en probabilités).

II.9 Domaines d'application des réseaux bayésien

II.9.1 Santé

Les premières applications des réseaux bayésiens ont été développées dans le domaine du diagnostic médical. Les réseaux bayésiens sont particulièrement adaptés à ce domaine parce qu'ils offrent la possibilité d'intégrer des sources de connaissances hétérogènes (expertise humaine et données statistiques), et surtout parce que leur capacité à traiter des requêtes complexes (explication la plus probable, action la plus appropriée) peuvent constituer une aide véritable et interactive pour le praticien.

Le système Pathfinder [12], développé au début des années 1990 a été conçu pour fournir une assistance au diagnostic histopathologique, c'est-à-dire basé sur l'analyse des biopsies. Il est aujourd'hui intégré au produit Intellipath, qui couvre un domaine d'une trentaine de types de pathologies. Ce produit est commercialisé par l'éditeur américain Chapman et Hall, et a été approuvé par l'American Medical Association.

Dans le domaine de la santé, une application intéressante des algorithmes issus des réseaux bayésiens a permis d'améliorer considérablement la recherche de la localisation de certains gènes, dans le cadre du projet Human Genome.

II.9.2 Industrie

Dans le domaine industriel, les réseaux bayésiens présentent également certains avantages par rapport aux autres techniques d'intelligence artificielle. Leur capacité réelle d'apprentissage incrémental. En effet, la propriété essentielle d'un système autonome, pour pouvoir « survivre », est de s'adapter aux modifications structurelles de son environnement.

La capacité du système à gérer ses propres altérations, en particulier la perte de certaines fonctions, est également importante. Ainsi, dans la situation où certains de ses capteurs ou effecteurs sont endommagés, le système doit être capable de mettre à jour son domaine de viabilité, c'est-à-dire de réévaluer les capacités d'action qu'il lui reste, malgré le dommage qu'il a subi.

C'est cette idée qui a été mise en œuvre par la société danoise Hugin[2], considérée comme l'un des pionniers dans le développement des réseaux bayésiens. Hugin a développé pour le compte de Lockheed Martin le système de contrôle d'un véhicule sous-marin autonome. Ce système évalue en permanence les capacités du véhicule à réagir à certains types d'événements.
De cette façon, en fonction des capacités qui sont cruciales pour le reste de la mission, le système peut prendre des décisions qui vont de la simple collecte d'informations complémentaires, à la modification de la mission, ou jusqu'à l'abandon de celle-ci.
Transposant cette idée de contrôle de systèmes autonomes du monde réel à l'univers virtuel des systèmes et réseaux informatiques, les réseaux bayésiens devraient également équiper les agents intelligents. Comme nous l'avons déjà mentionné, le diagnostic est un des autres domaines de prédilection des réseaux bayésiens dans l'industrie, en particulier grâce à l'utilisation des requêtes avancées sur les réseaux. Ce domaine est aujourd'hui l'un des plus développés en termes d'applications opérationnelles des réseaux bayésiens (Hewlett-Packard, General Electric, Ricoh, etc.)

II.9.3 Défense

Comme pour beaucoup de techniques issues de l'intelligence artificielle, c'est grâce à la défense américaine que les réseaux bayésiens ont pu connaître leurs premiers développements.

La fusion de données est en particulier un domaine d'application privilégié des réseaux bayésiens, grâce à leur capacité à prendre en compte des données incomplètes ou incertaines, et à guider la recherche ou la vérification de ces informations.
La fusion de données peut se définir comme le processus qui consiste à inférer une information à laquelle on n'a pas directement accès, mais qui est relayée par une ou plusieurs sources

[2]: http://www.hugin.com/

imparfaites. Finalement, un détective privé qui affine ses conclusions à mesure que les indices se complètent est un spécialiste de la fusion de données.

Il est clair que cette approche est essentielle dans le domaine du renseignement, tactique ou stratégique. Par exemple, l'identification d'un navire ennemi est impossible directement. On va combiner des informations issues de systèmes de mesure, éventuellement brouillées, avec d'autres types de renseignements, également incertains. Les informations disponibles se complètent au fur et à mesure des efforts accomplis pour identifier ce navire, permettant de renforcer ou, au contraire, de réviser les conclusions effectuées.

II.9.4 Banque/finance

Les applications dans le domaine de la banque et de la finance sont encore rares, ou du moins ne sont pas publiées. Mais cette technologie présente un potentiel très important pour un certain nombre d'applications relevant de ce domaine, comme l'analyse financière, le scoring, l'évaluation du risque ou la détection de fraudes.

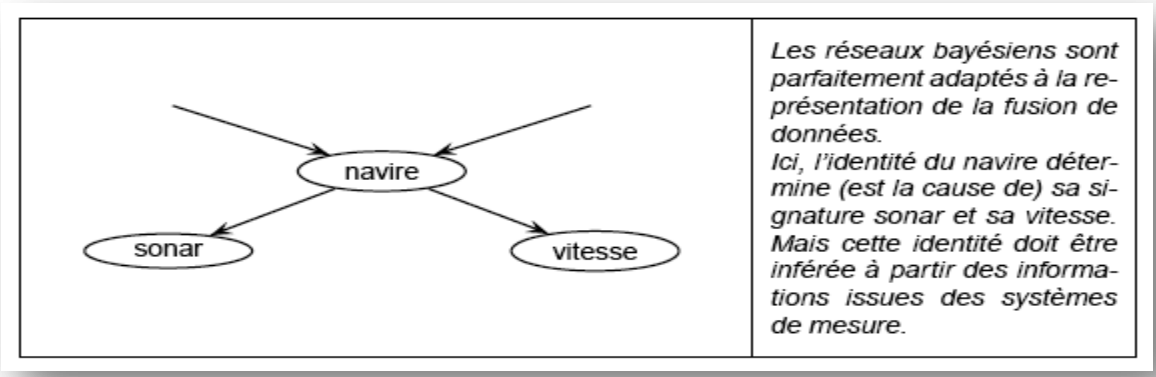


Fig. II.4. Principe de la fusion de données par réseau bayésien [12].

En premier lieu, les réseaux bayésiens offrent un formalisme unifié pour la manipulation de l'incertitude, autrement dit du risque, dont la prise en compte est essentielle dès qu'il s'agit de décision financière.

Ensuite, la possibilité de coupler expertise et apprentissage est ici très importante, non seulement parce que les deux sources de connaissances sont en général disponibles dans ce domaine, mais aussi et surtout parce que cette capacité peut aider à répondre au problème des changements structurels d'environnement.

Traitées dans les années 1980 avec des systèmes experts, des applications comme l'analyse financière, le scoring ou la détection de fraudes ont été progressivement considérées comme

relevant du domaine de la modélisation quantitative, et donc abordées par des techniques comme les réseaux neuronaux ou les arbres de décision, techniques quantitatives qui se révèlent incapables de prendre en compte par elles-mêmes la révision des modèles.
L'exemple de l'autorisation des transactions sur cartes bancaires est assez significatif. L'un des premiers systèmes experts développés dans ce domaine fut l'Authorizer Assistant d'American Express, au début des années 80 [12].

II.9.5 Marketing

Ce que l'on appelle aujourd'hui le data mining, est probablement le domaine où le potentiel des réseaux bayésiens est le plus élevé. Le data mining est défini par certains comme l'extraction automatique à partir de bases de données d'informations a priori inconnues et à valeur prédictive.

Quelle que soit la définition retenue, il reste que le développement actuel du data mining s'explique essentiellement par les applications dans le domaine du marketing, et que les réseaux bayésiens sont parfaitement adaptés à ces applications.
Le marketing est en train d'évoluer vers une gestion de plus en plus fine et individualisée du capital client, considéré comme un nouvel actif de l'entreprise. Les applications de prévision, de fidélisation, d'analyse du risque, d'anticipation des besoins, de ciblage d'actions s'inscrivent toutes dans cette démarche.
Toutes les caractéristiques des réseaux bayésiens sont autant d'atouts pour ces types d'applications [12]:

- La gestion de l'incertitude, car évidemment toutes les actions marketing sont prises dans un contexte d'incertitude, où l'on recherche avant tout à augmenter la probabilité de succès.
- La capacité à intégrer des données incomplètes au cours de l'apprentissage, car les données utilisées dans le data mining appliqué au marketing proviennent souvent de sources déclaratives, de qualité approximative.
- L'apprentissage incrémental, car les relations évoluent avec le temps.
- La gestion de requêtes complexes, comme l'analyse de sensibilités, la recherche de l'action la plus appropriée.
- Le datamining, et en particulier ses applications dans le domaine du marketing est l'un des moteurs principaux du développement des réseaux bayésiens .

II.9.6 Informatique

L'utilisation de réseaux bayésiens dans les agents bureautiques a été largement développée par Microsoft dans les outils d'aide et de diagnostic pour son système d'exploitation Windows, à partir de Windows 98. De même, l'agent Office Assistant est un système d'aide proactif intégré dans Office, à partir de la version 97. Plusieurs agents de support technique de Microsoft ont également été développés dans le cadre du projet LUMIERE du groupe DTAS (Decision Theory and Adaptive Systems).

L'application Vista, peut également être considérée comme un agent intelligent, dont le rôle est de sélectionner les données présentées à un utilisateur en fonction de l'état du système physique qu'il doit superviser.

Les réseaux bayésiens constituent le modèle idéal pour embarquer de l'intelligence ou de la connaissance. Embarquer de l'intelligence revient à doter un agent d'un équipement lui permettant de décider dans des environnements incertains, et de s'adapter lorsque ces environnements changent. Un module bayésien de prise de décision, éventuellement capable d'adaptation, est l'un des meilleurs équipements que l'on puisse fournir à un agent envoyé en mission sur Internet, ou sur d'autres types de réseau, où l'information est par nature incertaine et évolutive, voire manipulée.

II.10 Les outils bayésiens

Les réseaux bayésiens associent étroitement une structure de graphe (noeud et arc) et une information probabiliste (table de probabilités) en attribuant à chaque noeud du graphe une variable aléatoire. Un réseau bayésien peut être appris à partir de base de données et/ou modélisé par un expert. Il est ensuite possible de mettre à jour les probabilités d'occurrence de chaque état des variables en foncton d'informations sur l'état d'autres variables.

Il ya des outils de manipulation graphique de réseaux bayésiens. Ils ont pour but de permettre la saisie, la modification, l'utilisation et l'apprentissage de modèles à base de réseaux bayésiens.

II.10.1 BayesiaLab

BayesiaLab est un laboratoire complet de manipulation de réseaux bayésiens qui permet d'élaborer des modèles décisionnels par recueil d'expertise et automatiquement à partir des données, d'assimiler rapidement des connaissances représentées grâce à une boite à outils d'analyse originale, d'exploiter des modèles en mode interactif ou par lots et faire l'apprentissage des politiques d'actions. On peut consulter aussi le site http://www.bayesia.com/ pour avoir plus d'information [12].

Dans le domaine de data mining, les méthodes d'Intelligence Artificielle peuvent résoudre plusieurs types de problèmes comme la classification pour prédire les valeurs catégorielles des variables, la régression pour prédire les valeurs numériques des variables, etc. En, général on les divise en deux groupes : méthodes explicatives et méthodes non-explicatives.

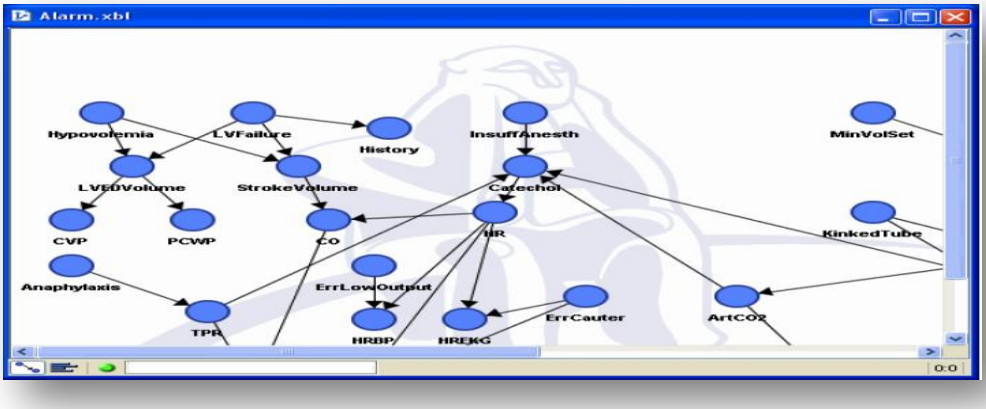


Fig. II.5. Visualisation des graphes dans bayesiaLab[3]

Le changement des paramètres va influencer la taille de la mémoire utilisée, le temps d'exécution des algorithmes et le résultat obtenu. Pour avoir de bons résultats avec des ressources matérielles limitées, il est nécessaire de choisir les valeurs appropriées des paramètres.

III.10.1.1 Domaines d'application (BayesiaLab)

- Modélisation de systèmes complexes (processus industriels)
- Analyse globale de risque et politique de sécurité (réseau de transport ferroviaire)
- Marketing (élaboration d'un profil client face à un produit ciblé)
- Risk manager
- Data Mining des bases clients (marketing et gestion des fraudes)
- Détection des intrusions

II.10.1.2 Inférence

Le logiciel gère deux types d'inférence : exacte (basée sur l'algorithme de l'arbre de jonction) et une inférence approchée lorsque les réseaux sont de complexité trop grande. L'approximation peut se faire soit par échantillonnage stochastique (Likelihood Weighting), soit par inférence exacte sur un graphe simplifié (suppression des relations les plus faibles et causant la plus grande complexité). Pour les réseaux de grande taille, un mode d'inférence exacte basé sur les requêtes

3 : http://www.bayesia.com

est également disponible (relevance reasoning). Ce mode permet, par l'analyse des observations et des nœuds requêtés, de construire l'arbre de jonction minimal.

L'exploitation nécessite la possibilité d'insérer des observations dans le réseau. BayesiaLab permet d'insérer des évidences certaines positives ou négatives (ce nœud a cette valeur ou n'a pas cette valeur), des vraisemblances (une valeur entre 0 et 100 sur chaque modalité), et des distributions de probabilités.

La console est le lieu des messages non préemptifs de BayesiaLab à l'utilisateur. Elle permet de visualiser (par exemple) des valeurs précises lors d'apprentissage, d'inférence, etc.

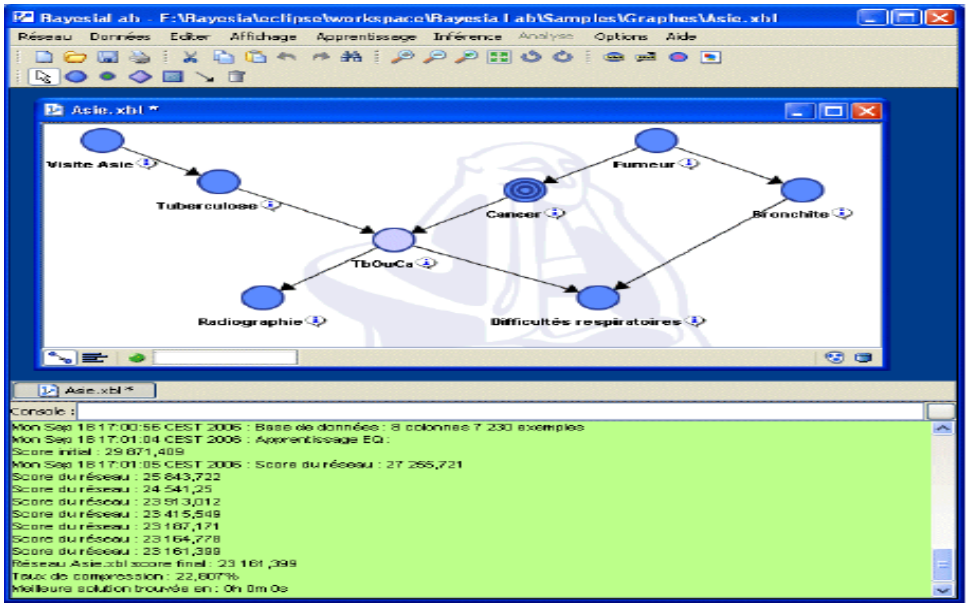


Fig.II.6 Affichage des résultats dans la console de BayesiaLab

BayesiaLab exploite le réseau bayésien en interactif (à partir d'observations entrées manuellement des «moniteurs » ou automatiquement par un fichier d'observations) ou en «batch»

II.10.1.3 Apprentissage

L'apprentissage est un des points forts de BayesiaLab. Il utilise des méthodes et des algorithmes qui sont à la pointe de la recherche dans le domaine (les fondateurs de Bayesia étant des chercheurs spécialisés dans l'apprentissage et particulièrement dans l'apprentissage de réseaux bayésiens).

L'apprentissage dans BayesiaLab prend comme entrée un fichier texte ou un lien ODBC[4] décrivant l'ensemble des cas (un cas par ligne ou un cas par colonne). Ce fichier peut intégrer un ensemble de caractères indiquant les valeurs manquantes.

Les assistants d'importation permettent la configuration de la lecture (séparateurs, ligne de titre, valeurs manquantes, transposition), l'échantillonnage, la sélection des colonnes à importer, le typage de ces colonnes (variable discrète ou continue, variable de pondération des individus, individu d'apprentissage ou de test), la scission de la base en ensembles d'apprentissage et de test.

En tant que laboratoire d'étude de réseaux bayésiens, BayesiaLab offre un très large choix dans les algorithmes à utiliser pour exploiter ces données. Il propose :

- La prise en compte de la connaissance experte exprimée sous la forme d'un graphe initial et d'un nombre de cas équivalents, des indices temporels des variables (pas d'ajout d'arc entre le futur vers le passé), des contraintes définies sur les nœuds et les classes.
- Une gestion rigoureuse des valeurs manquantes.
- Une fonction de stratification, ainsi que la prise en compte d'une variable de pondération (coefficient de redressement).
- Une complexité structurelle modifiable (jouant le rôle de seuil de significativité).
- Un apprentissage des paramètres (tables de probabilités).
- La découverte d'associations pour mettre en évidence l'ensemble des relations probabilistes directes présentes dans les données.
- La recherche commence généralement par un graphe non connecté, mais il est également possible de commencer à partir d'une structure initiale (fournie par un expert ou résultant d'un précédent apprentissage). Sauf s'ils sont fixés par l'expert, les arcs pourront alors être remis en cause lors de l'apprentissage.

➢ **Cinq algorithmes sont proposés (BayesiaLab)**:

- arbre de recouvrement maximal,
- deux algorithmes de recherche dans les classes d'équivalence,
- une recherche Taboo dans l'espace des RB et
- une recherche Taboo dans l'espace des ordres de nœuds.
- caractérisation probabiliste d'un nœud cible (apprentissage entièrement focalisé sur ce nœud cible).

[4]: *Open Database Connectivity*, permet à une application informatique, de manipuler plusieurs bases de données

II.10.2 Hugin

Hugin est un outil de construction de réseaux bayésiens, probablement le plus connu et le plus utilisé commercialement[2.]

Cet outil présente les fonctions principales suivantes :

• Construction de bases de connaissance fondées sur des réseaux bayésiens ou des diagrammes d'influence ;

• Développement de réseaux bayésiens orientés objets ;

• Apprentissage de structure et de paramètres.

Il est fourni sous forme d'un environnement graphique (Hugin Explorer), et d'un environnement de développement (Hugin Developer) permettant de piloter l'ensemble des fonctions de définition, d'inférence et d'apprentissage à partir d'une application Java, C ou Visual Basic. La société danoise Hugin Expert A/S, qui édite ce logiciel, a été créée en 1989 et est située à Aalborg au Danemark. La société a été créée après un projet ESPRIT, qui avait pour but de développer des systèmes experts de diagnostic dans le domaine médical.

Hugin s'est ensuite développée progressivement, toujours en relation étroite avec l'université d'Aalborg. Hewlett Packard a investi dans Hugin [3] en 1998, en prenant 45 % des parts de la société.

II.10.2.1 Construction des modèles

La création de réseaux bayésiens dans Hugin Explorer s'effectue avec un environnement graphique simple et assez intuitif. Cette interface permet de gérer plusieurs types de nœuds :

« nœud discret, nœud continu, nœud d'utilité, nœud de décision »

La création de modèles présente cependant certaines contraintes :

- Hugin ne permet de gérer que des nœuds continus gaussiens[5].
- Un nœud continu ne peut pas être parent d'un nœud discret.
- On ne peut pas utiliser dans le même modèle des nœuds continus et des nœuds d'utilité ou de décision [12].

- **Réseaux bayésiens à variables discrètes**

La construction d'un réseau bayésien standard à variables discrètes s'effectue de façon très simple en définissant graphiquement l'architecture du réseau et les tables de probabilités.

5 : Variable aléatoire gaussienne *X si sa densité* =

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

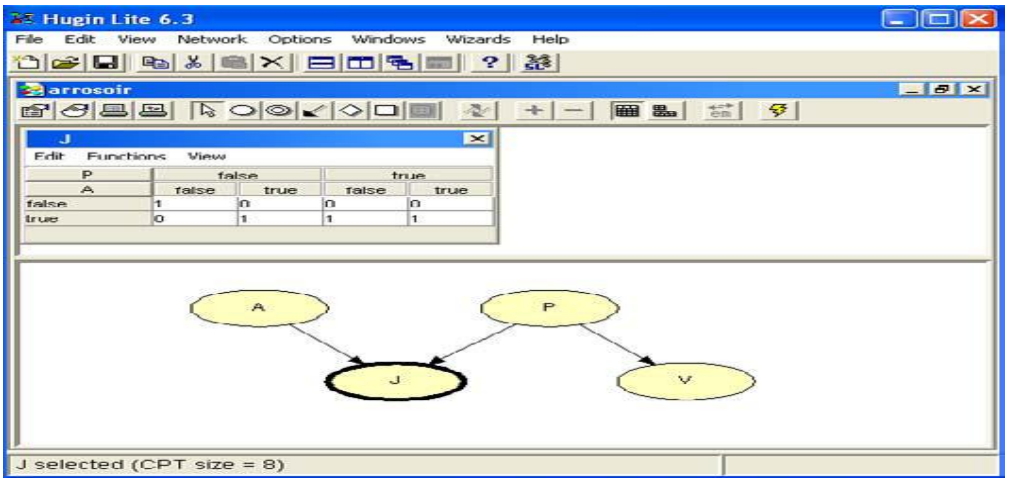


Fig. II.7 Création de modèles avec Hugin

La saisie des tables de probabilités peut être fastidieuse, notamment pour un nœud avec beaucoup de parents. Dans ce cas, et si cela est possible, Hugin permet de définir ce nœud comme une expression, arithmétique ou logique, de l'état de ses parents.

- **Réseaux bayésiens continus**

Hugin permet d'utiliser des nœuds continus dans un réseau bayésien. Lorsqu'un nœud discret est parent d'un nœud continu, la variance et la moyenne de ce dernier doivent être définies selon les états du nœud continu. Lorsqu'un nœud continu est parent d'un autre nœud continu, la distribution de ce dernier est égale à la somme de deux lois normales, l'une définie a priori, et l'autre égale à la distribution du nœud parent [12].

- **Diagrammes d'influence**

Un diagramme d'influence est, par définition, un réseau bayésien auquel on a ajouté des nœuds de décision et d'utilité. L'exemple ci dessus décrit la modélisation d'une prise de décision dans le domaine du forage pétrolier.

Un ingénieur doit choisir ou non de creuser à un certain point. Il ne connaît pas la quantité de pétrole éventuellement présente. Le puits peut être sec, humide, ou imbibé de pétrole.

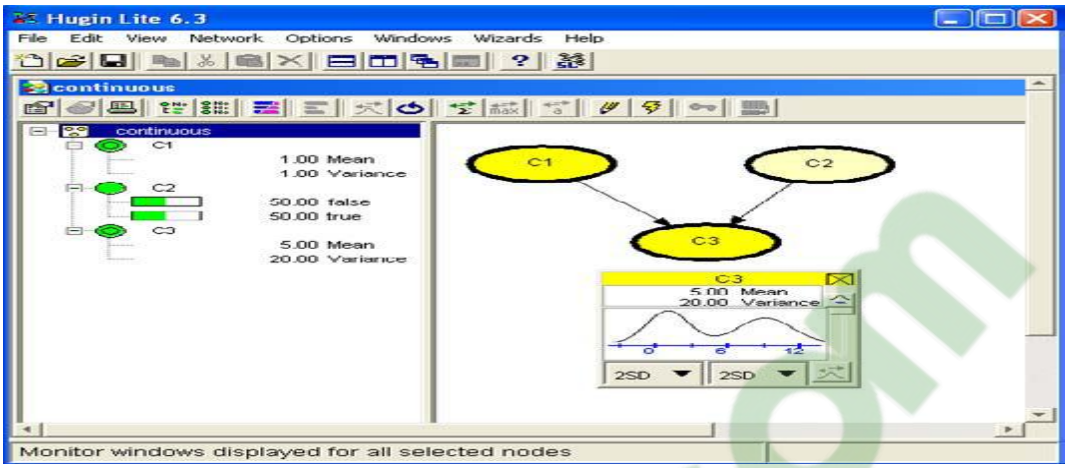


Fig.II.8 Modèles continus avec Hugin

II.10.2.2 Inférence

L'inférence dans Hugin s'effectue grâce au calcul d'un arbre de jonction sur le réseau.

Le mode le plus simple d'inférence consiste à entrer des observations dans le réseau, simplement en cliquant sur la valeur observée.

Les copies d'écran de la figure « Fig. III.6 » montrent l'utilisation de l'inférence pour l'exemple de l'arrosage du jardin. Dans l'écran de gauche, aucune observation n'a été effectuée.

Dans l'écran de droite, l'observation « l'herbe du jardin est mouillée » a été effectuée, et les probabilités des autres nœuds sont révisées.

Hugin permet également de saisir des observations partielles, grâce à la fonction de saisie de vraisemblance.

Dans l'exemple du forage ci-dessus, on peut disposer de l'information selon laquelle le puits n'est pas sec : il est donc nécessairement humide ou imbibé. Cette information peut être entrée dans Hugin en indiquant que la vraisemblance de l'observation « Le puit est sec » est nulle.

On remarque alors que, sauf information complémentaire, les probabilités des deux autres événements restent dans le même rapport qu'initialement. L'utilité de réaliser le test sismique devient alors inférieure à celle de ne pas le faire : en effet, le puits étant certainement humide ou imbibé, le forage aura toujours un résultat bénéficiaire, et le test devient inutile.

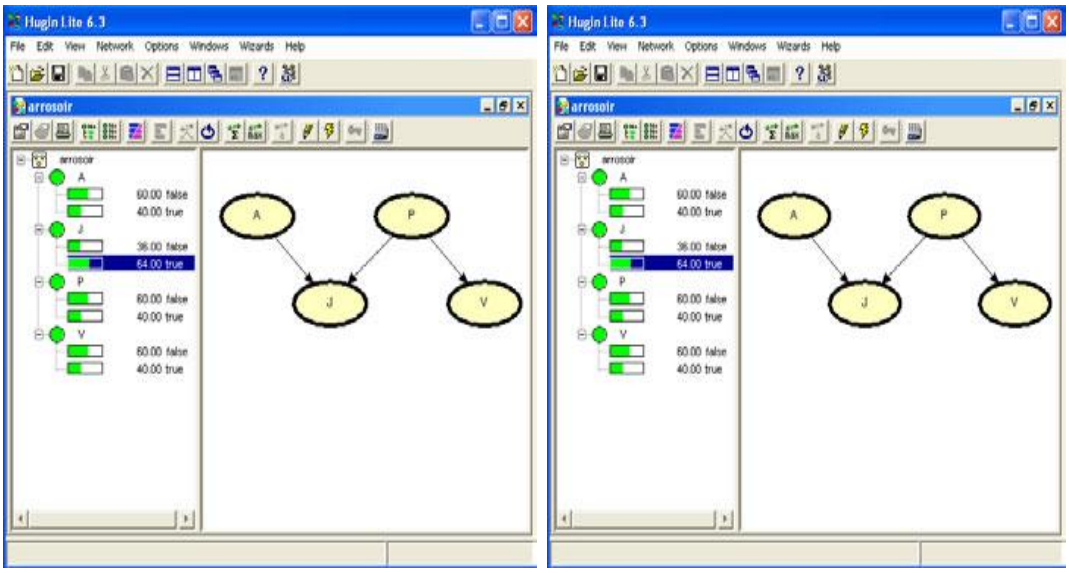


Fig. II.9 Utilisation de Hugin pour l'inférence

Le type d'inférence standard, c'est-à-dire le calcul de la probabilité des nœuds non observés conditionnellement aux observations, s'appelle la propagation Sum normal dans Hugin, qui offre d'autres modes d'inférences. En particulier, la propagation Max normal permet de trouver la configuration du réseau la plus probable, ayant effectué certaines observations

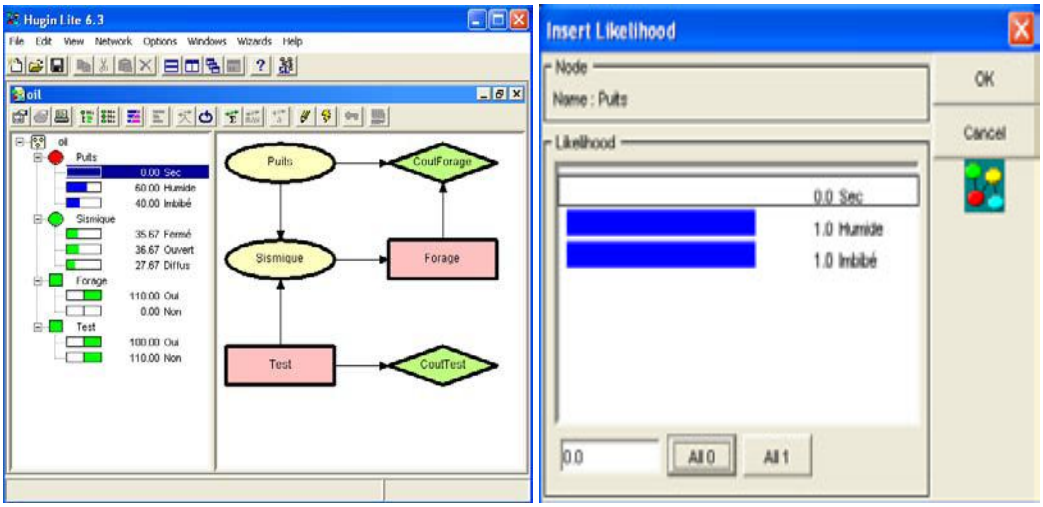


Fig. II.10 Observations partielles dans Hugin

II.10.2.3 Apprentissage

Hugin permet l'apprentissage de structure par le biais des deux algorithmes PC[6] et NPC[7].

Cette fonction est présentée comme un assistant, ou wizard, en plusieurs étapes :

- Acquisition des données : choix d'un fichier ou d'une table de base de données.

[6] : Algorithmes PC : pour évaluer s'il y a indépendance conditionnelle entre deux variables
[7] : Il tente de résoudre certains des problèmes de l'algorithme PC, principalement liée à la fiabilité de test (petits ensembles de données

- Prétraitement des données : sélection des entrées, discrétisation, etc.
- Contraintes structurelles : ici l'utilisateur peut spécifier manuellement les dépendances ou indépendances connues entre les variables.
- Apprentissage : choix de l'algorithme PC[7] ou NPC[8].
- Résolution des incertitudes : l'utilisateur est sollicité ici dans le cas où certains liens, ou certaines orientations des liens n'ont pu être établies par l'algorithme.
- Sélection des liens : l'utilisateur peut visualiser la significativité de chacun des liens, et sélectionner ceux qui dépassent un certain seuil.
- Distribution a priori : si une information sur la distribution des données est connue, on peut l'indiquer à ce stade, ainsi que le nombre d'exemples sur lesquels cette information a été obtenue.
- Apprentissage EM : c'est la dernière étape, au cours de laquelle les tables de probabilités du réseau sont apprises. [12]

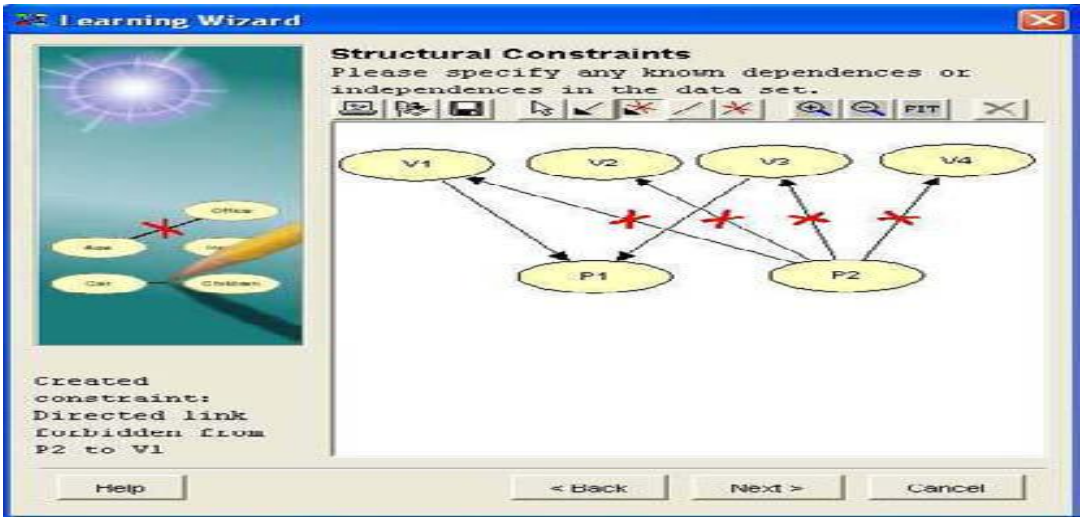


Fig.II.11 Assistant d'apprentissage de structure dans Hugin

L'apprentissage de paramètres, c'est-à-dire des tables de probabilités, peut s'effectuer à tout moment sur un réseau existant. Deux options existent pour cet apprentissage :

- L'apprentissage séquentiel, aussi appelé adaptation, permet de modifier la distribution du réseau à partir de chaque exemple observé.
- L'apprentissage global permet de recalculer les tables de probabilités du réseau à partir d'un ensemble d'exemples.

L'apprentissage global est réalisé par l'algorithme EM. Signalons enfin que Hugin peut également être utilisé pour générer des bases de cas à partir d'un réseau entièrement défini.

II.10.2.4 Compléments

Une fonctionnalité intéressante de Hugin est la possibilité de gérer des réseaux imbriqués, appelés réseaux orientés objet. Il s'agit d'insérer une instance d'un réseau déjà créé au sein d'un nouveau réseau, en le représentant par un seul nœud.[3]

Hugin offre également une API, c'est-à-dire une interface programmeur complète. Cette API est disponible en C/C++, Java, et Visual Basic.
Un langage de représentation de réseaux bayésiens permet également de créer des réseaux bayésiens par d'autres biais, pour les charger et les manipuler ensuite dans Hugin.
Un produit dérivé de Hugin, Hugin Advisor, a été créé pour faciliter le développement d'applications de diagnostic. Advisor est particulièrement adapté pour les centres d'appels de dépannage, afin de guider les opérateurs. Advisor permet en quelque sorte de systématiser l'approche des questionnaires adaptatifs qui a été présentée dans l'une des études de cas ci-dessus. La séquence de questions posées est optimisée pour aboutir le plus rapidement possible (en probabilité) à un diagnostic.

II.10.3 Bayesian network tools in Java « BNJ »

BNJ[8] (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 « Knowledge Discovery in Databases» 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.

On détaille dans la suite le contenu de cette boite à outils dans sa version 3.3+. 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 nœuds : distribution tabulaire discrète et distribution continue. Les réseaux bayésiens crées sont stockés dans des fichiers XML.

8: http: //bnj.sourceforge.net/

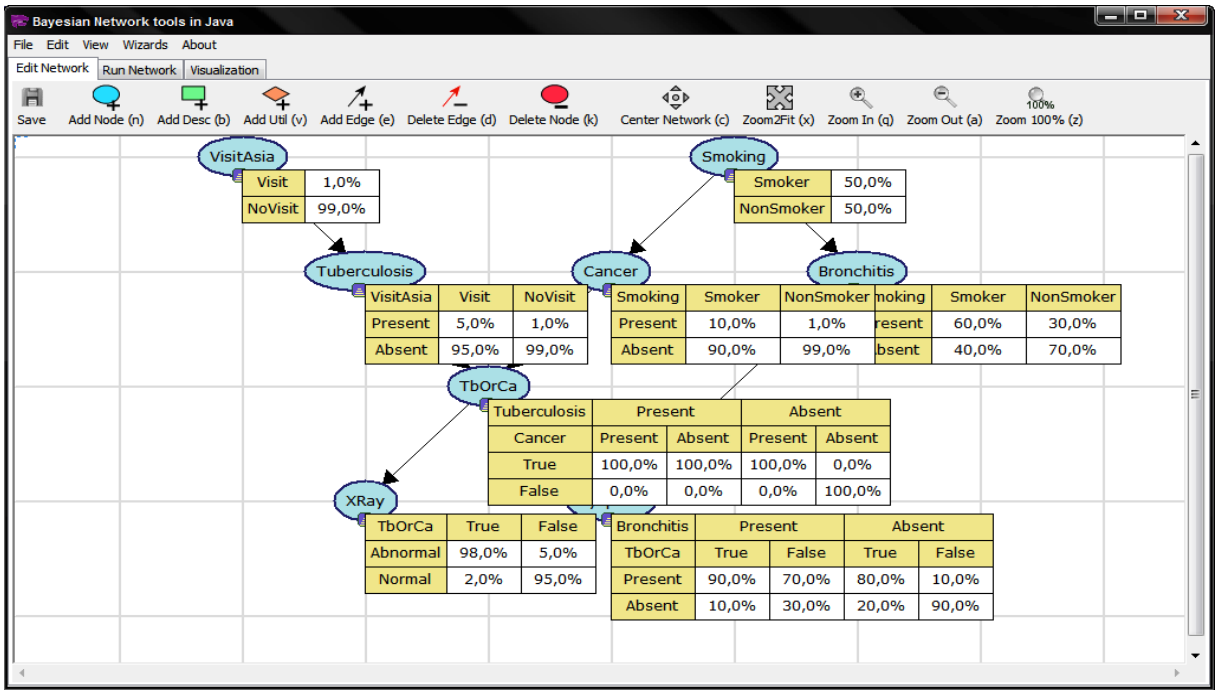


Fig. II.12 Exemple de Visit Asia sous BNJ [9]

II.10.3.1 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", "Elimination des variables avec optimisation", "Singly-connected network belief propagation"("Pearl") et "Cutset Conditioning".

- **Les algorithmes d'inférence approchée développés sont** :

Certaines méthodes utilisent la notion d'échantillonnage tel 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".

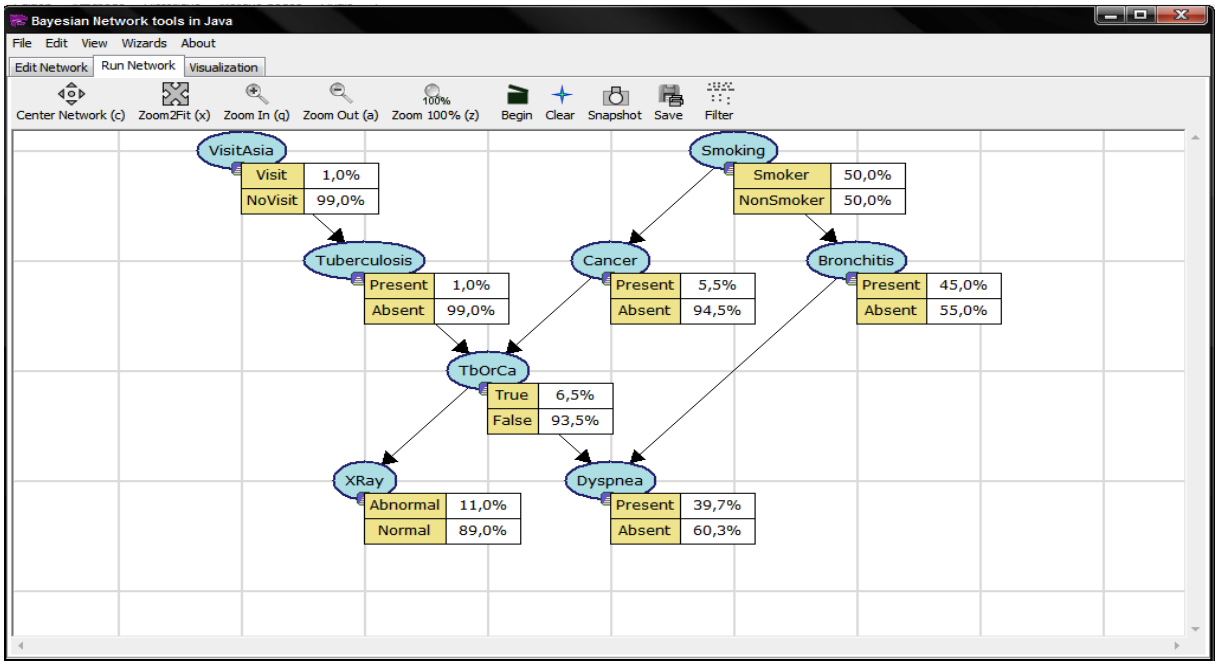


Fig. II.13 Résultats de l'inférence de l'exemple Visit Asia sous BNJ

En parcourant l'ensemble des fichiers sources de cette boite 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 2.03 a, 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 [12].

Le soin de la structure et du contenu des fichiers sources s'avère alors le seul refuge des nouveaux développeurs.

II.11 Comparaison entre les différents outils bayésien existants

Le tableau suivant illustre les différents outils existant avec les propriétés de chaque outil

- **Code src** = source inclus? (N = non) Si oui, dans quelle langage?

- **API** = interface de programme d'application inclus? (N signifie que le programme ne peut pas être intégré dans votre code, c'est à dire, il doit être exécuté comme un exécutable autonome.)
- **Exec** = Exécutable fonctionne sur W = Windows (95/98/NT), U = Unix, M = Mac, ou - = n'importe quelle machine disposant d'un compilateur.
- **Cts** = sont continues (latente) nœuds pris en charge? G = (conditionnelle) analytiquement nœuds Gaussiennes pris en charge, Cs = nœuds continus pris en charge par échantillonnage,

Cd = nœuds continus soutenus par discrétisation, Cx = nœuds continus pris en charge par une méthode non précisée, D = seuls les nœuds discrets pris en charge.

- **GUI** = interface utilisateur graphique inclus?
- **Apprentissage paramètres**?
- **Apprend structure**? CI = signifie utilise des tests conditionnels indépendantes
- **Utilité** = utilité et des nœuds de décision (par exemple, les diagrammes d'influence) sont supportés?
- **Free**? 0 = libre (bien que peut-être seulement pour un usage éducatif), = $ Logiciels commerciaux (même si la plupart ont des versions gratuites qui sont limités de diverses façons, par exemple, la taille du modèle est limitée, ou les modèles ne peuvent pas être sauvés, ou il n'ya pas d'API.)
- **Undir**? Quel genre de graphiques pris en charge? U = uniquement des graphes non orientés, D = seuls graphes orientés, UD = à la fois non-orientés et dirigés, des graphiques CG = (chaîne mélangé réalisé / non orienté).
- **Inférence** = algorithme d'inférence qui est utilisé? jtree = jonction arbre, varelim variable = (seau) l'élimination, MH = Metropols Hastings, G = échantillonnage de Gibbs, IS = importance sampling, l'échantillonnage = une autre méthode de Monte Carlo, l'algorithme polytree = Pearl limité à un graphe sans cycles, VMP = passage de message variationnelle, la propagation EP = attente, SL = le programme est conçu pour la structure d'apprentissage à partir des données observées complètement, pas d'estimation d'état.

Nom	Src	API	Exec.	Cts	GUI	App. Params	App. Struct	Echant	Utilité	Free	Undir	Inf
Analytica	N	N	W	O	W	N	N	N	O	R	N	?
Bassist	C++	O	U	O	N	O	N	O	N	O	N	MH
Bayda	Java	-	WUM	O	O	O	N	N	N	O	N	?
BayesBuilder	N	N	W	N	O	N	N	O	N	R	N	?
Bayesware Discoverer	N	N	WUM	D	O	O	O	O	O	R	N	?
B-course	N	N	WUM	D	O	O	O	N	N	O	N	?
Bayonnet	Java	-	WUM	N	O	O	N	N	N	O	N	?
Belief net power constructor	N	W	W	N	O	O	CI	N	N	O	N	?
BN Toolbox	Matlab	-	WU	O	N	O	O	O	O	O	N	Plusieurs
BN Toolkit	Visual Basic	-	W	N	O	N	O	N	N	O	N	Polytree
BucketElim	C++	-	WU	N	N	N	N	N	N	O	N	Varelim
BUGS	N	N	WU	O	W	O	N	O	N	O	N	Gibbs
Business Navigator 5	N	N	W	D	O	O	O	N	N	R	N	Jtree
Genie/Smile	N	WU	WU	N	W	N	N	O	O	O	N	Jtree
Hugin Expert	N	O	W	O	W	O	CI	O	O	R	O	Jtree
Ideal	Lisp	-	WUM	N	O	N	N	N	O	O	N	Jtree
MIM	N	N	W	O	O	O	O	N	N	R	O	Jtree
MSBNx	N	O	W	N	W	N	N	N	O	R	N	Jtree
Netica	N	WUM	W	O	W	O	N	O	O	R	N	?
Pulcinella	Lisp	-	WUM	N	O	N	N	N	N	O	N	?
RISO	Java	-	WUM	O	O	N	N	N	N	O	N	Polytree

Tab II.3 Différents outils bayésiens existant avec les propriétés[9]

II.12 Les avantages de l'approche bayesienne

À la fois outil de représentation intuitive des connaissances, et machine à calculer des probabilités conditionnelles, les réseaux bayésiens présentent les avantages suivants pour la modélisation des risques opérationnels : [12]

- La connaissance des experts n'est pas absorbée dans une boîte noire, elle est retranscrite directement.
- Les modèles sont donc contrôlables par les experts et auditables par les autorités de régulation.
- Les probabilités sont toujours le résultat de calculs simples (comptages) ou de l'expertise, renforçant ainsi la transparence des calculs effectués.
- Les réseaux bayésiens peuvent représenter l'ensemble des facteurs qui conditionnent les différentes composantes d'une vulnérabilité et permettront ainsi d'identifier les leviers de réduction et quantifier leur importance.

[9] : http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html

- Les réseaux relatifs à plusieurs vulnérabilités peuvent être interconnectés afin de mesurer les corrélations qui existent entre elles.
- Ils proposent, pour la représentation des connaissances, un formalisme commun qui sera appliqué à tous les types de risque.
- Les trois objectifs qui sont formulés pour la modélisation des risques : calculer, prévoir, comprendre sont accessibles.

II.13 Les inconvénients de l'approche bayesienne

La complexité des réseaux bayésiens ne se traduit pas seulement en termes de compréhension par les utilisateurs. Les problèmes sous-jacents sont pratiquement tous de complexité non polynomiale, et conduisent à développer des algorithmes approchés, dont le comportement n'est pas garanti pour des problèmes de grande taille.

Comme les paramètres utilisés sont maintenant issus de distributions de probabilité, il est nécessaire, pour connaître un paramètre, de calculer des intégrales faisant intervenir les distributions des autres paramètres. Il est, en général, impossible de calculer ces intégrales analytiquement, et plusieurs approches ont été proposées pour effectuer ces calculs. Mais soit ces méthodes sont très lourdes à implémenter, soit elles reposent sur des approximations qui peuvent fausser les résultats.

Dans les travaux qui utilisent les méthodes de Monte Carlo couplées à des modèles de Markov cachés pour calculer les différentes intégrales intervenant dans les différentes étapes. Les calculs sont très lourds à mettre en place et nécessitent beaucoup de temps de calcul.

Nous avons regroupé avantages et inconvénients selon les trois rubriques, l'acquisition, la représentation et l'utilisation des connaissances. La représentation adoptée est la suivante :

- À chaque ligne correspond une caractéristique, qui peut être un avantage, ou la prise en compte d'un problème spécifique.
- Si la technique considérée permet de prendre en compte ce problème, ou présente cet avantage, un signe + est placé dans la case correspondante.
- Un signe ⋆ est placé dans la case de la meilleure technique du point de vue de la caractéristique considérée.

Connaissances	*Analyse de données*	*Réseaux neuronaux*	*Arbres de décision*	*Systèmes experts*	*Réseaux bayésiens*
ACQUISITION					
Expertise seulement				★	
Données seulement	+	★	+		+
Mixte	+	+	+		★
Incrémental		+			★
Généralisation	+	★	+		+
Données incomplètes		+			★
REPRÉSENTATION					
Incertitude				+	★
Lisibilité	+		+	+	★
Facilité		+	★		
Homogénéité					★
UTILISATION					
Requêtes élaborées	+			+	★
Utilité économique	+	+			★
Performances	+	★			

Tab. II.4 Les avantages comparatifs des réseaux bayésiens [12].

II.14. Conclusion

La généralité du formalisme des réseaux bayésiens aussi bien en termes de représentation que d'utilisation les rend difficiles à manipuler à partir d'une certaine taille, et conduisent à développer des algorithmes approchés, dont le comportement n'est pas garanti pour des problèmes de grande taille.

Selon le type d'application, l'utilisation pratique des réseaux bayésiens peut être envisagée au même titre que celle d'autres modèles : réseaux de neurones, systèmes experts, arbres de décision, etc. Les aspects suivants des réseaux bayésiens les rendent, dans de nombreux cas, préférables à d'autres modèles :

- **Acquisition des connaissances :** Les réseaux bayésiens donnent la possibilité de rassembler et de fusionner des connaissances de diverses natures dans un même modèle : retour d'expérience (données historiques ou empiriques), expertise (exprimée sous forme de règles logiques, d'équations, de statistiques ou de probabilités subjectives), observations.

Dans le monde industriel, chacune de ces sources d'information est souvent insuffisante individuellement pour fournir une représentation précise et réaliste du système analysé.

- **Représentation des connaissances :** La représentation graphique d'un réseau bayésien est explicite, intuitive et compréhensible par un non spécialiste, ce qui facilite à la fois la validation du modèle, ses évolutions éventuelles et surtout son utilisation. Typiquement, un décideur est beaucoup plus enclin à s'appuyer sur un modèle dont il comprend le fonctionnement qu'à faire confiance à une «boîte noire».

- **Qualité de l'offre en matière de logiciels :** Il existe aujourd'hui de nombreux logiciels pour saisir et traiter des réseaux bayésiens. Ces outils présentent des fonctionnalités plus ou moins évoluées : apprentissage des probabilités, apprentissage de la structure du réseau bayésien, possibilité d'intégrer des variables continues, des variables d'utilité et de décision etc. [12]

Les Réseaux Bayésiens représentent un outil de choix dans la représentation de connaissances et dans l'exploitation de celles-ci. Par ailleurs, plusieurs domaines sont intéressés par ce type de représentation. En fait, l'inférence sur les réseaux bayésiens est un problème NP-difficile[2], c'est pourquoi il était convenable de le voir de façon complète pour des instances réalisables et incomplète dans les autres cas. L'identification structurelle de réseaux bayésiens à partir de données est un problème de forte complexité algorithmique, encore mal résolu à ce jour pour des jeux de données de grande taille.

[10] : NP est une classe de complexité mathématique

Chapitre III :

Conception du système

III.1.Introduction

Le but de ce travail est d'offrir aux concepteurs et aux mainteneurs un moyen de prédire les conséquences des changements qu'ils comptent effectuer sur des systèmes (programmes) à objets. En effet, comme l'affirment « plus on comprend un changement, mieux on peut le contrôler et donc minimiser son risque ». Le choix d'une approche probabiliste est justifié par, deux arguments. D'une part, l'analyse de l'impact se fait avant que le système ne soit réellement modifié et donc affecté. Ceci permet de réduire le coût et offre aux mainteneurs le choix entre plusieurs scénarios de maintenance selon l'ensemble des classes prédites comme affectées. D'autre part, avec cette approche probabiliste,

Nous proposons une approche probabiliste afin de déterminer l'impact des changements dans les programmes à objets. Cette approche sert à prédire, pour un changement donné dans une classe du système, l'ensemble des autres classes affectées par ce changement. Cette prédiction est donnée sous la forme d'une probabilité qui dépend des relations extraites à partir du code source du système considéré.

L'approche probabiliste proposée est évaluée par la mise en œuvre de plusieurs types de changements effectués sur différents systèmes orienté objet.

Une fois la structure du réseau est mise en place et que les tables de probabilité sont définies, le réseau bayésien est utilisé pour calculer des probabilités. Ce procédé est appelé l'inférence bayésienne.

L'inférence bayésienne est définie par Naim et al. [12] comme « le processus de propager une ou plusieurs informations certaines au sein d'un réseau pour en déduire comment sont modifiées les croyances concernant les autres nœuds ». En d'autres termes, l'inférence sert à calculer la probabilité d'une hypothèse suite à l'observation des évidences.

Les évidences correspondent aux nœuds d'entrée et les hypothèses sont les différents états des nœuds de sortie du réseau. L'injection des probabilités des nœuds d'entrée va modifier récursivement les probabilités des nœuds enfants jusqu'aux nœuds de sortie. Le calcul des probabilités utilise à la fois les tables de probabilités et le théorème de Bayes.

Nous présentons dans un second temps, l'architecture générale et les différents modules qui composent notre système « outil». Enfin nous montrons le principe détaillé des deux algorithmes choisies leurs déroulements.

III.2 Partie 1 : Conception du modèle « Réseau d'impact»

Notre objectif dans ce projet est d'améliorer la maintenance des systèmes à objets et d'intervenir plus précisément dans la tâche de l'analyse et de la prédiction de l'impact du changement. En identifiant l'impact potentiel d'une modification, on réduit le risque d'entamer des changements coûteux et imprévisibles. Pour cela, nous essayons de donner plus d'explications sur les facteurs réels et responsables de cet impact du changement ainsi que son évolution.

Les réseaux bayésiens (RBs) constituent une approche quantitative particulière qui peut intégrer l'incertitude dans le raisonnement, l'utilisation de réseau bayésien nous aide à faire l'analyse et la prédiction de l'impact du changement et de ses conséquences sur le reste du système.

Le modèle d'impact de changement (avec les réseaux bayésiens pour notre approche) permet de prédire les classes qui seront affectées en cas ou un tel changement avait été réellement fait.

III.2.1 Le réseau bayésien utilisé

Le réseau bayésien de notre problématique d' impact de changement a été tiré de [3].
Abdi,al [3] se sont intéressés à la compréhension des facteurs réels qui sont responsables de l'impact de changement et de son évolution. Des métriques de conception et d'implémentation sont étudiées afin de comprendre leurs effets sur les systèmes. Les auteurs ont proposé une approche probabiliste qui utilise les réseaux bayésiens pour déterminer l'impact des changements comme illustré dans la Figure suivante. Les nœuds des réseaux bayésiens sont donc décomposés en deux catégories :

- Les noeuds d'entrée : représentant des métriques que les auteurs ont classifiées en des métriques de conception et d'implémentation.
- Les noeuds intermédiaires regroupent les métriques selon les deux types cités ci-dessus.

Il s' agit d' un réseau bayésien reliant quelques métriques (mesurant certaines propriétés architecturelles du système considéré) à l' impact de changement. Un modèle d'impact a été construit et des probabilités ont été affectées aux différents sommets du réseau. Pour être

conformes à l'hypothèse de Bayes, nous considérons que les métriques mesurent différents types d'aspects de couplage et que leurs valeurs sont donc indépendantes les unes des autres.

Abdi et al [3] ont choisi pour cette étude empirique, un système disponible a notre niveau. Il s'agit de BOAP (Boite a Outils pour l'Analyse de Programmes) développé au Centre de Recherche Informatique de Montréal (CRIM). C'est un ensemble d'outils logiciels intégrés, qui permet a un expert d'évaluer rapidement le niveau de qualité d'un logiciel (faiblesses conceptuelles ou structurelles, instructions trop complexes, etc.). Le système BOAP (version 1.1.0) contient en tout 394 classes.

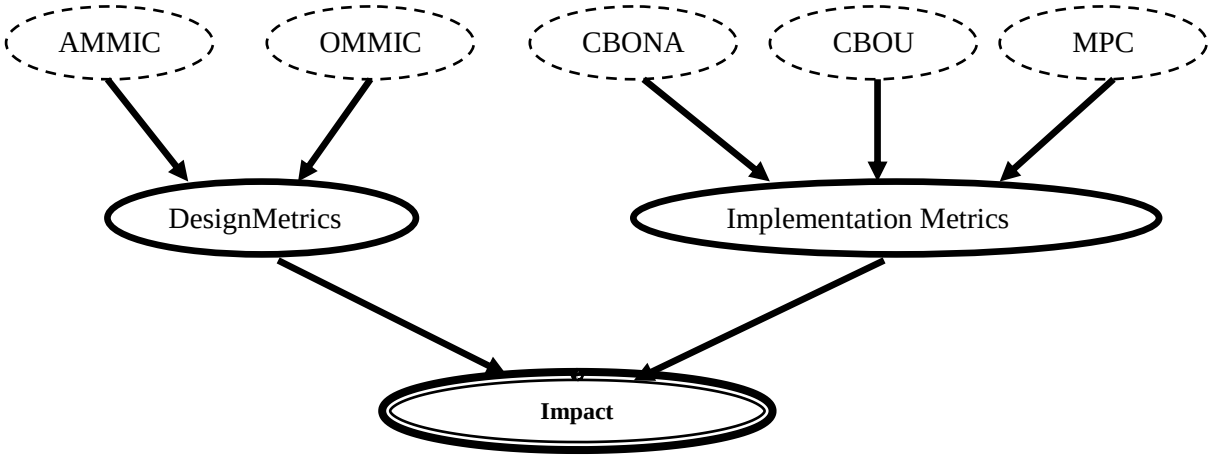
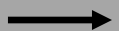


Fig. III.1 Réseau d'impact du changement [3]

III.2.2 Les différents composants du réseau d'impact de changement

Composants du réseau d'impact	Description
	**Les sommets d'entrée** représentent les différentes métriques. Toutes ces variables d'entrée sont des variables quantitatives qui ont des valeurs numériques mesurables.
	**Les sommets intermédiaires** ne sont pas directement mesurables. Ils sont définis ou influencés par leurs sommets parents. Les sommets intermédiaires ont une table de probabilité associée.
	**Le résultat final** qui représente l'impact de changement du système à partir des nœuds parents.
	**Les arcs** ce sont les relations qui relient les nœuds parent et les nœuds fils

Tab. III.1 Composants du réseau d'impact de changement

Certaines de ces métriques sont considérées comme des métriques de conception (AMMIC et OMMIC), car pouvant être utilisées dès la phase de conception, alors que d'autres, sont des métriques d'implémentation (MPC, CBOU, CBONA), car nécessitant la présence du code source.

III.2.3 Affectation des paramètres

Les nœuds du réseau bayésien sont donc décomposés en deux catégories : les variables d'entrée et les variables intermédiaires.

Les probabilités des sommets d'entrée sont déterminées directement à partir des mesures de ces variables données par un système de test. Dans notre cas, il s'agit de BOAP (Boîte à Outils pour l'Analyse de Programmes) développé au Centre de Recherche Informatique de Montréal (CRIM) (Alikacem et Snoussi, 2002) [62]. C'est un ensemble d'outils logiciels intégrés, qui permet à un expert d'évaluer rapidement le niveau de qualité d'un logiciel (faiblesses conceptuelles ou structurelles, instructions trop complexes, etc.). Le système BOAP (version 1.1.0) contient en tout 394 classes. Les métriques considérées dans ce travail sont extraites de ce système.

III.2.3.1 Les sommets d'entrée

Représentent les différentes métriques sélectionnées. Toutes ces variables d'entrée sont des variables quantitatives qui ont des valeurs numériques mesurables. Le nombre de valeurs possibles pour ces variables peut être infini, en fonction du système de test considéré. Afin de faciliter la définition des probabilités des sommets d'entrée, nous avons besoin de transformer ces variables en variables discrètes ayant un nombre limité de valeurs sont : « petit » et « grand ».

Dans la table suivante ; les probabilités servent à définir la tps du sommet AMMIC un des nœuds d'entrée: 0.35 % d'être « Petit » et avec 0.65 « Grand»

Petit	0.35
Grand	0.65

Table. III.1 TPS du sommet d'entrée AMMIC.

III.2.3.2 Les sommets intermédiaires

Représentent les deux nœuds DesignMetrics et ImplementationMetrics, ces nœuds ne sont pas directement mesurables. Ils sont définis ou influencés par leurs sommets parents. Les sommets intermédiaires ont une table de probabilité associée. DesignMetrics un des nœuds intermédiaire du réseau impact, la variable DesignMetrics est définie par ses deux parents AMMIC et OMMIC. Il s'agit de trouver la probabilité conditionnelle du sommet DesignMetrics : P (DesignMetrics / AMMIC & OMMIC). Or comme la relation entre les sommets parents AMMIC et OMMIC et leur sommet fils DesignMetrics est définitionnelle, la forte présence de ces métriques définit également la forte présence des métriques de conception (DesignMetrics). Un scénario possible pour la TPS du sommet DesignMetrics est présenté à la table suivante :

AMMIC	Petit		Grand	
OMMIC	Petit	Grand	Petit	Grand
Oui	0.2	0.4	0.4	0.8
Non	0.8	0.6	0.6	0.2

Table. III.2.3.2 TPS du sommet intermédiaire DesignMetrics.

Le tableau suivant montre les nœuds d'entrés du réseau impact et leurs définitions.

Métrique	**Définition**
CBOU	➢ Coupling Between Object : nombre de classes avec lesquelles une classe est couplée
CBONA	➢ CBO No Ancestors : CBO sans considérer les classes ancêtres.
MPC	➢ Message Passing Coupling : nombre de messages envoyés par une classe en direction des autres classes du système.
OMMIC	➢ Others Method–Method Import Coupling : nombre de classes (autres que les super-classes et les sous-classes) avec lesquelles une classe a une interaction de type méthode-méthode et un couplage de type IC (Import Coupling).
AMMIC	➢ Ancestors Method–Method Import Coupling : nombre de classes parentes avec lesquelles une classe a une interaction de type méthode-méthode et un couplage de type IC (Import Coupling).

Tab. III.2 Métriques utilisées dans le réseau d'impact

III.3 Partie 2 : Conception du système

III.3.1 Rappel sur le réseau bayésien

Un *réseau bayésien* est défini par :

- un graphe orienté sans circuit (*DAG*) G = (V, E), où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G;
- un *espace probabilisé* fini (Ω,Z, p) ;
- un ensemble de *variables aléatoires* associées aux nœuds du graphe et définies sur (Ω, Z, p), tel que [12] :

$$P(V_1,V_2,.....,V_n)= \Pi_{i=1..n} [\, p(V_i \mid C(V_i)) \,]$$

Où Ω : *un* ensemble de définition fini non vide « *univers*», Z : Nœud , p : est une probabilité $C(V_i)$ est l'ensemble des causes (parents) de V_i dans le graphe G.

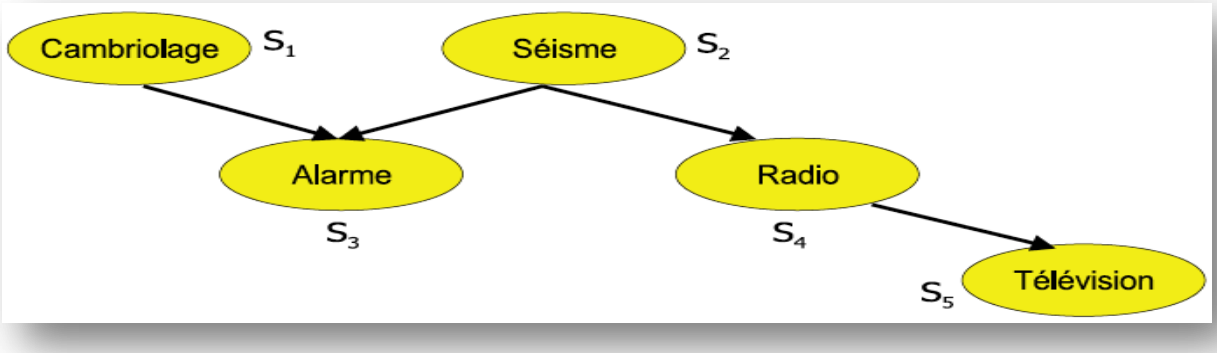
C'est exactement ce que nous avons construit sur les deux exemples ci-dessous.

Il existe deux volets essentiels pour l'élaboration de réseaux bayésiens :

- construction de la structure du réseau bayésien
- définition des probabilités associée. « Table de probabilité»

Exemple :

L'exemple suivant représente un Réseau bayésien modélisant la probabilité de déclencher une alarme :



III.2 Réseau bayésien d'un système d'alarme

III.3.2 La table de probabilité

Une fois que les variables (nœuds) sont représentées et que les relations entre elles sont identifiées, la deuxième étape est la génération des tables de probabilités relatives à chaque nœud.

Cette table exprime la probabilité conditionnelle entre un nœud fils et ses parents. En effet, les arcs dans un réseau bayésien définissent une relation entre un nœud père (nœud de départ) et son enfant (nœud d'arrivée). Cette relation entre pères et fils se reflète dans la table de probabilité relative à chaque nœud. Ainsi la table de probabilité relative à une variable a la forme de probabilité (variable | parents) ou littéralement la probabilité d'une variable « enfant » sachant les états des variables « parents».

- **Exemple**

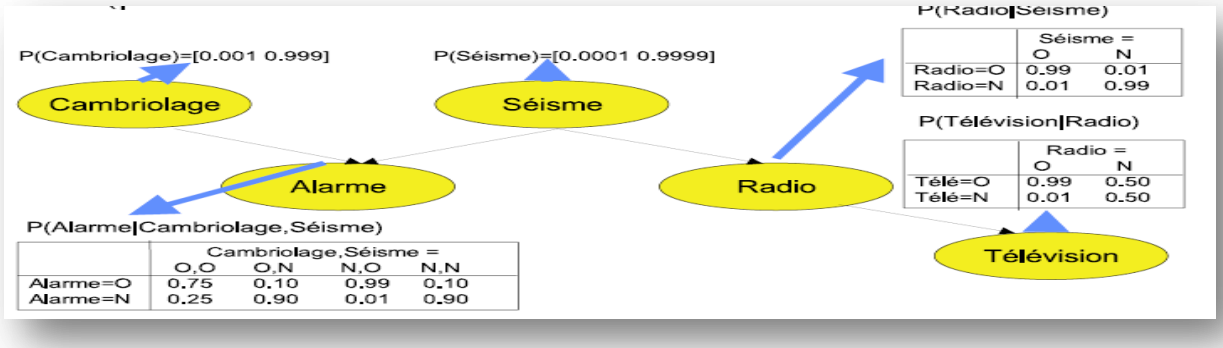


Fig. III.3 Table de probabilité dans un réseau bayésien

III.3.3 Les algorithme d'inférence utilisés

Nous avons utilisé l'algorithme d'inférence de l'approche exacte, qui est détaillé comme suit :

III.3.3.1 Arbre de jonction

Il résout le problème de l'identification du maximum a posteriori (MAP) avec une complexité en temps. La méthode de l'arbre de jonction (clustering ou clique-tree propagation algorithm) a été introduite par Lauritzen & Spiegelhalter et Jensen, Lauritzen & Olesen. Elle est aussi appelée méthode JLO (pour Jensen, Lauritzen, Olessen). Elle est applicable pour toute structure de DAG contrairement à la méthode des messages locaux. Néanmoins, s'il y a peu de circuits dans le graphe, il peut être préférable d'utiliser une méthode basée sur un ensemble de coupe.
Cette méthode est divisée en cinq étapes qui sont :

- Moralisation du graphe,
- Triangulation du graphe moral,
- Construction de l'arbre de jonction,
- Inférence dans l'arbre de jonction en utilisant l'algorithme des messages locaux,
- Transformation des potentiels de clique en lois conditionnelles mises à jour.

Les étapes sont organisées dans les deux phases suivantes :

➢ **la phase de construction** : elle nécessite un ensemble de sous-étapes permettant de transformer le graphe initial en un arbre de jonction, dont les nœuds sont des clusters (regroupement) de nœuds du graphe initial. Cette transformation est nécessaire, d'une part pour éliminer les boucles du graphe, et d'autre part, pour obtenir un graphe plus efficace quant au temps de calcul nécessaire à l'inférence, mais qui reste équivalent au niveau de la distribution de probabilité représentée.

Cette transformation se fait en trois étapes :

- la moralisation du graphe,
- la triangulation du graphe et l'extraction des cliques qui formeront les nœuds du futur arbre,
- la création d'un arbre couvrant minimal, appelé arbre de jonction ;

➢ **la phase de propagation** : il s'agit de la phase de calcul probabiliste à proprement parler où les nouvelles informations concernant une ou plusieurs variables sont propagées à l'ensemble du réseau, de manière à mettre à jour l'ensemble des distributions de probabilités du réseau. Ceci ce fait en passant des messages contenant une information de mise à jour entre les nœuds de l'arbre de jonction précédemment construit. A la fin de cette phase, l'arbre de jonction contiendra à la distribution de probabilité sachant les nouvelles informations, c'est-à-dire P(Uje) où U représente l'ensemble des variables du réseau bayésien et e l'ensemble des nouvelles informations sur les dites variables.

a) Moralisation

La moralisation se décompose suivant les étapes suivantes :

- Mariage des nœuds parents : pour les nœuds possédant plusieurs parents, liaison des parents deux à deux avec des arcs supplémentaires.

- Récupération du squelette du graphe ainsi obtenu, Nous obtenons alors un graphe non dirigé dit moralisé. Le déroulement de cet algorithme sera illustré sur l'exemple précédent

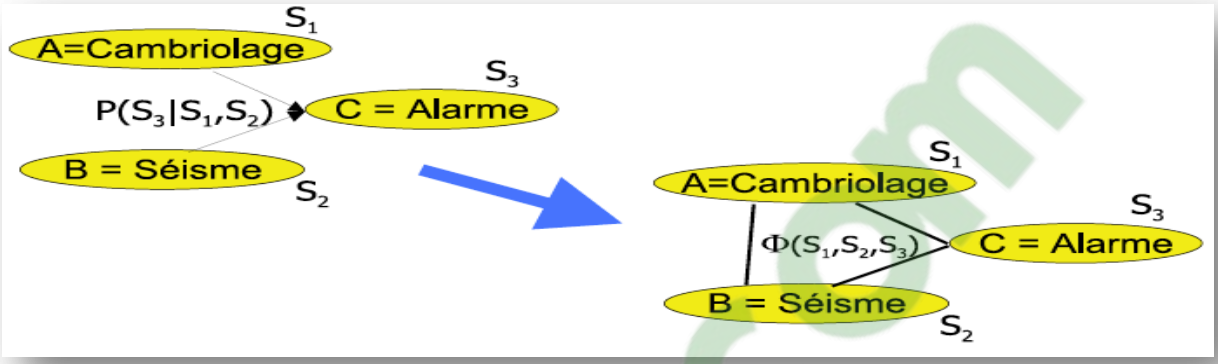


Fig. III.4 Moralisation du graphe « Arbre de jonction»

b) Triangulation

La deuxième étape consiste à trianguler le graphe moral G_m et en extraire des cliques de nœuds, qui sont des sous graphes complets de G. Ces cliques formeront les nœuds de l'arbre de jonction utilisé pour l'inférence. Il faut donc ajouter suffisamment d'arcs au graphe moral G_m afin d'obtenir un graphe triangulé G_T.

L'algorithme de triangulation opère d'une manière très simple. Un graphe est triangulé si est seulement si l'ensemble de ses nœuds peuvent être éliminés. Un nœud peut être éliminé si tous ses voisins sont connectés deux à deux. Donc un nœud peut être éliminé s'il appartient à une clique dans le graphe. Une telle clique forme un nœud pour le futur arbre de jonction qui est en train d'être construit. Ainsi, il est possible de trianguler le graphe et de construire les nœuds de l'arbre de jonction en même temps en éliminant les nœuds dans un certain ordre. Si aucun nœud n'est éliminable, il faut en choisir un parmi les nœuds restants et rajouter les arcs nécessaires entre ses voisins pour qu'il devienne éliminable. Le nœud choisi sera celui pour lequel l'espace d'état de la clique formée sera le plus petit possible. En effet, plus les cliques sont petites, plus l'espace de stockage, et à fortiori le temps de calcul, sont réduits.

L'efficacité de l'algorithme JLO reste dépendant de la qualité de la triangulation. Mais trouver une bonne triangulation dépend de l'ordre d'élimination des variables. D'une manière générale, trouver une triangulation optimale pour des graphes non-dirigés reste un problème NP-difficile

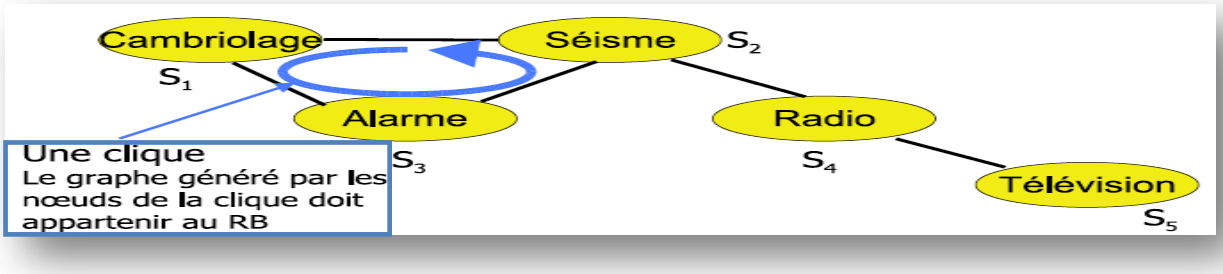


Fig. III.5 Triangulation « Arbre de jonction»

Si le graphe est moralisé et triangulé, alors les cliques peuvent être organisées en un arbre de jonction.

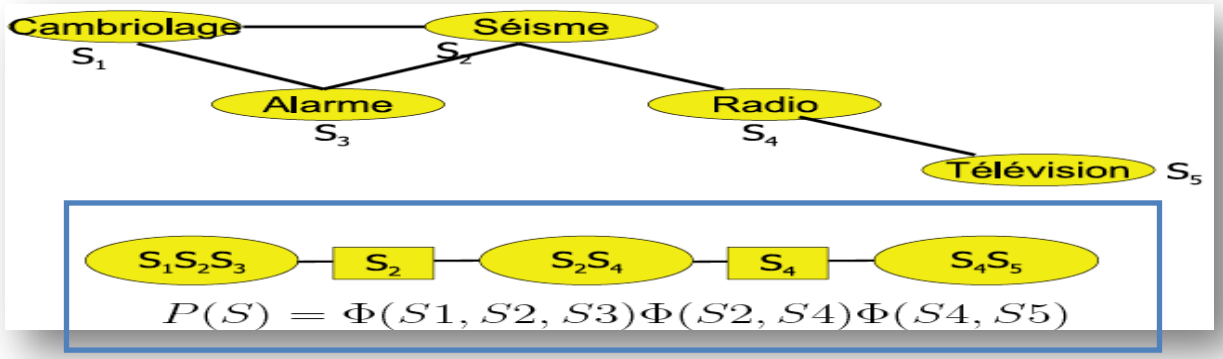


Fig. III.6 Arbre de jonction

III.4 Fonctionnement du système

Le système Impact Bayes Net est un ensemble d'outils pour la création et la manipulation de réseaux bayésiens. L'éditeur graphique du système vous permet de créer et modifier des réseaux bayésiens. Il vous permet aussi d'importer des réseaux bayésiens en deux formats « XML, BIF» (voir la section V.3.9.2).

L'éditeur des réseaux bayésien est un outil de manipulation pour faire les différentes tâches essentielles d'édition, il est composé de :

1- La fenêtre d'édition du graphe

- Créer, Déplacer, Supprimer « Nœud/Arc » ;
- Observer ;
- Interroger ;
- Modifier «Nœud/Table de Probabilité/Réseau Bayésien »

2- Le menu

- Ficher « ouvrir, enregistrer, nettoyer, décharger la console, quitter» ;
- Option « Algorithmes d'inférence, formats d'enregistrement» ;
- Aide «A-propos »

La figure IV.7 ci dessous représente une architecture générale d'éditeur bayésien « Impact Bayes Net» qui illustre les composants et le taches de fonctionnement du système.

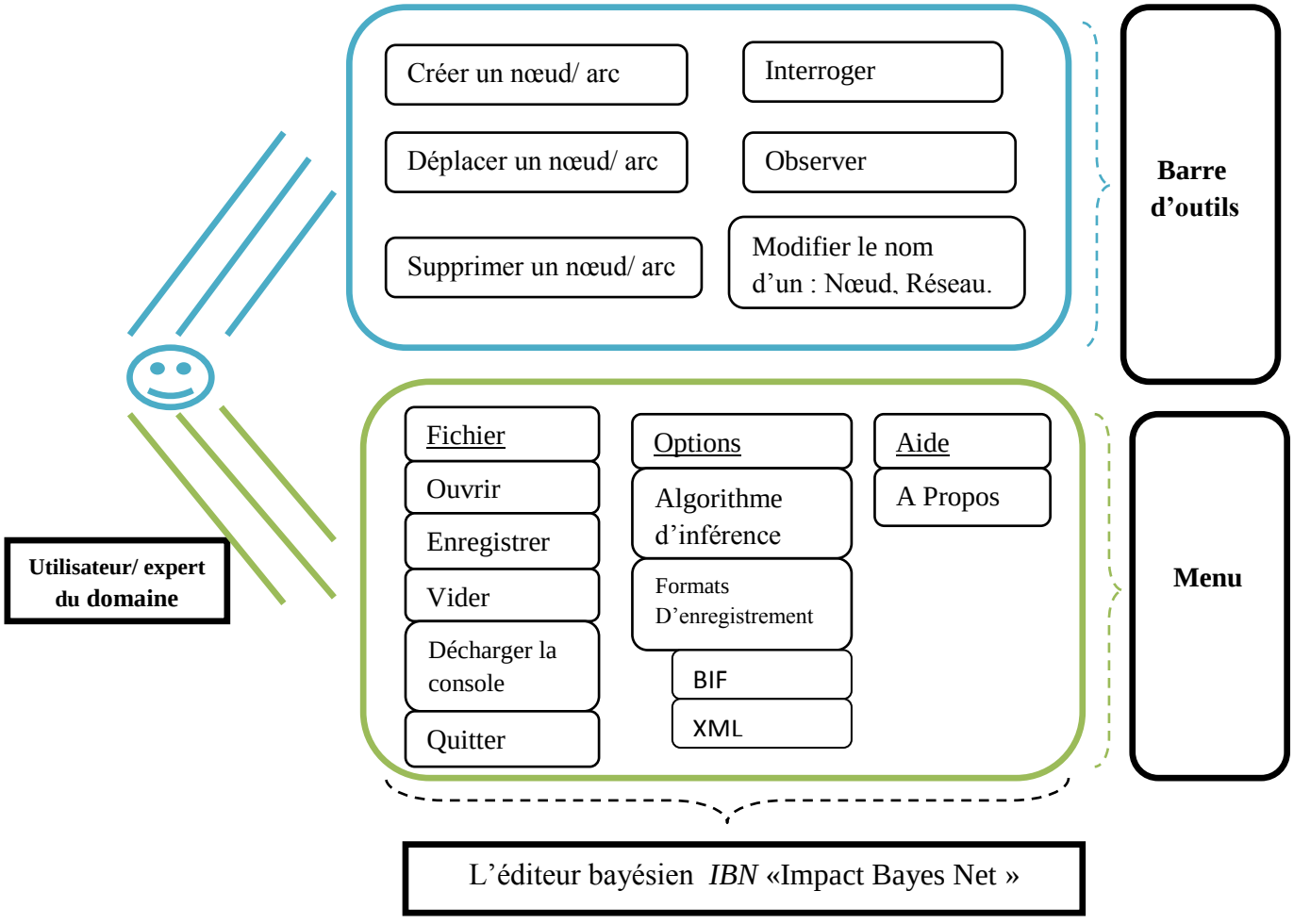


Fig. III.7 Architecture générale d'éditeur bayésien « Impact Bayes Net »

III.4.1 Algorithme d'inférence

Pour le choix d'algorithme d'inférence, nous avons opté pour l'algorithme d'arbre de jonction.

III.4.1.1 Algorithme d'arbre de jonction « Jonction Tree»

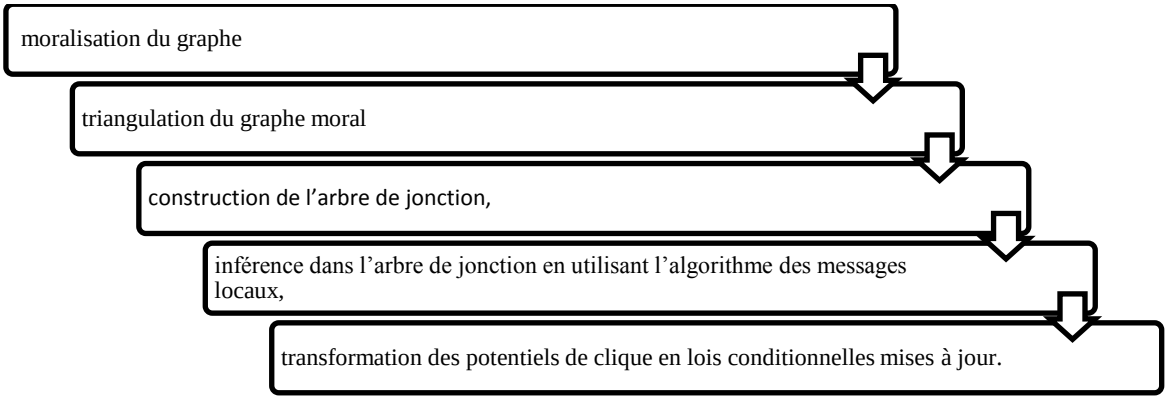


Fig. III.8 Etapes d'algorithme « Arbre de jonction»

Voici le pseudo algorithme d'arbre de jonction qui montre la propagation de l'information dans un arbre de jonction. Le principe avec l'algorithme est de transmettre l'information de cliques en cliques. Pour cela, la notion de flux d'une clique Ci `a une clique C_j ($\forall(i, j) \in \{1, \ldots, n\}$) est introduite. Or ce flux entre deux cliques est définie grâce au séparateur S_{ij} Si, j pour désigner le séparateur mis $x_i \cap x_j$) . Tout d'abord le flux initial de S_{ij} est mis à jour par marginalisation du potentiel des cliques `a travers les variables qui sont dans Ci mais pas dans S_{ij} :

Algorithme : Arbre de jonction « propagation de l'information entre les clique de l'arbre»

Tant que quelques cliques c n'ont pas reçu de messages de tous leurs voisins **Faire**

Pour $c \in C$ Faire

Si x_c a reçu des messages de tous ses voisins, **Alors**

Pour $x_k \in \Gamma x_c$ **alors** /* Γx_c l'ensemble des voisins de x_c

Calculer et envoyer $M_{c \longrightarrow k}$ /* s'il n'a pas été envoyé

Fin

Fin

Sinon Si x_c a reçu des messages de tous les voisins sauf x_k **Alors**

Calculer et envoyer $M_{c \longrightarrow k}$ /* M : message envoyer de c au k

Fin

Fin

Pour $c \in C$ **Faire**

Calcul $D_c(x_c)$

Fin

Fin

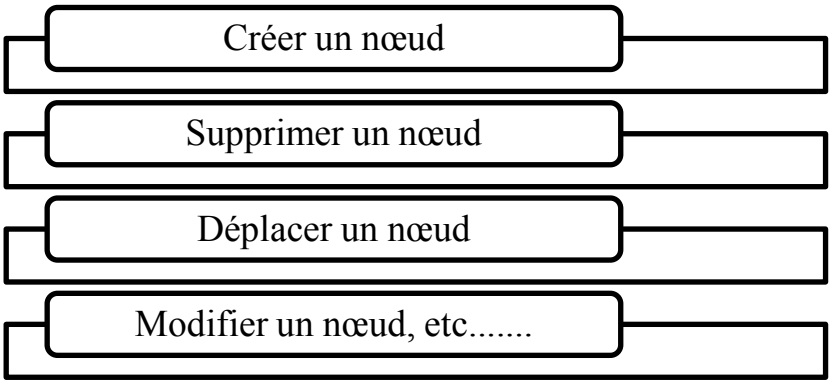
Retour D ; / * Le vecteur des probabilités marginales * /

III.5 La console

Dans la console, une fois une inférence est effectuée. ***Impact Bayes Net*** affiche un court message indiquant les valeurs finales dans une requête. ***Impact Bayes Net*** pouvez également afficher l'ensemble du réseau qui est utilisé dans le traitement d'une requête. Notez que pour afficher réellement les informations que vous souhaitez, vous devez effectuer une requête sur le réseau.

Pour effacer le texte dans la console ***Impact Bayes Net*** et envoyer le contenu de la console vers un fichier, utilisez l'option décharger la console dans le menu Fichier. Cette option demande un nom de fichier, et décharge tout le contenu de la fenêtre de console dans le fichier indiqué, et la fenêtre de la console sera effacer.

La console est le lieu des messages destinés à l'utilisateur. Par exemple comment faire pour :



III.6 Conclusion

Au cours de ce chapitre nous avons fait une analyse du problème que nous avons spécifié au niveau de l'introduction de ce même chapitre. Puis nous avons présenté l'architecture générale qui illustre le fonctionnement global de l'éditeur bayésien.

Nous avons détaillé la conception de chacun des éléments composants le système ainsi que la relation de certains composants avec l'acteur humain. Enfin, nous avons présenté l'algorithme choisi pour la tâche d'inférence.

Chapitre IV :

Implémentation du système

IV.1 Introduction

Les réseaux bayésiens ont été utilisés comme un outil fondamental pour la représentation et la manipulation des croyances en intelligence artificielle. Il ya eu des implémentations de réseaux bayésiens dans une variété de formats et de langues .

IBN « Impact Bayes Net» est un système qui gère les réseaux bayésiens: il calcule les probabilités de chaque nœud dans le réseau, et permet à l'utilisateur d'importer, créer, modifier et exporter des réseaux.

IBN est une implémentation complète des réseaux bayésiens en Java. L'un des avantages évidents de ce langage est une bibliothèque d'exécution qui se veut indépendante de la plateforme:
En théorie, il vous est possible d'utiliser le même code pour Windows 95/98/NT, Solaris UNIX Macintosh, etc. Cette propriété est indispensable pour une programmation sur Internet (cependant, par rapport à la disponibilité sur Windows et Solaris les implémentations sur d'autres plates-formes ont toujours un léger décalage).

Le système ***IBN*** est un ensemble d'outils pour la création et la manipulation de réseaux bayésiens. Le système est composé d'un éditeur graphique, L'éditeur graphique vous permet de créer et de modifier des réseaux bayésiens dans une interface conviviale. Il permet d'importer des réseaux bayésiens dans une variété de formats. IBN responsable pour manipuler les structures de données qui représentent des réseaux bayésiens. il peut produire:

- la probabilité marginale pour une variable dans un réseau bayésien.
- configurations avec un maximum de probabilité a posteriori

IV.2 Langage java (l'IDE JBuilderX)

Java est un langage de programmation développé par Sun Microsystems. Il n'a que quelques années de vie (les premières versions datent de 1995), et pourtant il a réussi à intéresser et intriguer beaucoup de développeurs à travers le monde. Et pourquoi donc ? La réponse est vaste et forcément sujet à polémiques.

Nous avons développée notre application sur PC portable Dell inspiron, un processeur intel Core 2 Duo avec la RAM de 1,90 Go, sous Windows XP Sweet 5.1 pack 3. L'application est développée par le langage de programmation Java.

IV.2 .1 Caractéristiques du langage

Les créateurs de Java ont écrit un livre blanc qui présent les caractéristiques fondamentales de Java. Ce livre est articulé autour des termes suivants :

• Distribué

Java possède une importante bibliothèque de routines permettant de gérer les protocoles TCP/IP tels que HTTP et FTP. Les applications Java peuvent charger et accéder à des sur Internet via des URL avec la même facilité qu'elles accèdent à un fichier local sur le système.

« les fonctionnalités réseau de Java sont à la fois fiables et d'utilisation aisée. Toute personne ayant essayé de faire de la programmation pour Internet avec un autre langage se réjouira de la simplicité de Java lorsqu'il s'agit de mettre en oeuvre des tâches lourdes, comme l'ouverture d'une connexion avec un socket. De plus, Java rend plus facile l'élaboration des scripts CGI (Common Gateway Interface), et un mécanisme élégant, nommé servlet, augmente considérablement l'efficacité du traitement côté serveur, assuré par Java. De nombreux serveurs Web, parmi les plus courants, supportent les servlets. Le mécanisme d'invocation de méthode à distance (RMI) autorise la communication entre objets distribués. »

• Fiabilité

Java a été conçu pour que les programmes qui l'utilisent soient fiables sous différents aspects. Sa conception encourage le programmeur à traquer préventivement les éventuels problèmes, à lancer des vérifications dynamiques en cours d'exécution et à éliminer les situations génératrices d'erreurs... La seule et unique grosse différence entre C++ et Java réside dans le fait que ce dernier intègre un modèle de pointeur qui écarte les risques d'écrasement de la mémoire et d'endommagement des données.

• Orienté objet

Pour rester simples, disons que la conception orientée objet est une technique de programmation qui se concentre sur les données (les objets) et sur les interfaces avec ces objets. Pour faire une analogie avec la menuiserie, on pourrait dire qu'un menuisier "orienté objet " s'intéresse

essentiellement à la chaise l'objet qu'il fabrique et non à sa conception (le "comment"). Par opposition, le menuisier "non orienté objet " penserait d'abord au "comment "...

• **Simple**

Nous avoirs voulu créer un système qui puisse être programmé simplement sans nécessiter un apprentissage ésotérique, et qui tire parti de l'expérience standard actuelle. En conséquence, même si nous pensions que C++ ne convenait pas, Java a été conçu de façon relativement proche de ce langage dans le dessein de faciliter la compréhension du système. De nombreuses fonctions compliquées, mal comprises, rarement utilisées de C++, qui semblaient plus d'inconvénients que d'avantages, ont été supprimées de Java.

• **Sécurité**

Java a été conçu pour être exploité dans des environnements serveur et distribués. Dans ce but, la sécurité n'a pas été négligée. Java permet la construction de systèmes inaltérables et sans virus.

• **Architecture neutre**

Le compilateur génère un format de fichier objet dont l'architecture est neutre, le code compilé est exécutable sur de nombreux processeurs, à partir du moment où le système d'exécution de Java est présent. Pour ce faire, le compilateur Java génère des instructions en bytecode qui n'ont de lien avec aucune architecture particulière. Au contraire, ces instructions ont été conçues pour être à la fois faciles à interpréter et faciles à traduire en code natif.

• **Portable**

A la différence du C/C++, on ne trouve pas les aspects de dépendance de la mise ne oeuvre dans la spécification. Les tailles des types de données primaires sont spécifiées, ainsi que le comportement arithmétique qui leur est applicable.

• **Interprété**

L'interpréteur Java peut exécuter les bytecode directement sur n'importe quelle machine sur laquelle il a été porté. Dans la mesure où la liaison est un processus plus incrémentiel et léger, le processus de développement peut se révéler plus rapide et exploratoire.

• **Performances élevées**

En général, les performances des bytecodes interprétés sont tout à fait suffisantes, il existe toutefois des situations dans lesquelles des performances plus élevées sont nécessaires. Les bytecodes peuvent être traduits à la volée en code machine pour l'unité centrale destinée à accueillir l'application.

• **Multithread**

Les avantages du multithread sont une meilleure interréactivité et un meilleur comportement en temps réel.

IV. 3. Les interfaces d'éditeur bayésien « Impact Bayes Net »

Les fenêtres des graphes sont les fenêtres encapsulant chacune un jeu de données indépendant. Les données incluses dans un graphe sont de plusieurs types :

- les données qualitatives : la structure du graphe (**Nœud** et **Arc**),
- les données quantitatives : **Table de probabilités** et **l'ensemble de données « Dataset».**

IV.3.1 La fenêtre de chargement d'éditeur

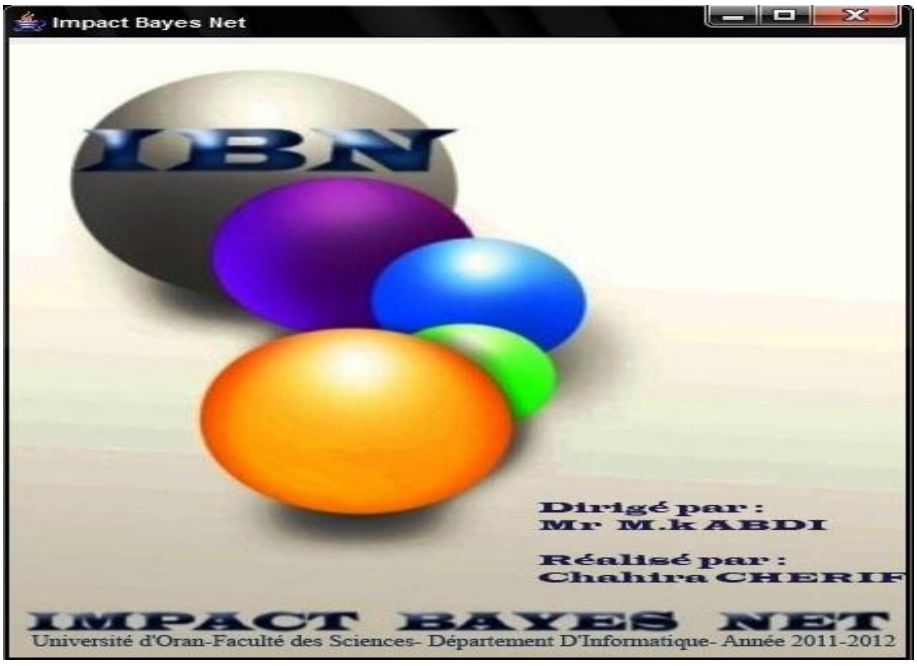


Fig. IV.1 Fenêtre de chargement d'éditeur

IV.3.2 La fenêtre d'éditeur du réseau bayésien

L'interface principale est l'environnement de travail de *IBN*. Elle se divise en 3 parties principales :

- la zone des commandes (**menus** et **barre d'outils**) qui intègrent l'ensemble des commandes pouvant intervenir soit sur tous les graphes, soit sur le graphe actif.
- la zone des graphes dans laquelle s'ouvrent des **fenêtres de graphes**.
- la console : la zone d'affichage des résultats obtenus.

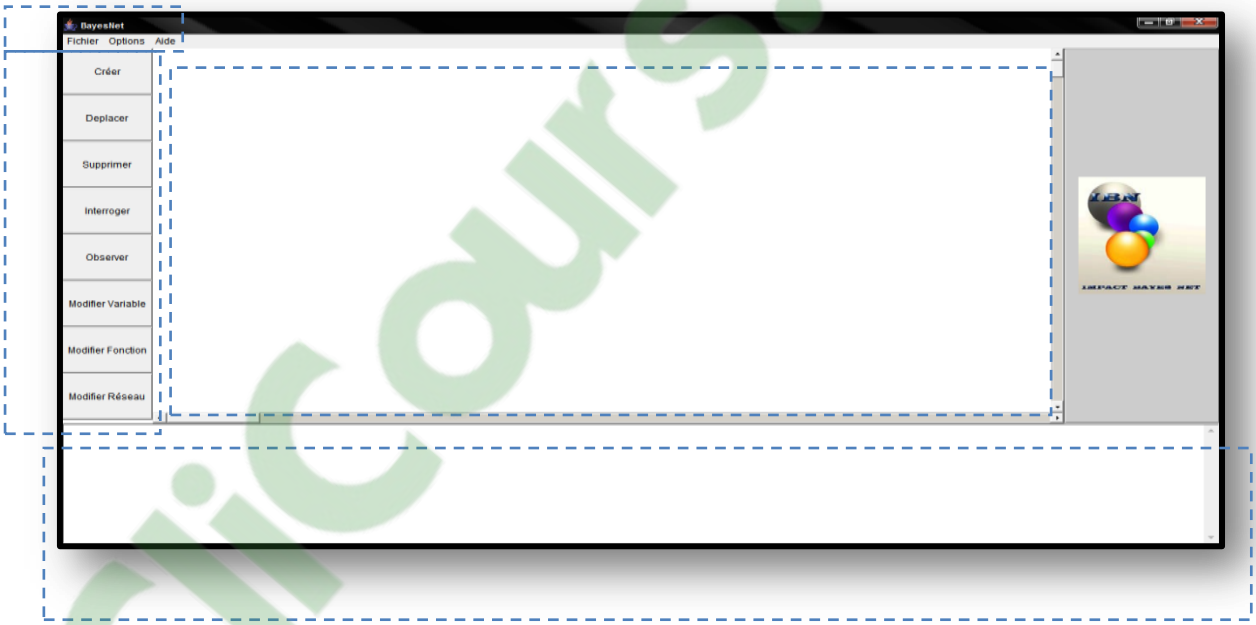


Fig. IV.2 La fenêtre d'éditeur du réseau bayésien

IV.3.3 La création du réseau bayésien

Les fenêtres des graphes sont les fenêtres encapsulant chacune un jeu de données indépendant. Les données incluses dans un graphe sont de plusieurs types :

- les données qualitatives : la structure du graphe (**Nœud** et **Arc**),
- les données quantitatives : **Table de probabilités** et **les données**.

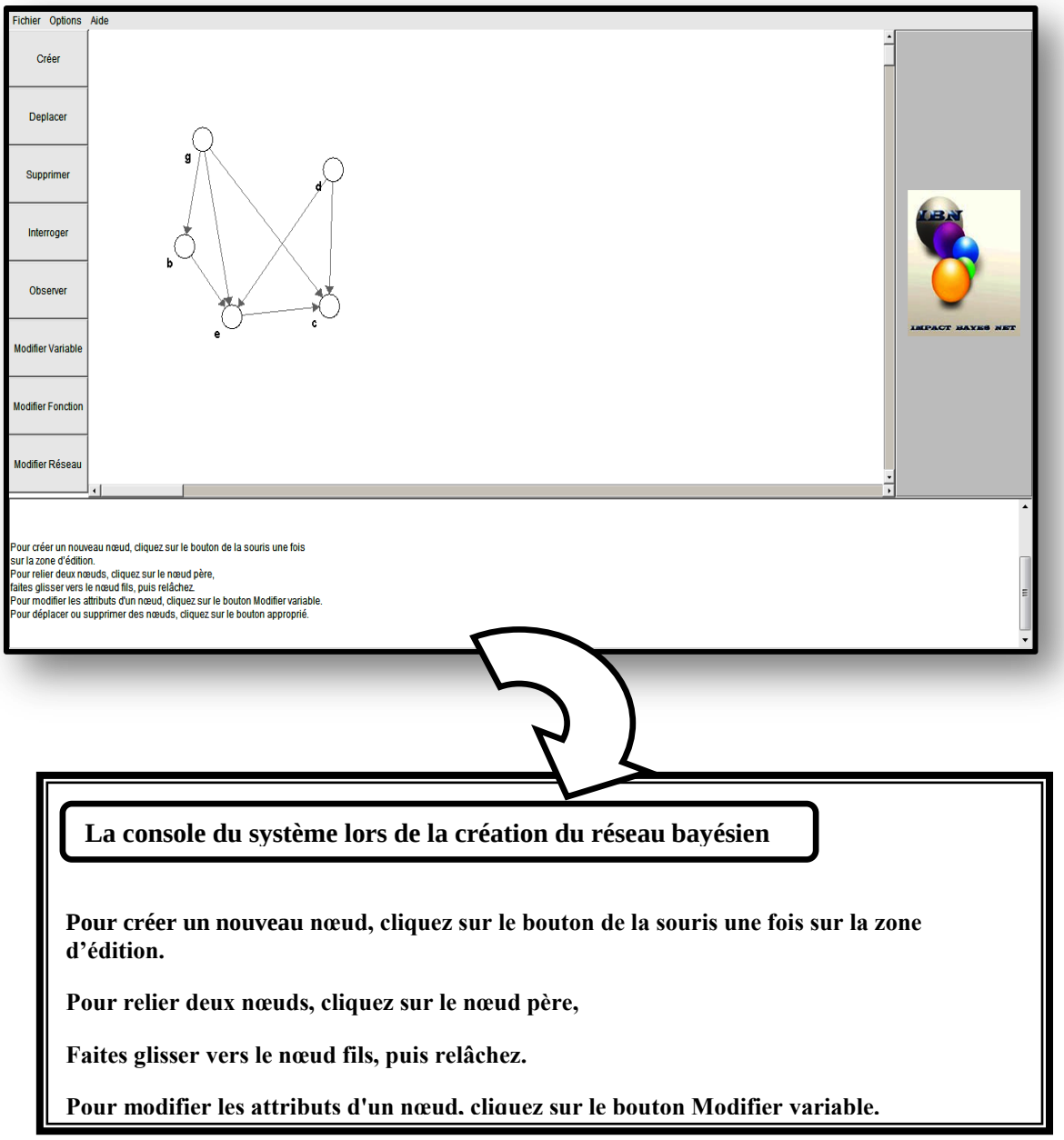


Fig. IV.3 La création du réseau bayésien

IV.3.4 La suppression d'un nœud/arc

Dans ce mode, cliquer sur le bouton supprimer et puis avec un seul clique sur un l'objet (arc ou nœud) le supprime.

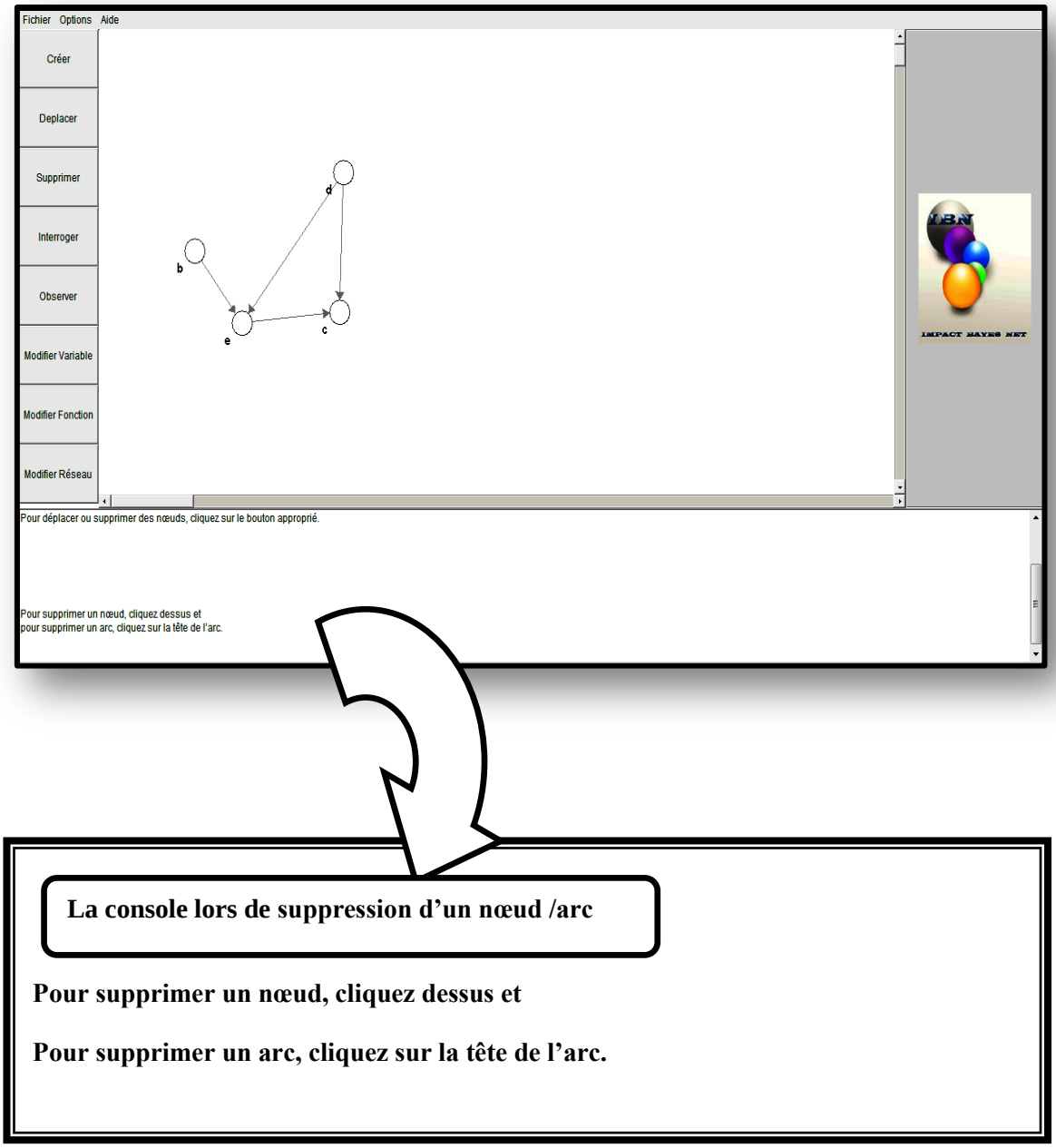


Fig. IV.4 La suppression d'un nœud/arc

IV.3.5 La modification de variable / fonction

Dans ce mode, cliquer sur le bouton « Modifier variable ou bien fonction » et puis sur le nœud pour modifier le nom/ la valeur du nœud sinon dans la modification de fonction ou vous pouvez changer les valeurs de la table de probabilité du nœud sélectionné.

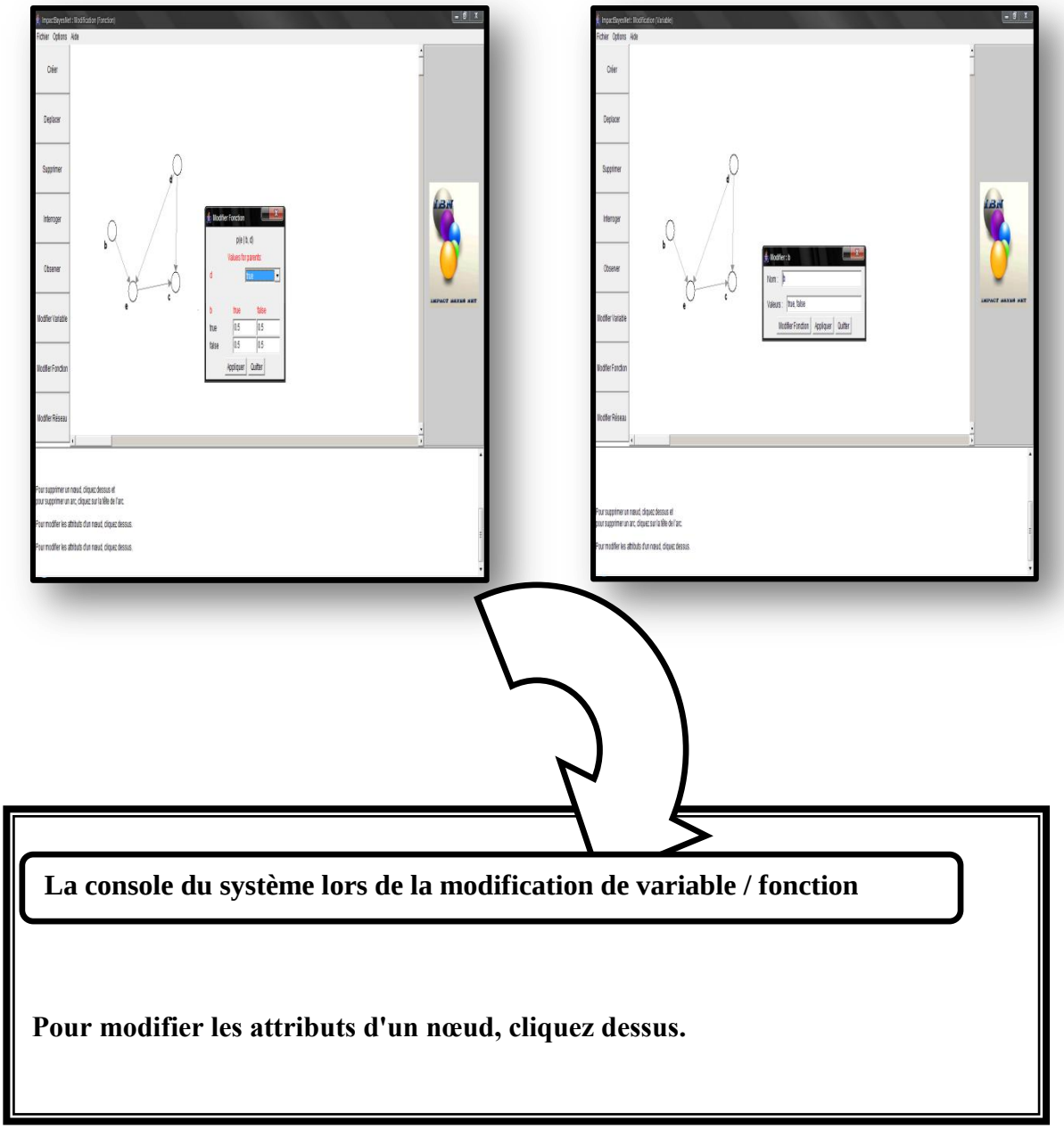


Fig. IV.5 Modification de variable / Fonction

IV.3.6 L'observation d'une variable « nœud»

Un tel nœud représente une variable chance pour laquelle l'utilisateur a saisi des probabilités fixées avec un choix d'une des valeurs du nœud.

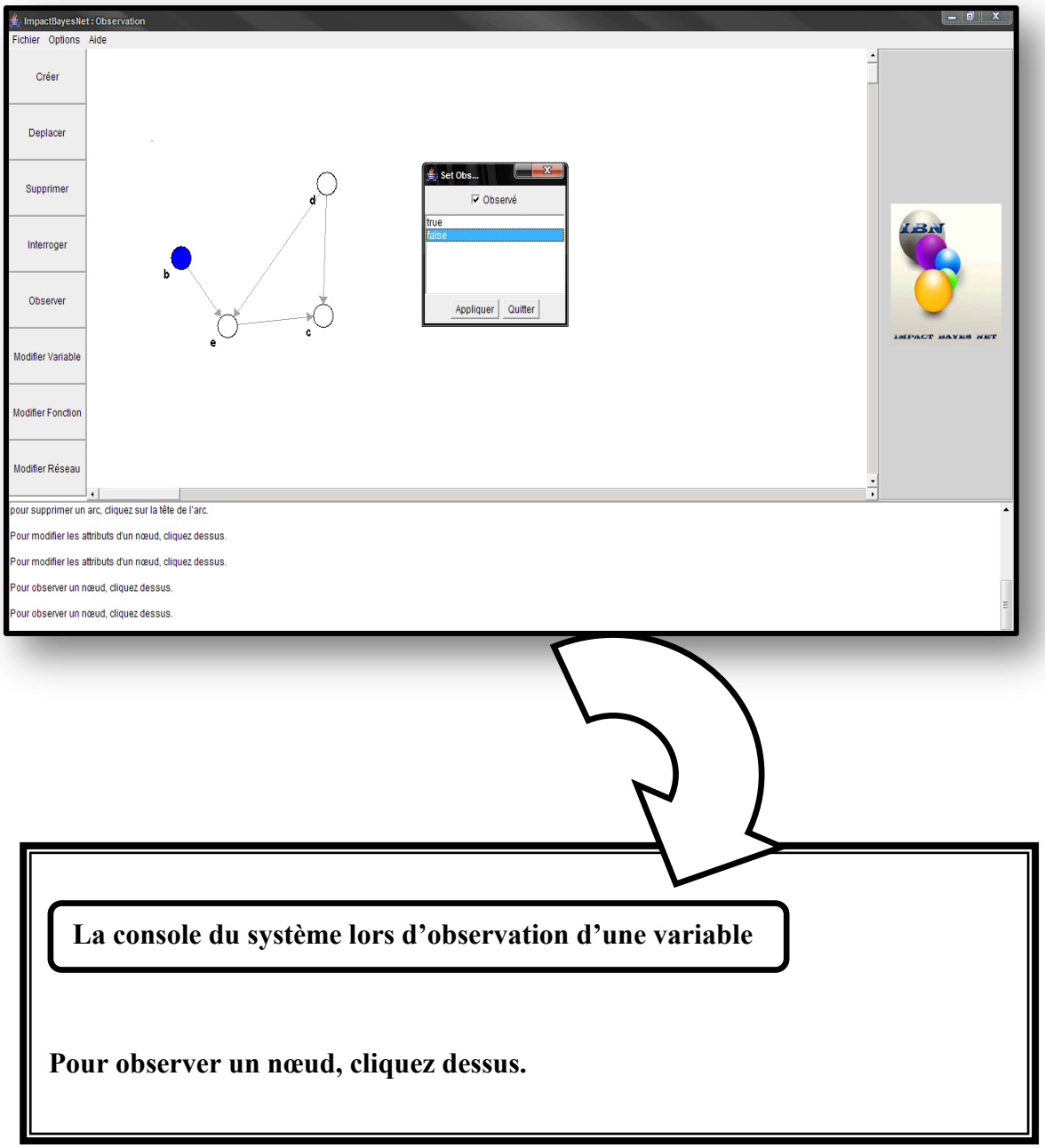
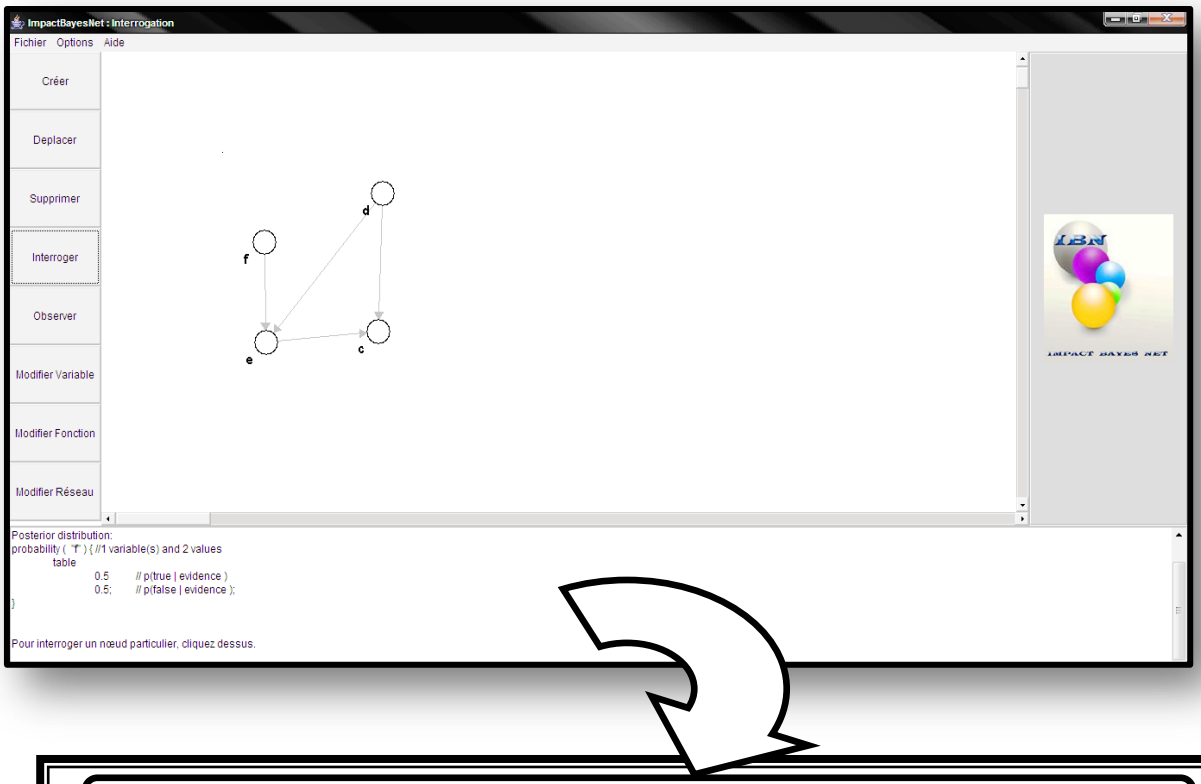


Fig. IV.6 Observation d'une variable « nœud»

IV.3.7 L'interrogation des nœuds du réseau bayésien

L'utilisateur affecte des valeurs à certaines variables dans un réseau et demande la probabilité a posteriori marginale. L'ensemble des variables qui ont attribué des valeurs est appelée la ***preuve.*** Probabilités marginales peuvent être calculés conditionnelle à un certain nombre d'observations insérées dans le réseau



La console du système lors de l'interrogation des nœuds du réseau

Pour interroger un nœud particulier, cliquez dessus.

Distribution à posteriori:

Probabilité ("f") { //1 variable(s) Et 2 valeurs

Table

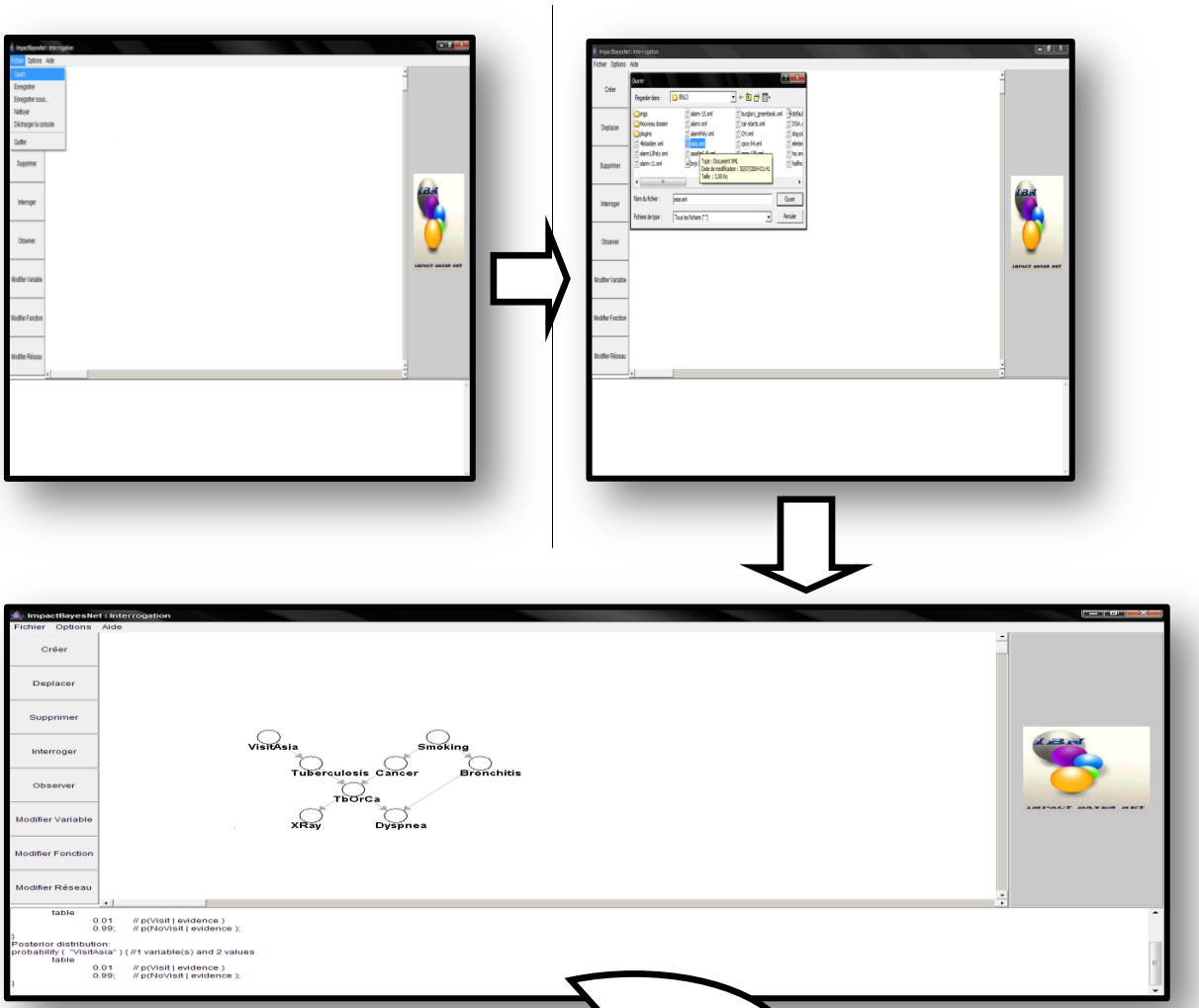
0.5 // p(oui)

0.5; // p(non); }

Fig. IV.7 Interrogation des nœuds du réseau bayésien

IV.3.8 Interrogation d'un fichier existant

Les interfaces suivantes montrent comment on peut ouvrir un fichier qui existe déjà et de calculer l'inférence du réseau.



La console du système lors l'ouverture et l'interrogation un fichier existant

interroger un nœud particulier, cliquez dessus.

Distribution à posteriori:

Probabilité ("Tuberculoses") {//1 variable(s) et 2 valeurs

Table

0.010 // p(Present)

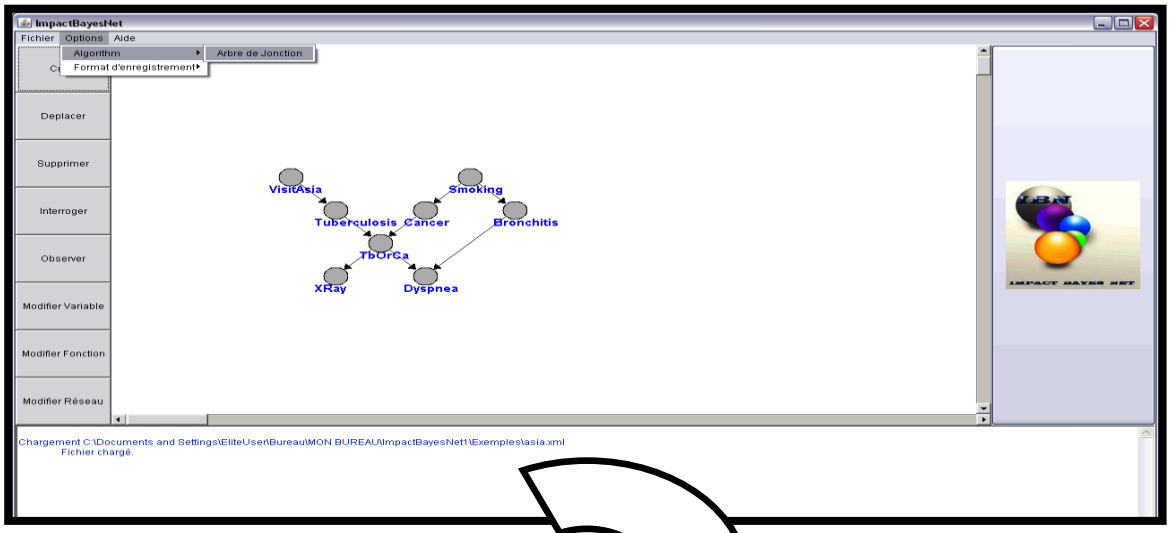
0.989; // p(Absent); }

Fig. IV.8 Interrogation d'un fichier existant

IV.3.9 Options

IV.3.9 .1 Algorithme d'inférence

L'icone options contient l'algorithme d'inférence « Arbre de jonction» pour le sélectionner.



Cette option permet de sélectionner l'algorithme d'inférence :

- **Algorithme d'arbre de jonction**

Fig. IV.9 Algorithme d'inférence

IV.3.9.2 Format d'enregistrement

Les données lorsque vous utilisez impact bayes net on peut charger les données et les enregistrer localement**.** Et l'éditeur prend en charges les deux formats différents existants et nous permet de lire le contenu écrit sur les fichiers.

a) Le format BIF

La version « Bayesian Interchange Format» est un format simple, qui a été utilisée avec succès pour représenter une variété de réseaux. Le BIF réfère à trois entités: les mots, les nombres

entiers non négatifs et réels non négatifs. Un *mot* est une séquence de caractères contigus, avec la restriction que le premier caractère est une lettre. Caractères sont des lettres ainsi que « des numéros + le symbole de soulignement (_), plus le tiret (-) ». Un *nombre non négatif* est une séquence de caractères numériques, contenant une virgule ou un exposant, ou les deux.

L'unité de base d'information est un *bloc:* un morceau de texte qui commence par ce mot et se termine avec la fin d'une liste d'attributs. Les caractères arbitraires sont permis entre les blocs. Cela permet à l'utilisateur d'insérer des commentaires de longueur arbitraire en dehors des blocs.

b) Le format XML

Le but du format actuel XMLBIF est de représenter des graphes acycliques orientés qui peuvent être associées à des mesures de probabilité conditionnelle des variables discrètes, avec la possibilité que les variables de décision et l'utilité être présents dans le graphique.

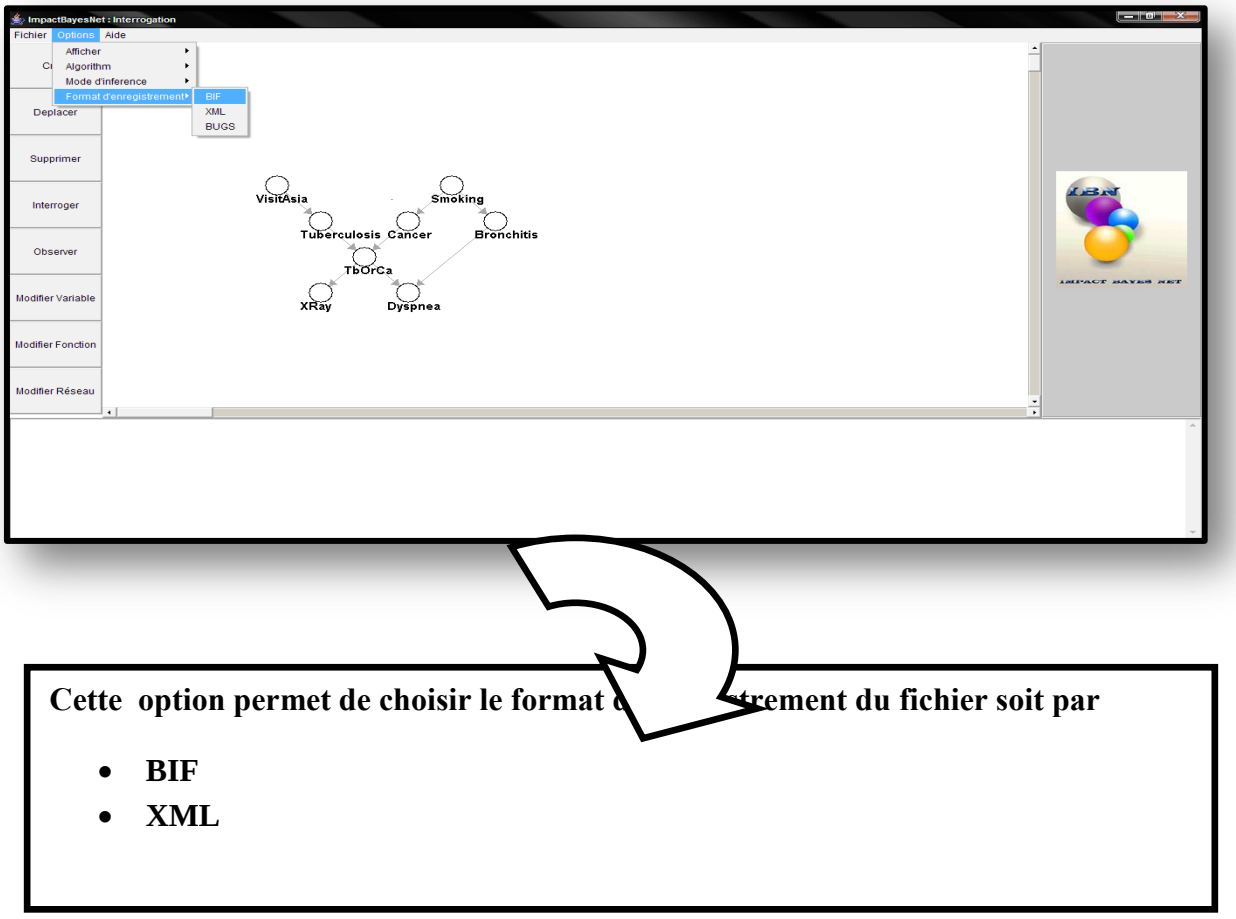


Fig. IV.10 Format d'enregistrement

IV.4 La validation du système

Pour valider notre outil, nous avons choisis un exemple « Asia », c'est un Réseau bayésien composé de 8 nœuds, et on va comparer les résultats d'inférence par l'éditeur BNJ avec notre outil *IBN* après la mise en œuvre du même réseau bayisien.

IV.4.1 Présentation du problème

Tout au long de sa carrière, le spécialiste des maladies des poumons a créé une base de données contenant les informations sur les patients qu'il a consultés afin de laisser une trace à son successeur. Une ligne de la base de données correspond au diagnostic d'un patient. Voici les deux premières lignes de la base de données :

Le tableau V.4.1 suivant illustre l'inférence de l'exemple « ASIA» sous BNJ

Fumeur	Cancer	Tuberculose	TbOuCa	VisiteAsie	Radiographie	Bronchite	DifficultéRespiratoire
Oui	Non	Non	Non	Non	Normal	Non	Oui

Tab. IV.4.1 Informations de l'exemple « ASIA»

La première ligne correspond aux noms des variables et la deuxième correspond aux informations d'un patient. Voici le détail des variables :

N° : numéro d'identification du patient. (Un entier)

Fumeur : Est il fumeur ? Oui/Non

Cancer : A-t-il un cancer ? Oui/Non

Tuberculose : A-t-il la tuberculose ? Oui/Non

TbOuCa : il s'agit d'une variable créée pour simplifier les tables de probabilités.

VisiteAsie : Le patient a-t-il visité l'Asie ? Oui/Non

Radiographie : Comment est la radio ? Normale/Anormale

Bronchite : A-t-il une bronchite ? Oui/Non

Difficulté Respiratoire : A-t-il des difficultés respiratoires ? Oui/Non

Le spécialiste qui le remplace veut déterminer toutes les associations entre ces variables afin de prédire les risques pour un patient d'avoir un cancer ou la tuberculose et surtout pouvoir juger s'il est nécessaire d'effectuer une radiographie.

IV.4.2 Les tables de probabilités des nœuds du réseau bayésien «ASIA »

L'interface suivante illustre les valeurs de probabilités dans les TPs de chaque nœud du réseau « ASIA».

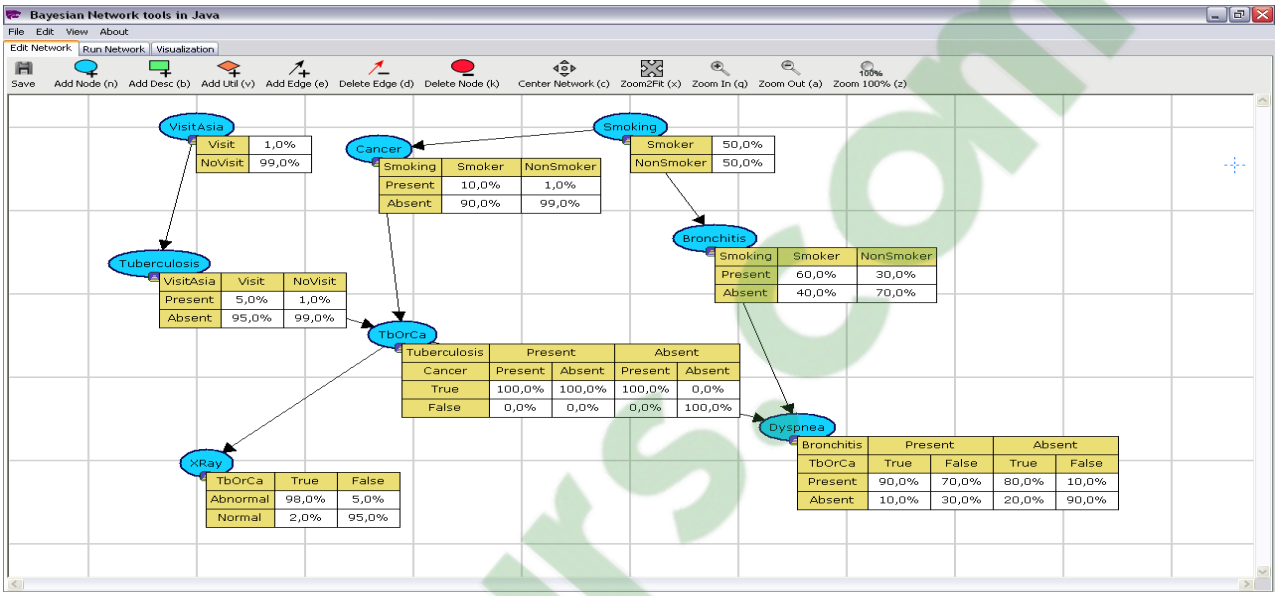


Fig. IV.11 TPs du réseau bayésien «ASIA »

IV.4.3 Les résultats de l'inférence sous BNJ

Suivant les valeurs des tables de probabilités (Fig. IV.4.2), et après la mise en œuvre du BNJ on a obtenu les résultats qui sont affichées dans l'interface ci-dessous :

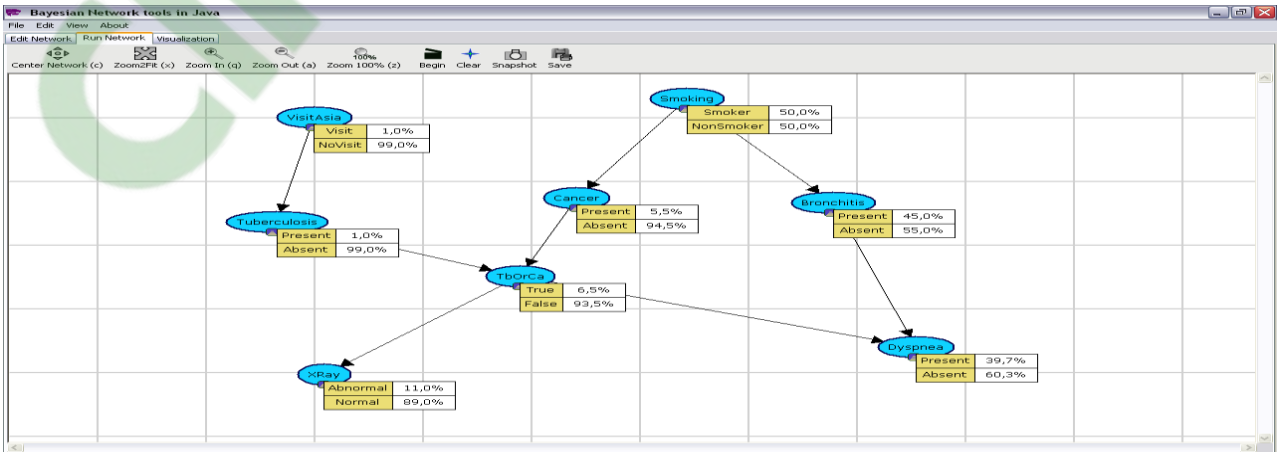
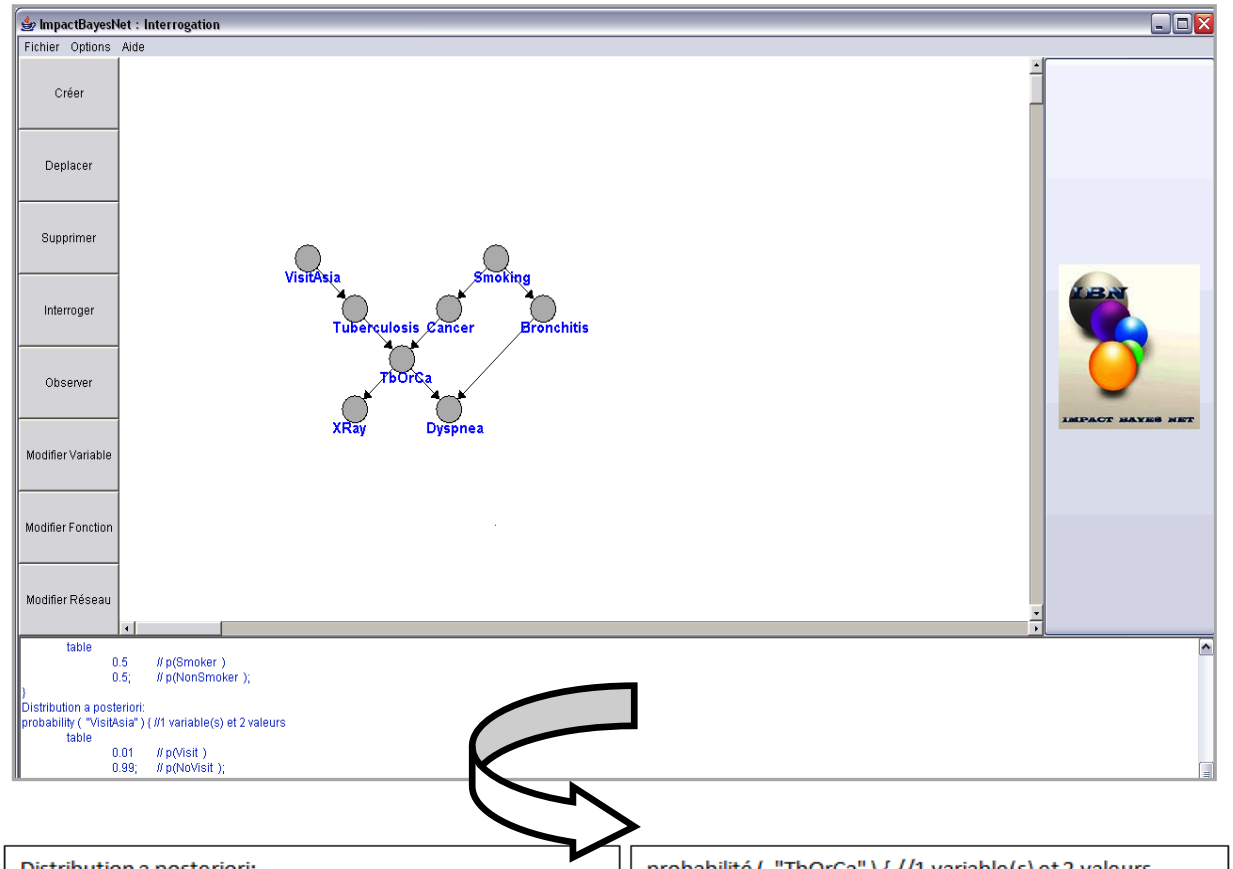


Fig. IV.12 Résultats de l'inférence sous BNJ

IV.4.4 Les résultats du réseau bayésien « ASIA »



Distribution a posteriori:

probabilité ("XRay") { //1 variable(s) et 2 valeurs

table

0.110 // p(Abnormal)

0.889; // p(Normal);

probabilité ("Dyspnea") { //1 variable(s) et 2 valeurs

table

0.397 // p(Present)

0.602; // p(Absent);

probabilité ("Bronchitis") { //1 variable(s) et 2 valeurs

table

0.449 // p(Present)

0.55; // p(Absent);

probabilité ("Cancer") { //1 variable(s) et 2 valeurs

table

0.055 // p(Present)

0.945; // p(Absent);

probabilité ("TbOrCa") { //1 variable(s) et 2 valeurs

table

0.064 // p(True)

0.935; // p(False);

probabilité ("Tuberculosis") { //1 variable(s) et 2 valeurs

table

0.010 // p(Present)

0.989; // p(Absent);

probabilité ("Smoking") { //1 variable(s) et 2 valeurs

table

0.5 // p(Smoker)

0.5; // p(NonSmoker);

probabilité ("VisitAsia") { //1 variable(s) et 2 valeurs

table

0.01 // p(Visit)

0.99; // p(NoVisit); }

Fig. IV.13 Résultats du réseau bayésien « ASIA » sous IBN

IV.4.5 Tableau comparatif des résultats

Le tableau suivant contient les résultats du réseau «ASIA » sous BNJ et IBN

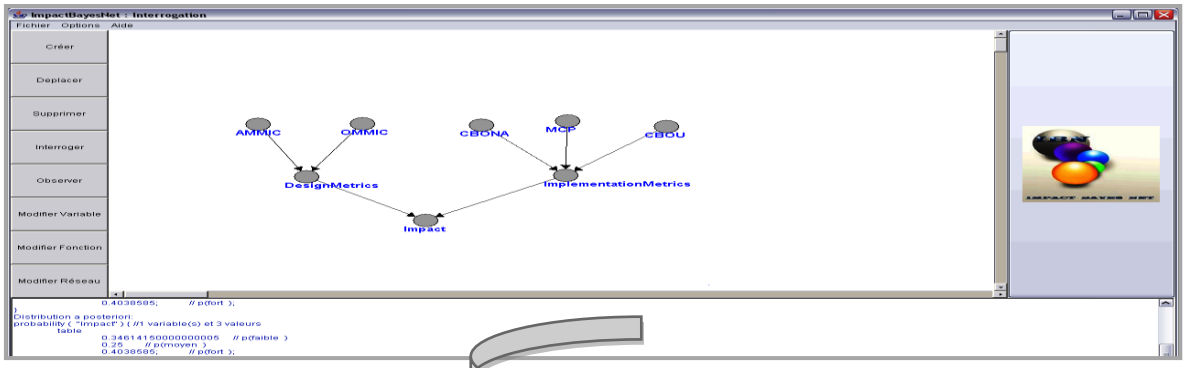
Nœuds	Fumeur	Cancer	Tuberculose	TbOuCa	VisiteAsie	Radiographie	Bronchite	Difficulté Respiratoire
Inférence sous Bnj	Oui :50% Non :50%	Présent : 5.5% Absent :94.5%	Présent :1% Absent :99%	Oui :6.5% Non :93.5	Oui : 1% Non: 99%	Anormal :11% Normal :89%	Présent :45% Absent :55%	Present :39.7% Absent :60.3%
Inférence sous IBN	Oui :0.5 Non :0.5	Présent : 0.055 Absent : 0.945	Present:0.0104 Absent: 0.9896	Oui : 0.064 Non: 0.935	Oui : 0.01 Non:0.99	Anormal : 0.110 Normal: 0.889	Présent : 0.4499 Absent: 0.55	Présent : 0.3974 Absent: 0.6025

Tab. IV.1 Résultats comparatifs entre BNJ et IBN

IV.5 Résultats et expérimentations

Une fois la structure du graphe et toutes les TPS définies, nous pouvons procéder à l'inférence bayésienne. Il en résulte une mise à jour de toutes les probabilités conditionnelles de tous les sommets.

Après la mise en œuvre du réseau impact en utilisant les tables de probabilité définies dans Abdi et al. [3], nous avons exécuté l'algorithme d'inférence d'approche exacte arbre de jonction sous l'éditeur bayésien IBN « Impact Bayes Net » .



Probabilité («IMPACT») {//1 variable(s) et 3 valeurs

Table

0.346 **// p (faible)**

0.25 **// p (moyen)**

0.403; **// p (fort);**

Fig.IV.14 Réseau d'impact du changement après inférence (Scénario 1)

Ayant affecté trois états « Faible », « Moyen » et « Fort » au sommet Impact, et avec les données d'entrées utilisées dans les quatre scénarios illustré dans le tableau (Tab V.5.5.2) suivant :

Scénarios	Scénario 1		Scénario 2		Scénario 3		Scénario 4	
Nœuds d'entrée	Petit	Grand	Petit	Grand	Petit	Grand	Petit	Grand
AMMIC	0.35	0.65	0.5	0.5	0.05	0.95	0.05	0.95
OMMIC	0.6	0.4	0.5	0.5	0.5	0.5	0.5	0.5
CBONA	0.75	0.25	0.9	0.1	0.05	0.95	0.9	0.1
CBOU	0.45	0.55	0.9	0.1	0.05	0.95	0.9	0.1
MPC	0.2	0.8	0.5	0.5	0.5	0.5	0.5	0.5

Tab. IV.2 Tables de probabilités des nœuds d'entrée

L'exécution du réseau sur ce jeu de données a généré ces estimations d'impact (valeurs de probabilités sur le sommet Impact). Nous pouvons conclure par exemple que l'impact de changement a 40.3% de chances d'être « Fort ».

La figure IV.5.1 montre qu'en diminuant les valeurs des métriques CBONA et CBOU, l'impact de changement s'affaiblit de plus en plus (sa probabilité d'être «faible» passe de 34.6 % à 37.8 %). Dans le scénario2 (Tab. IV.5.2) en augmentant cette fois ci les valeurs des métriques CBONA et CBOU, l'impact de changement devient de plus en plus fort. La probabilité de l'état « Fort » passe de 37,1% à 44,4% avec un gain de plus que 7 points (Tab. IV.5.3**).**

Enfin, le dernier scénario exécuté montre qu'en diminuant valeurs des métriques CBONA et CBOU et en augmentant la valeur de AMMIC, l'impact du changement devient un peu plus fort. La probabilité de l'état « Fort » passe de 37,8% à 41,5%.

Nous avons fait appel à l'environnement *BNJ* (Bayesian Network tools in Java) pour faire une comparaison avec les résultats obtenus sous *IBN*. Cet outil utilise l'algorithme d'arbre de jonction pour l'inférence bayésienne.

SCENARIOS	IMPACT		
	FAIBLE	MOYEN	FORT
SCENARIO1 BNJ	34.6 %	25.0 %	40.4%
SCENARIO1 IBN	0.346	0.25	0.403
SCENARIO 2 BNJ	37.8 %	25.0%	37.2%
SCENARIO 2 IBN	0.378	0.25	0.371
SCENARIO 3 BNJ	30.6 %	25.0%	44.4%

SCENARIO 3 IBN	0.3058	0.25	0.444
SCENARIO 4 BNJ	34.0 %	25.0%	41.0%
SCENARIO 4 IBN	0.339	0.25	0.410

Tab. IV.3 Résultats de l'impact après l'inférence « Les quatre scénarios »

IV.6 Conclusion

Impact Bayes Net « IBN» est un outil d'aide à la création et à l'utilisation de réseaux bayésiens.

Il a pour but de permettre la saisie, la modification, l'utilisation d'inférence de modèles à base de réseaux bayésiens. Nous avons présenté dans ce chapitre :

- les différentes étapes de la mise en œuvre de notre système développé Impact Bayes Net avec une explication de chaque interface qui illustre le fonctionnement du système.
- la mise en œuvre de notre approche utilisant un réseau bayésien d'impact [3], composé d'un ensemble des nœuds d'entrée qui sont des métriques de conception et d'implémentation et qui sont étudiées afin de comprendre leurs effets sur les systèmes.

Les résultats d'inférence sous BNJ vs IBN pour valider l'éditeur utilisé pour la mise en œuvre du réseau impact. A travers la présente étude, nous essayons d'évaluer ces avantages dans le cadre de l'analyse et la prédiction de l'impact du changement dans un système à objets. Nous cherchons à donner plus d'explications sur les facteurs réels et responsables de cet impact du changement ainsi que de son évolution.

Enfin, nous pensons que l'exploitation des capacités d'apprentissage automatique qu'offrent les réseaux bayésiens nous permettra d'avoir dans une perspective à court terme une meilleure précision de la prédiction et ainsi des résultats plus convaincants.

Conclusion Générale

Conclusion générale

La volonté de réduire le coût de maintenance des logiciels motive les chercheurs et les industriels. La tâche de maintenance peut se traduire par une succession de changements que les mainteneurs effectuent dans un but correctif ou perfectif. Ainsi, la clé pour maîtriser le coût de maintenance consiste à bien mener ces changements. Dans cette perspective, plusieurs approches sont proposées. Elles utilisent différents artefacts logiciels et dont l'efficacité et la possibilité de mise en place varient considérablement.

Ce mémoire s'intéresse à un volet de la maintenance logicielle qui consiste à l'analyse de l'impact des changements. Nous proposons une approche probabiliste qui prédit les classes affectées suite à un changement. La mise en œuvre de notre approche est assurée par des réseaux bayésiens.

Nos prédictions se basent sur à la compréhension des facteurs réels qui sont responsables de l'impact de changement et de son évolution. Des métriques de conception et d'implémentation sont étudiées afin de comprendre leurs effets sur les systèmes [3]. Nous avons choisi l'approche probabiliste en utilisant les réseaux bayésiens pour déterminer l'impact des changements.

Les réseaux bayésien sont polyvalents on peut se servir du même modèle pour évaluer, prévoir, diagnostiquer, ou optimiser des décisions, ce qui contribue à rentabiliser l'effort de construction du réseau bayésien mais il y a pas mal d'inconvénients surtout la complexité des algorithmes.

Par ailleurs, un des principaux avantages liés à l'utilisation des réseaux bayésiens est l'inférence par l'algorithme de l'arbre de jonction implémenté qui fait partie de l'approche exacte d'inférence bayésienne, afin de l'intégrer à notre approche en espérant améliorer l'efficacité de celle-ci.

Bien que des résultats intéressants soient obtenus lors de notre évaluation, le travail présenté dans ce mémoire peut être enrichi de différentes manières. Nous envisageons deux perspectives majeures, à savoir :

- Offrir d'autres possibilités d'inférence en implémentant d'autres algorithmes d'inférence.
- Ajouter un module d'apprentissage au système développé.

Bibliographie

Bibliographie

[1] R.L. Baber ; “Software Engineer’s Reference Book Epilogue: future developments”, in Software Engineer’s Reference Book, Butterworth-heinemann, ISBN 0-7506-1040-9, 1991, Chapter 63, pp. 1-15.

[2] S.L. Pfleeger. Software Engineering—Theory and Practice. Prentice Hall, 2nd ed. 2001.

[3] Analyse et prédiction de l’impact de changements dans un système à objets : Approche probabiliste, M.K Abdi, H. Lounis, H. Sahraoui, 2009

[4] Di Gallo Frédéric, « cours de Génie logiciel »,*Cycle probatoire*, cnam bordeaux, 2001.

[5] G.E. Stark, L. C. Kern, and C. V. Vowell. A Software Metric Set for Program Maintenance Management. Journal of Systems and Software, Vol. 24, no. 3, March 1994.

[6] A. April et D. Al-Shurougi, Software Product Measurement for Supplier Evaluation, FESMA 2000, Madrid, October 18-20, 2000.

[7] R. Moreton, A Process Model for Software Maintenance. Journal Information Technology,Volume 5, 1990, Pages100-104.

[8] S. A. Bohner, Impact Analysis in Software Change Process : A year 2000 perspectives. In International Conference on Software Maintenance, Pages 42-51, 1996

[9] L.C. Briand, Jurgen Wust, Hakim Lounis, Using Coupling Mesurement for Impact Analysis in Object Oriented Systems. IEEE International Conference on Software Maintenance (ICSM), 1999.

[10] Thèse de doctorat, De l’identification de structure de réseaux bayésiens à la reconnaissance de formes à partir d’informations complètes ou incomplètes, François Olivier, 2006

[11] Thèse de doctorat, Une approche pro-pédagogique du diagnostic cognitif dans les sti : conception, formalisation et implémentation, Joséphine Muriel pelle Tchetagni,2005

[12] Edition Eyrolles, "Réseaux bayésiens", Naïm, P., P. Wuillemin, P. Leray, O. Pourret, et A. Becker 2004.

[13] T. Verma et J. Pearl. Causal networks : Semantics and expressiveness.In Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence, pages 352–359.1988.

[14] Judea Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[15] Fung R. and Chang K.C. Weighting and integrating evidence for stochastic simulation in Bayesian Network. In Proc. of UAI, vol. 5, page 209-219, New York Elsevier science publishing company, Inc 1989.

[16] Fung R. and Del Favero B. Backward simulation in Bayesian network. In R. Lopez de Mantaras and D. Poole Proc. of UAI,, page 227-234, July 1994.

[17] Shachter R. D. and Peot M. A. Simulation approaches to general probabilistic inference on belief networks. In Proc. of UAI, vol. 5, pages 311-318, 1989.

[18] Mémoire PFE, Réseaux Bayésiens, Ecole d'ingénieurs de Genève, CANEL Christophen, 2004

[19] Aymen Zoghlami ,Mémoire (M.Sc.), Approche probabiliste pour l'analyse de l'impact des changements dans les programmes orientés objet, 2011.

[20] Robert Fung et Del B. Favero. « Backward Simulation in Bayesian Networks ». Proceedings of the Conference on Uncertainty in Artificial Intelligence, pages 227–234, San Francisco, CA, USA, july 1994.

[21] Kim, J. & Pearl, J.. A computational model for combined causal and diagnostic reasoning in inference systems. Dans Proceedings IJCAI-83, (pp. 190–193)., Karlsruhe, Germany. 1983

[22] Fusion, propagation, and structuring in belief networks. Journal of Artificial Intelligence, 29, 241–288. 1986

[23] 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

[24] Bayesian updating in causal probabilistic networks by local computations. Computational Statistics Quaterly, 4, 269–282. 1990

[25] A simple approach to bayesian network computations. Dans In Proceedings of the tenth Canadian Conference on Artificial Intelligence, (pp. 171–178). 1994.

[26] Bucket elimination: a unifying framework for structure-driven inference Technical report, Dept. of Computer and Information Science, University of California, Irvine, USA.1998.

[27] Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society 1988

[28] An aco algorithm for the most probable explanation problem. Dans Australian Conference on Artificial Intelligence, (pp. 778–790). 2004

[29] Symbolic probabilistic inference in belief networks. Dans Proceedings of the Eighth National Conference on Artificial Intelligence, 1990.

[30] Efficient inference in bayes nets as a combinatorial optimization problem. International Journal of Approximate Reasoning, 11(1), 55–81. 1994.

[31] A new method for symbolic inference in bayesian networks. Networks, 28, 31–43. 1996.

[32] A differential approach to inference in bayesian networks. Dans Proceedings of Uncertainty In Artificial Intelligence, (pp. 123–132). 2000.

[33] A randomized approximation algorithm for probabilistic inference on bayesian belief networks. Networks, 20(5), 661–685. 1990.

[34] Markov Chain Monte Carlo in Practice. Interdisciplinary Statistics. Chapman & Hall. 1996.

[35] Monte Carlo Statistical Methods. Springer texts in statistics 2004.

[36] Graphical models, exponential families, and variational inference. Technical report, Departement of Statistics, University of California, Berkeley. 2003.

[37] An introduction to algorithms for inference in belief nets. Dans Proceedings of the 5th Annual Conference on Uncertainty in Artificial Intelligence (UAI-90), New York, NY. Elsevier Science Publishing Company, Inc. 1990

[38] Average-case analysis of a search algorithm for estimating prior and posterior probabilities in bayesian networks with extreme probabilities. Dans Proceeding of the thirteenth International Joint Conference on Artificial Intelligence, (pp. 606–612). 1993.

[39] Bayesian networks : a model of self-activated memory for evidential reasoning.Technical Report 850021 (R-43), UCLA Computer Science Department Technical Report, and in Cognitive Science Society, UC Irvine, 329-334. 1985.

[40] Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference. Morgan Kaufmann, second edition in 1991.

[41] Loopy belief propagation for approximate inference: An empirical study. Dans Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99), (pp. 467, 475), San Francisco, CA. Morgan Kaufmann Publishers. 1999.

[42] Approximation of Bayesian networks through edge removals. Research Report IR-93-2007, The Machine Intelligence Group, Aalborg University. 1993.

[43] A survey of algorithms for real-time bayesian network inference. 2001.

[44] SMILE, Structural Modeling, Inference, and Learning Engine. Application Programmer's Manual, version 1.1. Decision Systems Laboratory, University of Pittsburgh, School of Information Sciences, 2003.

[45] Jun Han, Supporting Impact Analysis and Change Propagation in Software Engineering Environments. Peninsula School of Computing and Information Technology. Monash University, McMahons Road, Frankston, Vic. 3199, Australia. Octobre 1996.

[46] D. C. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, and C. Chen, Change Impact Identification in Object-Oriented Software Maintenance. Proceedings of the Conference on Software Maintenance, 1994.

[47] M. A. Chaumun, H. Kabaili, R. K. Keller and F. Lustman, A Change Impact Model for Changeability Assessment in Object-Oriented Software Systems. In Proceedings of the Third European Working Conference on Software Maintenance and Reengineering, Pages130-138, Amsterdam, The Netherlands, March 1999.

[48] H. Kabaili, Rudolf K. Keller, and Francois Lustman, Predicting the Changeability of Object Oriented Software with the Design Metrics. Submitted to Journal of Software and Systems, 2002.

[49] Li Li and A. Jefferson Offutt, Algorithmic Analysis of the Impact of Changes to Object-Oriented Software. IEEE International Conference on Software Maintenance, Pages 171-184.1996.

[50] M. Lindvall, Measurement of Change: stable and Change-prone constructs in a commercial C++ System. In Proceedings of the 6th International Software Metrics Symposium, Pages 40-49, Nov.1999.

[51] G.Antoniol, G,Canfora and A. de Lucia, Estimating the Size of Changes for Evolving Object Oriented Systems: a case study". In proceedings of the 6th International Software Metrics Symposium, Pages 250-258, boca Raton florida, Nov. 1999.

[52] S. L. Pfleeger and Shawn A. Bohner, A Framework for Software Maintenance Metrics. In proceedings of the Conference on Software Engineering, Pages 320-327. May 1990.

[53] R. S. Arnold and S. A. Bohner, Impact Analysis - Towards A Framework for Comparison. Proceedings of the Conference on Software Maintenance, Pages 292-301. September 1993.

[54] L.J. Arthur. Software Evolution: The Software Maintenance Challenge. John Wiley & Sons, 1988.

[55] N. H. Madhavji, Environment Evolution: The Prism Model of Changes. IEEE Transaction on Software Engineering, Vol. 18, No. 5, Pages 380-392. May 1992.

[56] Laila Cheikhi , Thèse master, Estimation de l'impact du changement dans les programmes à Objets,Université de Montréal, 2004.

[57] Johnson, W. E. Logic. Cambridge University Press. 1921.

[58] Keynes, J. M.A treatise on probability. Macmillan and Co. 1

[59] Venn, J.The logic of chance. 2nd ed., Macmillan and co, reprinted, New York. 1876.

[60] Ramsey, F. P. Truth and probability. in Foundations of Mathematics and other Essays, R.B. Braithwaite (ed.), Routledge & P. Kegan , 1931, 156-198 ; reprinted in Studies in Subjective Probability, H. E. Kyburg, Jr. and H. E. Smokler (eds.), 2nd ed., R. E. Krieger Publishing Company, 1980, 23-52 ; reprinted in Philosophical Papers, D. H. Mellor (ed.) Cambridge University Press, 1990.

[61] Shachter, R.. Bayes-ball : The rational pastime (for determining irrelevance an requisite information in belief networks and influence diagrams). Dans Cooper, G. & Moral, S. (Eds.), Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, (pp. 480– 487), San Francisco. Morgan Kaufmann. 1998.

[62] Alikacem, E. H. et H. Snoussi BOAP 1.1.0. Manuel d'utilisation, CRIM. 2002.

[63] B.W. Boehm; “Software Engineering”, in IEEE Transactions on Computers pp. 1226-1241 , December 1976

[64] B.Lientz, E.B. Swanson et G.E. Tompkins, Characteristics of Applications Software Maintenance Comm. ACM, Vol. 21, 1978.

[65] © IEEE – Version Française de “IEEE STD 1219: Standard for Software Maintenance, 1998.” April– Septembre 2005

[66] © IEEE – Version Française de “ISO/IEC 12207: Information Technology- Software Life Cycle Processes, 1995”, April– Septembre 2005.

[67] © IEEE – Version Française de “ISO/IEC 14764: Software Engineering-Software Maintenance, 2000”, April– Septembre 2005.

[68] R.B. Grady and D. L. Caswell. Software Metrics: Establishing a Company-wide Program. Prentice-Hall, 1987.

[69] H.D. Rombach, “Design Measurement: Some Lessons Learned”, in proceedings of IEEE Software, Vol. 7, No. 2, pages 17-25, 1990.

[70] Reinhard Schauer, Rudolf K. Keller, Bruno Lague, Gregory Knapen,Sebastien Robitaille, and Guy Saint-Denis. The SPOOL DesignRepository: Architecture, Schema, and Mechanisms. In Hakan Erdogmus and Oryal Tanir, editors, Advances in Software Engineering. Topics inEvolution, Comprehension, and Evaluation. Springer-Verlag, 2001.

[71] Mustapha Kamel ABDI, Analyse et Prédiction d’impact de changement dans système à objets, Thèse de doctorat d’état en Informatique, Université Es-Sénia d’Oran, Avril 2007.

[72] Laplace, P. S. (1814). A philosophical essay on probabilities. English edition, New York,Dover Publications Inc. 1951.

[73] A. Dempster, N. Laird, et D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, B 39 :1–38, 1977.

[74] A. Tang, A. Nicholson, Y. Jin, and J. Han. Using bayesian belief networks for change impact analysis in architecture design. J. Syst. Softw, 80: pages 127-148, January 2007

[75] S. Mirarab, A. Hassouna, and L. Tahvildari. Using bayesian belief networks to predict change propagation in software systems. Proceedings of the 15th IEEE International Conference on Program Comprehension, pages 177-188, 2007.

[76] Y. Zhou, M. Würsch, E. Giger, H. C. Gall, and J. Lü. A bayesian network based approach for change coupling prediction. In Proceedings of the 2008 15th Working Conference on Reverse Engineering, pages 27-36, Washington, DC, USA, 2008. IEEE Computer Society.

Annexe A

Table A. *Résultats de l'impact des changements (Java)* [71]

Id Changement	Description de Changement	Expression d'impact
v.1.1	Changement de valeur de variable	-
v.1.2	Changement de type de variable	S+L
v.1.3	Ajout de variable	-
v.1.4	Suppression de variable	S+L
v.1.5	Changement de portée de variable	
v.1.5.1	Public → Private	S
v.1.5.2	Public → Protected	S~H *
v.1.5.3	Protected → Private	SH
v.1.5.4	Protected → Public	-
v.1.5.5	Private → Public	-
v.1.5.6	Private → Protected	-
v.1.6	Changement de variable (Static/Non-static)	
v.1.6.1	Static → Non-static	S+L
v.1.6.2	Non-static → Static	-
m.2.1	Changement de méthode (Static/Non-static)	
m.2.1.1	Static → Non-static	I+L
m.2.1.2	Non-static → Static	L
m.2.2	Changement de méthode (Abstract/Non-abstract)	
m.2.2.1	Abstract → Non-abstract	H+ie(3.1.2)+L
m.2.2.2	Non-abstract → Abstract	H+ie(3.1.1)+L
m.2.3	Changement de type de retour de méthode	
m.2.3.1	Non-abstract method	H+ie(3.1.2)+L
m.2.3.2	Abstract method	H+L
m.2.4	Changement d'implémentation de méthode	L
m.2.5	Changement de signature de méthode	
m.2.5.1	Non-abstract method	I+ie(3.1.2)+L
m.2.5.2	Abstract method	H+L
m.2.6	Changement de portée de méthode	
m.2.6.1	Public → Private	
m.2.6.1.1	Non-abstract method	I
m.2.6.1.2	Abstract method	-
m.2.6.2	Public → Protected	
m.2.6.2.1	Non-abstract method	I ~H *
m.2.6.2.2	Abstract method	-
m.2.6.3	Protected → Private	
m.2.6.3.1	Non-abstract method	H I
m.2.6.3.2	Abstract method	-
m.2.6.4	Protected → Public	
m.2.6.4.1	Non-abstract method	-
m.2.6.4.2	Abstract method	-
m.2.6.5	Private → Public	

m.2.6.5.1	Non-abstract method	-
m.2.6.5.2	Abstract method	-
m.2.6.6	Private → Protected	
m.2.6.6.1	Non-abstract method	-
m.2.6.6.2	Abstract method	-
m.2.7	Ajout de méthode	
m.2.7.1	Abstract method	ie(3.1.1)
m.2.7.2	Non-abstract method	I+ie(3.1.2)+L
m.2.8	Suppression de méthode	
m.2.8.1	Abstract method	ie(3.1.2)
m.2.8.2	Non-abstract method	I + ie(3.1.1)+ L
c.3.1	Changement de classe (Abstract/Non-abstract)	
c.3.1.1	Non-abstract → Abstract	G+H+I+L
c.3.1.2	Abstract → Non-abstract	H+L
c.3.2	Suppression de classe	
c.3.2.1	Non-abstract class	S+G+H+I
c.3.2.2	Abstract class	S+H+I
c.3.3	Dérivation d'héritage de classe	
c.3.3.1	Public → Private	S+I
c.3.3.2	Public → Protected	(S+I) ~H *
c.3.3.3	Protected → Private	H(S+~SG+~S I)
c.3.3.4	Protected → Public	-
c.3.3.5	Private → Public	-
c.3.3.6	Private → Protected	-
c.3.4	Ajout de classe	-
c.3.5	Structure d'héritage de classe	
c.3.5.1	Ajout de classe abstraite	S+G+H+I+ie(3.1.1)+ L
c.3.5.2	Ajout de classe non abstraite	H+L
c.3.5.3	Suppression de classe abstraite	H+ie(3.1.2)+ L
c.3.5.4	Suppression de classe non abstraite	H+L

* : l'expression obtenue est diminuée du nombre de classes du même package (ces classes gardent l'accès après ce changement).

Notons qu'il y a en tout 52 Changements : 12 changements de variable, 25 changements de méthode et 15 changements de classe.

Nous signalons aussi qu'il se peut qu'un changement déclenche un autre changement (un changement déclenché). Par exemple, le changement identifié "m.2.7.1", qui est l'ajout d'une méthode abstraite dans une classe non abstraite déclenche un autre changement identifié : "c.3.1.1" puisque la classe est devenue maintenant abstraite.

Pour indiquer un changement déclenché, nous ajoutons une note à l'expression finale de la forme : ie (change id) Avec : "ie" : " impact expression " : l'expression d'impact et "change id" : se réfère au changement déclenché. Et c'est effectivement le cas aussi des changements identifiés m.2.2.1, m.2.2.2, m.2.3.1, etc.

Annexe B

DC :	Domaine de Connaissance
DTAS:	Decision Theory and Adaptive Systems
EM :	Expectation- Maximisation
GUI :	Interface Utilisateur Graphique.
IC :	Impact de Changement
IS :	Importance Sampling
ISO :	Internationale pour la Standardisation
MCMC :	Markov Chains Monte Carlo
MH :	Metropols Hastings
MPE :	Most probable explanation
MV:	Maximum de vraisemblance
O.O :	Orienté Objet
PSM:	Practical Software and Systems Measurement
RB :	Réseaux Bayésiens
SEI :	Software Engineering Institute
SPI :	Symbolic Probabilistic Inference
UML:	Unified Model Language

D-séparation (blocage)

On dira que X et Y sont *d-séparés* par Z si pour tous les chemins entre Xet Y, l'une au moins des deux conditions suivantes est vérifiée :

• Le chemin converge en un nœudW, tel que W ≠ Z, et W n'est pasune cause directe de Z.
• Le chemin passe par Z, et est soit divergent, soit en série au nœudZ. On a prouvé que si X et Y sont d-séparés par Z, alors X et Y sont indépendants conditionnelle à Z. Ce résultat est très important, car il permet de limiter les calculs de probabilités grâce à des propriétés du graphe. Supposons que X et Y soient d-séparés par Z, et que Z soit connu. Supposons, par ailleurs, que je vienne de calculer p(X|Z). Si une nouvelle information sur Y est alors connue, ce résultat me permet de conserver mon calcul de p(X|Z) comme valeur de p(X|Z,Y).
Combiné avec un autre résultat, qui établit qu'un nœud est d-séparé du reste du graphe par l'ensemble constitué de ses parents, de ses enfants, et des autres parents de ses enfants, cette propriété permet de rendre locaux tous les calculs de probabilités dans un graphe causal.

(« X est d-séparé de Y par Z » est noté (<X | Z | Y>)

Indépendance conditionnelle :

A et B sont indépendants conditionnellement à C ssi :

lorsque l'état de C est connu, toute connaissance sur B n'altère pas A : P(A|B,C) = P(A|C)

Résumé

L’objectif de ce projet est d’améliorer la maintenance des systèmes OO, et d’intervenir plus précisément dans la tâche de l’analyse et la prédiction de l’impact du changement. Parmi plusieurs modèles de représentation, les Réseaux Bayésiens (RBs) constituent une approche quantitative particulière qui peut intégrer l’incertitude dans le raisonnement offrant ainsi des explications proches à la réalité. De plus, avec les RBs, il est aussi possible d’exploiter les jugements des experts pour anticiper les prédictions, dans notre cas sur l’impact de changement. Nous nous intéressons en premier lieu à l’aspect inférence puis éventuellement a l’aspect apprentissage par les RBs Nous proposons une approche probabiliste afin de déterminer l’impact des changements dans les systèmes à objets. Cette approche sert à prédire, pour un changement donné dans un système à objet, l’ensemble des effets de ce changement donnée sous la forme de probabilité. Un des principaux avantages liés à l’utilisation des réseaux bayésiens est l’inférence qui permet de simuler le comportement du système en fonction du jeu de données en entrée. Dans le cadre de cette étude, nous avons implémenté l’algorithme d’inférence « Arbre de jonction», puis nous l’avons utilisé afin d’améliorer l’efficacité de l‘impact de changement.

Mots-clés :

Analyse D’impact De Changement, Maintenance; Modèle Probabiliste; Réseaux Bayésiens; Inférence Bayésienne; Réseaux De Croyance; Réseaux Probabilistes; Impact De Modifications; Maintenance Des Systèmes Orientée Objet; Algorithmes D’inférence.