

Développement d'un outil d'édition et de manipulation de réseaux bayésiens

R. Agathon
Telecom Nancy

Mail : raphael.agathon@telecomnancy.eu

V. Wegmann-Serin
Télécom Nancy

Mail : valentin.wegmann-serin@telecomnancy.eu

C. Simon

Laboratoire du CRAN

Mail : christophe.simon@univ-lorraine.fr

Java, Matlab, inférence, probabilité, graphe probabiliste

travailler sur les réseaux bayésiens. [3] [2]

Résumé—Ce projet de recherche s'inscrit dans le cadre d'études avec l'aide de graphe probabiliste. Les chercheurs utilisent principalement l'outil Matlab afin d'effectuer leur simulation cependant il n'existe pas d'IHM (Interface Homme Machine) dynamique qui permette de créer des réseaux bayésiens et d'interagir directement après avoir rentré les probabilités. Ainsi, le problème est de développer une interface permettant d'utiliser des réseaux bayésiens et qui permettent aussi de faire des calculs d'inférences pour étudier les scénarios donnés. Cependant, développer une IHM sous Matlab n'est pas pratique ainsi, l'application doit être développée sous Java et doit pouvoir communiquer dynamiquement avec Matlab. Afin de répondre à ce problème, nous avons choisi d'analyser les différents outils existant et de les trier pour enfin améliorer celui qui se rapprochait le plus de nos critères à savoir la présence d'une IHM, la communication avec Matlab et Java et la possibilité de faire des calculs.

I. INTRODUCTION

Dans le cadre de la recherche, les chercheurs sont régulièrement amenés à utiliser des outils basés sur des graphes probabilistes ou quasi-probabilistes.

Par exemple, l'étude de la fiabilité d'un système passe par l'analyse d'un graphe comprenant tous les différents composants du système avec leurs états et probabilités associés - la fiabilité étant l'étude des défaillances des systèmes d'un point de vue statistique -. Les graphes utilisés dans cette étude sont des réseaux bayésiens. Ce sont des modèles probabilistes que l'on trouve sous la forme de graphe acyclique et orientés. Ces modèles utilisent la méthode de l'inférence qui est un procédé permettant de déduire la probabilité d'un événement à partir de celles d'autres événements déjà évalués. Elle repose sur le théorème de Bayes qui repose sur les probabilités conditionnelles et permettant de connaître la probabilité d'un événement : A sachant B ; lorsque l'on connaît la probabilité de A, B, et de B sachant A.

Cette usage est souvent accompagné d'une simulation sur logiciel tel que Matlab. Cependant, Matlab ne dispose pas d'outils pratique avec interface graphique permettant de

TABLE DES MATIÈRES

I	Introduction	1
II	Analyse des outils existants	2
III	Méthodologie	2
III-A	Analyse de l'outil DynGraph	2
III-B	Implémentation des réseaux bayésiens .	3
III-C	Implémentation des probabilités	4
III-D	Calcul de l'inférence	5
III-E	Découpage de l'édition de réseaux bayésiens	5
III-F	Modification de l'interface graphique .	5
IV	Résultats	6
V	Conclusion	7
VI	Remerciements	7
	Références	7

Les chercheurs utilisant principalement l'outil Matlab qui offre un environnement numérique qui donne donc la possibilité de faire des calculs, d'évaluer et de simuler des scénarios. Dans ce cadre, il est nécessaire d'avoir une interface homme machine permettant la création de graphes et de réseaux bayésiens. Cette interface doit aussi nous permettre de manipuler et d'interagir avec les graphes créer et par la suite de faire des calculs d'inférences. Cependant, la création d'interface graphique sous Matlab n'est pas aisée et le logiciel que nous utilisons est donc Java. Ainsi, nous avons une contrainte d'intercommunication entre les deux logiciels. Afin de répondre au problème donné nous avons choisi de trier tous les logiciels déjà existants puis de surcharger un programme possédant en partie les critères demandés.

II. ANALYSE DES OUTILS EXISTANTS

Il existe de nombreuses applications permettant l'édition de réseaux bayésiens [1]. Celles-ci ont toutes leurs spécificités que l'on peut organiser sous forme de catégories.

- Langage de programmation : Il existe des boîtes à outils pour les réseaux bayésiens dans presque tous les langages de programmation. Les plus connues sont Java Bayes qui est en Java et Bayes Net Toolbox sous Matlab. Il existe d'autres applications dont on ne connaît pas le langage utilisé. Celle-ci sont la plupart du temps payantes ou sont des applications web. La plus connue d'entre elles est BayesiaLab. C'est une application fermée offrant une interface graphique mais sans interaction avec Matlab. BayesiaLab possède un API pour les calculs d'inférence mais cette API ne passe pas par l'interface graphique.

Les contraintes imposent que l'outil à développer soit sous Java et Matlab en même temps. Dimple est une API qui possède une version Java et une Matlab, mais il n'y a aucune communication entre les deux ce qui ne correspond pas à la problématique et aux contraintes. DynGraph, l'application utilisée est la seule à répondre à cette contrainte. Cette contrainte d'intercommunication est la plus restrictive car l'outil doit permettre l'utilisation de Matlab et de Java en parallèle. Toutes les modifications faites dans l'un des langages doit directement impacter l'autre.

- Interface graphique : Parmi toutes les applications connues de réseaux bayésiens, seules 30% possèdent une interface graphique. La plupart de celles-ci sont en Java comme Java Bayes ou payantes comme BayesiaLab. Bayes Net Toolbox, CRFtoolbox, DBNbox, OpenGM2 et Probabilistic Modeling Toolkit sont toutes les applications codées sous Matlab. Aucune d'entre elles ne possède d'interface graphique.

La contrainte de l'interface graphique impose que l'outil en possède déjà une ou que celle-ci soit codée par dessus. DynGraph possède une interface graphique en Java Swing, qui offre une grande liberté contrairement à ce qui peut être fait sous Matlab.

- Le prix : l'outil doit être open source et gratuit. La plupart des applications existantes le sont, mais certaines sont payantes comme BayesiaLab (BayesiaLab a été utilisé dans le processus de construction de l'outil de développement de réseaux bayésiens). DynGraph a été codée par des étudiants de notre école, dans le même environnement que le nôtre. Cette application est open source et gratuite et nous pouvons facilement communiquer avec les développeurs.
- Réseaux bayésiens et Inférence : Les différentes applications permettent de générer des réseaux

bayésiens et de faire de l'inférence sur ceux-ci. Il existe différents algorithmes pour le calcul de l'inférence comme JTree et VarElim. DynGraph, l'outil utilisé ne possède aucun outil d'édition de réseaux bayésiens ou de calcul d'inférence. Cet outil permet de construire des graphes pouvant être à nœuds colorés et arcs évalués, orientés ou non.

DynGraph est une application dédiée à l'édition et à la manipulation de p-graphes colorés sous Matlab. Cette application Java en open source gratuite assure une interaction dynamique avec Matlab. En revanche elle n'est pas restreinte aux graphes acycliques non colorés et est exempte d'outil d'inférence. En restreignant les possibilités de DynGraph dans la construction de graphes, en associant la saisie de probabilités et l'interfacage d'un outil d'inférence alors DynGraph peut être une solution à la problématique.

Pour combler ces lacunes, il a été choisi de combiner cet outil avec une application Java d'édition de réseaux bayésiens. L'application sélectionnée est Jayes.

Jayes est un outil d'édition de réseaux bayésiens, open source et gratuit qui ne dispose pas d'interface graphique. Jayes permet aussi de faire des calculs d'inférences. Parmi les autres applications Java, Jayes possède un code simple de compréhension et permettant une réécriture du code plus facile.

Jayes ne dispose d'aucun contrôle sur l'acyclicité des graphes, les dimensions des tables de probabilités ne peuvent être changées qu'une seule fois et il n'y a aucun contrôle sur la saisie de données.

III. MÉTHODOLOGIE

Pour arriver à l'objectif, le travail a été divisé en deux parties. Une partie de compréhension et de modification de l'outil DynGraph réalisée par M. Wegmann-Serin et une seconde partie sur la compréhension et l'implémentation des réseaux bayésiens ainsi que des calculs d'inférences. Cette seconde partie a été réalisée par M. Agathon.

A. Analyse de l'outil DynGraph

DynGraph est un outil permettant la création et l'édition de graphes. Celui-ci est implémenté en Java et se lance depuis Matlab. L'interface (cf fig 1) est composée d'un menu basique et d'une barre composée de boutons permettant d'éditer le graphe. L'éditeur ne permet actuellement que de créer des graphes qui sont orientés ou non, colorés ou non. L'éditeur ne permet pas d'imposer des contraintes lors de la création du graphe telle que l'acyclicité ou la non-évaluation des arcs. Chaque graphe créé est représenté par une liste de nœuds et d'arcs (un arc étant défini par un nœud-début et un nœud-fin ainsi que sa couleur et sa direction). Toute la partie utile aux calculs n'est pas présente : la liste des successeurs et prédécesseurs ou une matrice d'adjacence ou d'incidence. Le

côté front-end est implémenté, il s'agit de tous l'aspect visuel. Pour ce qui est du back-end, il s'agit de la communication entre Java et Matlab qui est présente puisque chaque action effectuée sur l'interface - en java - doit être faite en Matlab et inversement. Ainsi, Dyngraph est une IHM permettant l'édition de graphe simple mais pas de réseaux bayésiens et ne permettant pas de faire de calculs d'inférence. Les stratégies possibles pour remédier au problème était de trouver un autre outil offrant plus de fonctionnalités ou d'améliorer Dyngraph. Nous avons choisis de l'améliorer puisque Dyngraph possèdent beaucoup de critères que nous recherchons telle que l'intercommunication Matlab-Java dynamique et le fait que ce soit déjà une IHM. Ainsi, en ajoutant des restrictions offrant la possibilité de créer des graphes acyclique non colorés et en couplant le logiciel avec des outils permettant de faire des calculs d'inférence. Dyngraph se pose alors comme solution convenable pour notre problème.

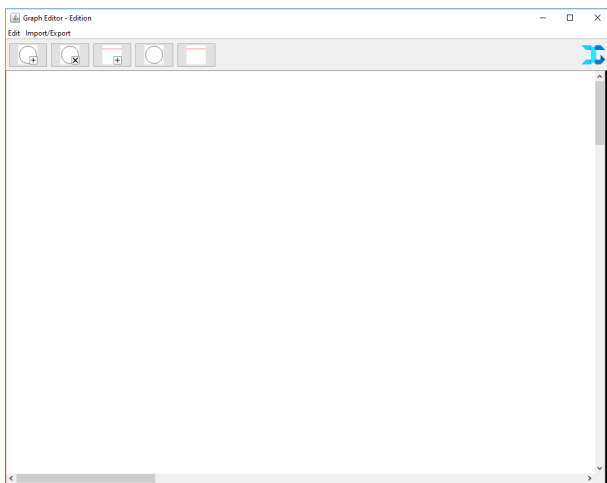


FIGURE 1. interface de Dyngraph

B. Implémentation des réseaux bayésiens

Un réseau bayésien est un système représentant la connaissance et permettant de calculer des probabilités conditionnelles apportant des solutions à différentes sortes de problématiques.

La structure de ce type de réseau est simple : un graphe dans lequel les nœuds représentent des variables aléatoires, et les arcs (le graphe est donc orienté) reliant ces dernières sont rattachées à des probabilités conditionnelles.

Le graphe doit être acyclique : il ne contient pas de boucle. Les arcs représentent des relations entre variables qui sont soit déterministes, soit probabilistes. Ainsi, l'observation d'une ou plusieurs causes n'entraîne pas systématiquement l'effet ou les effets qui en dépendent, mais modifie seulement la probabilité de les observer.

a) Algorithme d'acyclicité

: Ainsi, pour transformer un graphe en réseau bayésien il faut vérifier les contraintes d'acyclicité, de directivité du graphe

et de non coloration. La coloration peut être enlevée par une simple modification car elle n'impacte pas les données présentes dans le graphe.

Contrôler que le graphe est dirigé se fait par une simple vérification sur tous les arcs du graphe.

Pour l'acyclicité il existe différents algorithmes de détection de cycle dans un graphe. Des algorithmes comme celui du tri topologique, ou celui de Floyd's Tortoise and Hare sont performants mais nécessitent des outils qui implique une totale refonte de la gestion des graphes dans l'outil DynGraph. Cela car ils nécessitent des éléments comme pour chaque nœud une liste des nœuds père et une liste des nœuds fils, ainsi qu'une matrice d'adjacence pour le graphe. C'est élément ne sont pas présent dans l'outil DynGraph.

Nous avons donc décidé de coder notre propre algorithme de détection de cycle. Cet algorithme a été spécifiquement construit pour notre cas d'étude : permettre à un utilisateur de savoir si il y a un cycle dans le graphe et si oui afficher la ou les zones à problèmes.

L'algorithme repose sur le fait que tous nœud uniquement père ou uniquement fils ne fait pas partie d'un cycle. Par suppression successive de ces nœuds et arcs associés l'algorithme déterminer s'il y a un ou des cycles dans la ou les zones restantes.

Data: Graphe G contenant une liste de nœuds et une liste d'arcs

Result: liste des nœuds à problème : N

N liste de nœuds initialiser avec les noeuds de G;

A liste d'arcs initialiser avec les arcs de G;

test = True;

while test = True **do**

 Ra liste d'arcs initialement vide;

 Rp liste des nœuds père, initialement vide;

 Rf liste des nœuds fils initialement vide;

 Rnpf liste des nœuds non père et non fils initialement vide;

 test = False;

forall Arc a dans A **do**

 Ajout du noeud head à la liste Rp;

 Ajout du noeud tail à la liste Rf;

end

forall Node n dans N **do**

 Ajout des nœuds non présent dans l'intersection entre Rp et Rf dans Rnpf;

end

if Rnpf de taille non nulle **then**

 test = True;

forall Arc a dans A **do**

if Si un des nœuds de a appartient à Rnpf **then**

 Ajout de l'arc a dans la liste Ra;

end

end

 Suppression des arcs de Ra dans la liste d'arcs A;

 Suppression des nœuds de Rnpf dans la liste de nœuds N;

end

end

Algorithm 1: Algorithme de détection de cycle

Cet algorithme permet de détecter la présence de cycle dans un graphe et permet d'identifier les nœuds inscrits dans un cycle. L'algorithme à une complexité en $O(\max(n,a))$ avec n le nombre de nœud et a le nombre d'arc. Il est possible que certains nœuds, non présent dans un cycle soient inscrits dans les nœuds à problème. Cela car ils sont à l'intérieur d'un ou plusieurs cycle et sont donc classé comme nœuds à problème.

Cet algorithme utilise iniquement les outils présents dans DynGraph ce qui le rend compatible extrêmement compatible avec cette application

b) Transformation du graphe en réseau bayésien :

Comme cela est écrit plus haut, le graphe présent dans l'interface DynGraph ne contient aucune information permettant de faire des calculs à partir d'éléments du graphe. Le graphe n'est présent que pour la partie graphique. Il permet à l'interface graphique de générer l'affichage. Deux méthodes ont été envisagées pour la construction de réseaux bayésiens.

Surcharger le graphe de l'outil DynGraph

Cette méthode est la première à laquelle nous avons pensé. Par surcharger il faut comprendre que pour chaque nœud il doit être ajouté :

- la liste des nœuds pères
- la liste des nœuds fils
- une table de probabilités

Ainsi que toutes les fonctions et procédures pour que ces différents éléments se mettent à jour automatiquement.

De plus comme il est écrit dans la partie III-A l'outil d'édition de réseaux bayésiens est une extension de l'outil DynGraph. C'est pour cela que nous n'avons pas choisi cette méthode car elle implique d'énormes modifications sur le code existant.

Ajouter un graphe de réseau bayésien en parallèle du graphe de l'outil DynGraph

Cette seconde méthode est celle qui est implémentée. Les réseaux bayésiens et le calcul d'inférence ont été ajoutés via l'ajout de l'outil Jayes (éditeur de réseaux bayésiens, Java, open source, sans interface graphique).

Cette seconde méthode nous permet de ne pas surcharger l'outil initial. Ainsi, si le graphe n'est pas un réseau bayésien, le graphe reste identique. Si le graphe est un réseau bayésien, le graphe visible est celui de DynGraph et en back-end il y a le graphe de Jayes.

Lors de la transformation en réseau bayésien. Il y création du graphe de Jayes.

Data: Graphe G contenant une liste de nœuds et une liste d'arcs

Result: Graphe de réseau bayésien R

N liste de nœuds initialiser avec les noeuds de G;

A liste d'arcs initialiser avec les arcs de G;

test = True;

forall Node n dans N **do**

 Création du nœud dans graphe R;

 Ajout des états True et False pour le nouveau noeud de R;

end

forall Arc a dans A **do**

 Ajout du nœud père dans la liste des nœuds père du fils;

end

Algorithm 2: Algorithme de transformation de graphe

Après cette étape le graphe réseau bayésien est valide. Toutes les tables de probabilités ont la bonne taille et il ne reste plus qu'à les remplir.

C. Implémentation des probabilités

Dans un réseau bayésien chaque nœud possède une table de probabilités. Cette table possède un nombre de lignes et de colonnes dépendant d'un certain nombre de paramètres.

Le nombre de lignes dépend des différents états que peuvent prendre les variables des nœuds parents de ce nœud. Ce nombre est égal au produit du nombre d'états de chaque nœud

parent peut prendre. Si un nœud à 3 nœuds pères, le premier à 2 états, le second et le troisième à 3 états, il y aura $2 \times 3 \times 3 = 18$ lignes. Le nombre de colonnes, dépend du nombre d'état du nœud. Ainsi chaque case du tableau contient la probabilité que le nœud soit dans un état sachant que les nœuds pères sont dans un certain état.

En plus de cette contrainte sur le nombre de lignes et de colonnes, il y a des contraintes sur les valeurs entrées en paramètre. Premièrement il doit y avoir le bon nombre de valeurs et la somme des valeurs sur une ligne doit être égal à 1. Cette dernière contrainte est une contrainte probabiliste. Une table de probabilité est considérée comme valide lorsque toutes

D. Calcul de l'inférence

Le calcul de l'inférence repose sur l'algorithme de l'arbre de jonction [4]. Cet algorithme est présent dans l'outil Jayes. Pour pouvoir l'utiliser sans problème il suffit de vérifier que tous les nœuds du graphe ont bien leur table de probabilités valide.

E. Découpage de l'édition de réseaux bayésiens

Éditer un réseau bayésien impose certaines contraintes sur le graphe comme son acyclicité mais aussi des modifications dynamiques liées à toutes évolutions de celui-ci. Par exemple, l'ajout d'un arc entre deux nœuds va modifier la liste des nœuds parent pour l'un et la liste des nœuds enfant pour l'autre. Mais cela implique une mise à jour de la table de probabilité du nœud enfant. Son nombre de lignes est multiplié par le nombre d'états du nœud ajouté en parent. Pour minimiser ces problèmes de modifications dynamiques et minimiser le nombre de calculs liés aux contraintes, il a été choisi de découper l'édition et l'utilisation de réseaux bayésiens en plusieurs étapes. Ces étapes correspondent aux pratiques utilisateur : une édition de graphe, quelques modifications de probabilités et de nombreux calculs d'inférences.

a) Étape de création de graphe

: Cette étape est celle présente dans l'outil DynGraph. L'utilisateur a la possibilité de créer le graphe qu'il souhaite, acyclique ou non. Durant cette étape aucun réseau bayésien n'est instancié car l'outil ne sait pas si l'utilisateur veut transformer son graphe en réseau bayésien. Si une transformation est demandée, l'algorithme de détection de cycle est lancé. Cette méthode de découpage fait qu'il n'y a qu'un seul appel à l'algorithme de détection de cycle.

b) Étape de mise en place des probabilités

: Dès lors que le graphe est transformé en réseau bayésien, tout retour à l'étape précédente invalide le réseau bayésien. Si des nœuds et des arcs sont ajoutés ou supprimés, certaines tables de probabilités changent de dimensions qui rend leurs anciennes valeurs fausses. La table de probabilité de chaque nœud dépend de ses états et des différents états que peuvent prendre ses nœuds parent. Cette construction de table

est faite durant la transformation du graphe. La table est automatiquement initialisée avec le bon nombre de colonnes et de lignes.

L'ajout de probabilités se fait nœud par nœud. Chaque modification d'une table de probabilités n'est prise en compte que si pour chaque ligne, la somme des valeurs est égale à 1. Un passage à l'étape de calcul d'inférence n'est validé que si tous les nœuds ont une table valide.

c) Étape de calcul d'inférence

: Les calculs d'inférences ne nécessitent pas de contrôle sur les paramètres rentrés. Si un cas d'inférence est invalide, c'est le calcul qui le montrera. Le calcul de l'inférence repose uniquement sur une bonne construction du réseau bayésien.

F. Modification de l'interface graphique

Une fois la production du code terminée, les nouvelles fonctionnalités doivent être incorporées dans l'interface. Ce qui a été introduit en premier fut la création de réseaux bayésiens car il s'agit du point de départ de l'analyse scientifique. Cette phase permet d'éditer et de manipuler des graphes, lorsque l'on clique sur le bouton DAG (Directed Acyclic Graph), on vérifie que le graphe ne présente pas de cycle (cf. figure 2), que tous les arcs soient dirigés et la coloration du graphe est supprimée. Sur la figure 2, on peut voir que certains nœuds sont colorés en vert, indiquant qu'ils font partie d'un cycle. Le nœud numéro 4 est aussi coloré même s'il ne fait pas partie d'un cycle car il est situé entre deux cycles.

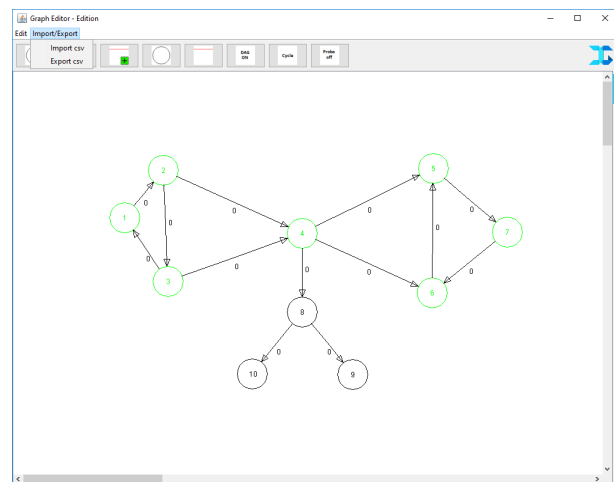


FIGURE 2. présence de cycle

Cette étape est la première dans le processus d'étude. Viens ensuite la phase de mise en place des probabilités. L'utilisateur sélectionne les nœuds racines puis remplit la table de probabilité associée (cf. Figure 3). La figure 3 représente l'interface permettant de remplir la table de probabilité qui est associée au nœud racine. Pour les nœuds fils, l'utilisateur remplit les tables de probabilités conditionnelles associées.

Enfin vient la dernière phase qui est celle du calcul des inférences où l'utilisateur aura le choix de lancer une

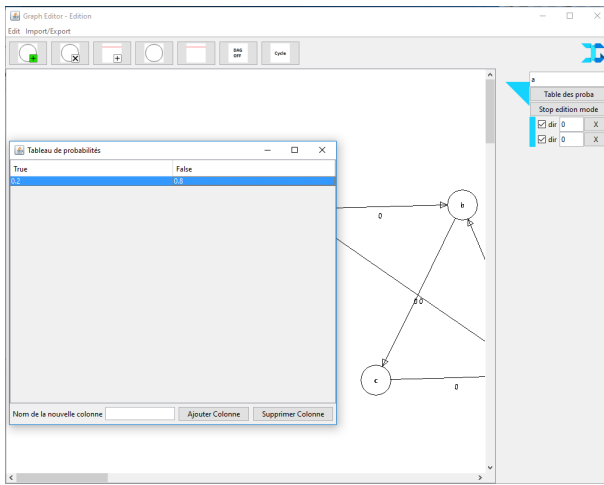


FIGURE 3. table de probabilité

simulation et en choisissant quel événement sont sûrs.

Nous avons choisis de découper le processus en trois étape : la création du graphe, la mise en place de probabilité et le calcul d'inférence. Le but de cette découpe est de réduire le nombre de contrôle effectué à chaque modification du graphe. Du fait que l'on utilise des réseaux bayésiens qui possèdent de fortes contraintes - acyclique, dirigé, non-coloré -, le nombre de contrôle à effectuer à chaque fois que l'on modifie le graphe serait lourd. Ainsi, n'effectuer les contrôles qu'une seule fois après avoir fini d'éditer le graphe permet de réduire ce nombre. De plus, les utilisateurs i.e les chercheurs ont l'habitude de créer le graphe et de ne pas le modifier par la suite, d'ajuster les probabilités et enfin de faire les calculs d'inférences nécessaires.

IV. RÉSULTATS

L'outil créer permet l'édition de réseau bayésiens tous en conservant les possibilité qu'offre l'outil DynGraph. Les graphes de réseau bayésien et les calculs d'inférence sont fonctionnels mais peuvent être optimisés.

Afin d'illustrer le logiciel, prenons un modèle mathématique(cf. Figure 3). Soit un système simple à trois composants, le système fonctionne si R3 fonctionne et que R1 ou R2 fonctionnent.

Chaque composant R_i suit la loi de probabilité suivante :

$R_i(t) = \exp(-\lambda_i t)$ où R représente la probabilité que le composant fonctionne avec :

- $t = 1000 \text{ h}$
 - $\lambda_1 = 10^{-3} \text{ h}^{-1}$
 - $\lambda_2 = 10^{-4} \text{ h}^{-1}$
 - $\lambda_3 = 10^{-3} \text{ h}^{-1}$
- (1)

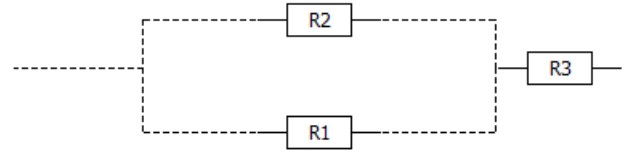


FIGURE 4. Système simple à trois composants

et

$$R_s(t) = R_3(t) \cdot [1 - (1 - R_1(t)) \cdot (1 - R_2(t))] \quad (2)$$

Nous avons notre système, il faut le représenter sous forme de réseau bayésien (cf. figure 5). Les nœuds racines représentent les trois composants du système, étant donné que R1 et R2 sont en parallèle, on obtient le nœud $R1+R2$ au sens du ou logique. Enfin, R3 et $R1+R2$ donne le nœud donnant la probabilité du système.

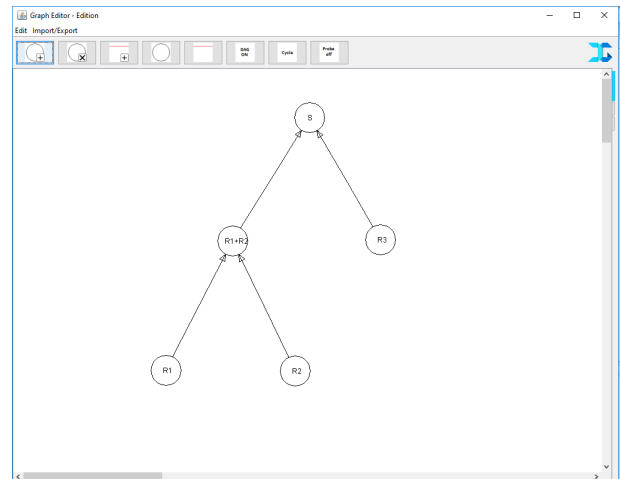


FIGURE 5. Système simple à trois composants

Une fois le réseau bayésien établi, il faut rentrer les probabilités dans les tables de chaque nœud (cf. figure 6). La ta de gauche représente les états des nœuds parents tandis que la table de droite est celle du nœud fils.

Lorsque toutes les probabilités ont été rentrées, on passe sur le calcul d'inférence : par exemple, on observe que le système ne marche pas et on veut retrouver la loi de probabilité de R1. Ainsi on indique au logiciel que le nœud S est dans l'état "Not

Tableau de probabilités			
R1	R2	Working	Not Working
Working	Working	1.0	0.0
Working	Not Working	0.0	1.0
Not Working	Working	0.0	1.0
Not Working	Not Working	0.0	1.0

FIGURE 6. Table de probabilités du noeud R1+R2

Working". Si on fait d'autres observations, on les indiquent au logiciel. Enfin, on répercute ceci sur les nœuds non-observés et on retrouve la probabilité de R1.

Prenons maintenant un exemple plus concret : Soit un système défaillant, on veut connaître qu'elle est la cause de ce problème. Dans ce cas, le réseau bayésien fait office d'arbre de diagnostic pour l'isolation de défaut(cf. figure 7). Le nœud S représente le système et possède deux états : Vrai s'il est défaillant, Faux s'il est en état de marche. Chaque nœud C représente une cause du problème, ces nœuds pouvant être redécoupé en sous-causes. Chacun des nœuds C possède aussi deux états : Vrai s'il est la cause du problème, faux sinon. Au départ, les nœuds racines sont initialisés à faux (0 pour la valeur vraie, 1 pour la valeur fausse). Chacun des nœuds fils étant des nœuds XOR. Les nœuds racines sont initialisés à 0 pour l'état vrai et 1 pour l'état faux. Les autres nœuds possèdent des tables probabilités induites des probabilités des nœuds racine. Lorsque le calcul d'inférence est lancé avec comme entrée vrai pour S. On obtient les probabilités pour les autres nœuds. L'utilisateur choisit le nœud avec la plus grande probabilité d'être dans l'état Vrai. Si ce nœud est dans l'état Vrai, on a trouvé la cause du problème. Sinon, on observe donc que le nœud est dans l'état faux et on notifie l'application. Puis on réitère le calcul d'inférence en sachant toujours que le système ne marche pas. Puis on rechoisit le nœud qui a la plus grande probabilité d'être dans l'état vrai et ainsi de suite jusqu'à trouvé la cause du problème.

Les réseaux bayésiens peuvent aussi être utilisé pour des états non booléens. Cela s'effectue de la même manière que pour le système simple à trois composants.

V. CONCLUSION

L'objectif de notre projet à été de créer un outil Matlab et Java pour l'édition de réseaux bayésiens. Avec l'amélioration de Dyngraph, nous avons répondu à cette problématique car nous pouvons créer des réseaux bayésiens, effectuer des calculs d'inférence dessus et donc faire l'étude de différents systèmes. La prochaine étape de se projet serait d'améliorer encore le dynGraph actuel afin que la communication Matlab permette aussi d'exporter les données dû à la phase de calculs dans le logiciel Matlab.

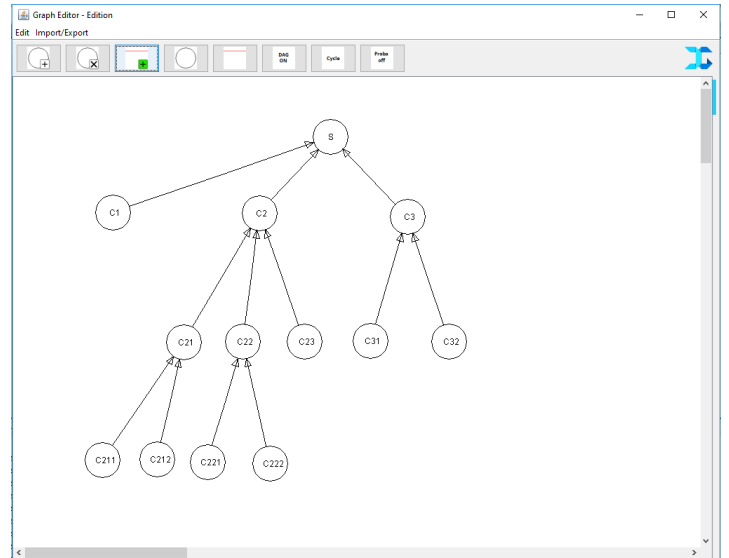


FIGURE 7. Arbre de diagnostic pour l'isolation de défaut

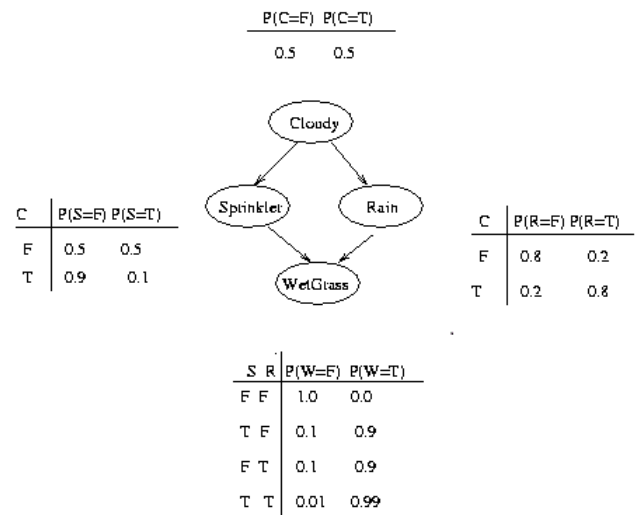


FIGURE 8. Réseau bayésien avec état non booléen

VI. REMERCIEMENTS

Nous souhaitons adresser nos remerciements aux personnes nous ayant aidé dans la rédaction de cet article, en particulier à M.Simon, notre encadrant de PIDR qui a su nous éclairer dans l'avancée de notre projet.

Nous remercions aussi M.Moulet et M.Domingez les créateurs de DynGraph de nous avoir permis d'utiliser leur outil et pour l'aide qu'il nous ont apporté tout au long de notre projet.

RÉFÉRENCES

- [1] Kevin Murphy. Software packages for graphical models.

- [2] Kevin Murphy. Software for graphical models : a review. *ISBA Bulletin*, 2007.
- [3] Linda Smail. *Algorithmique pour les Reseaux Bayesiens et leurs extensions*. PhD thesis, Université de Marne la Vallée, 2004.
- [4] Salem Benferhat Salma Smaoui. *Réseaux causaux possibilistes pour le traitement des interventions*. PhD thesis, CRIL, Université d'Artois, 2007.