Université Paris 6
Licence d'informatique
Maîtrise de Sciences et Techniques
(Experts en Systèmes Informatiques)

Travaux Pratiques n° 2

Module de Programmation
T. Hardin
Année 1998-1999

1 But du TP 2

Le but du TP 2 consiste à développer un algorithme permettant la saisie et l'exploitation d'un réseau bayésien. Ce dernier est un outil issu de l'Intelligence Artificielle, qui permet la gestion de situations incertaines grâce à l'utilisation de probabilités. Un réseau bayésien est composé d'une part d'une représentation graphique, et d'autre part d'une loi de probabilité stockée sous une forme compacte.

Le TP se déroulera en 3 parties : dans la première, vous écrirez l'algorithme permettant la saisie de la partie graphique du réseau. Dans la deuxième, vous vous attaquerez à la saisie de la loi de probabilité. Enfin, dans la troisième partie, vous rédigerez l'algorithme de calcul qui permet l'exploitation du réseau, c'est-à-dire qui permet de quantifier dans quelle mesure une situation paraît plus ou moins incertaine qu'une autre.

Vous trouverez dans le répertoire `/Infos/caml-ens/98-99/tp2` un certain nombre de fichiers dont vous aurez besoin. Votre mission, si vous l'acceptez, sera donc, en premier lieu de rapatrier tous ces fichiers chez vous (à cet effet, un petit `cp -r` sera le bienvenu).

2 Partie I du TP

Le but de cette partie est de programmer une application graphique qui permette à l'utilisateur, en l'occurence vous, de créer aisément des réseaux bayésiens. Cette application propose les fonctionnalités suivantes, que l'on peut sélectionner en cliquant sur les boutons appropriés (cf. figure 1 sur la page suivante) :

- création d'un nouveau noeud (variable aléatoire) dans le réseau (accessible grâce au bouton «`add node`») ; cliquer dans la partie droite de la fenêtre CAML suffit pour créer le noeud ;
- suppression d'un noeud (bouton «`remove node`») ; cliquer sur un noeud le détruit, ainsi que l'ensemble des arcs qui lui sont adjacents ;
- déplacement d'un noeud sur l'écran (bouton «`move node`»). Cette fonctionnalité est assez utile pour éviter que trop d'arcs ne se croisent ; cliquer sur le noeud, laisser le bouton de la souris appuyé pendant tout le déplacement, et relâcher lorsque le noeud est à la bonne place ;
- ajout d'un arc au réseau (bouton «`add arc`») ; cliquer sur l'un des noeuds extrémités de l'arc, laisser le bouton de la souris appuyé, relâcher le bouton sur l'autre noeud ;
- suppression d'un arc (bouton «`remove arc`») ; cliquer sur l'un des noeuds extrémités de l'arc, laisser le bouton de la souris appuyé, relâcher le bouton sur l'autre noeud ;
- fin de la saisie du réseau bayésien (bouton «`end construction`»).

2.1 Mais qu'est-ce qu'un réseau bayésien ?

2.1.1 Monsieur NASA découvre les probabilités

Lors des débuts de la conquête spatiale, moulte fusées se sont écrasées à cause de problèmes de propulsion. C'était bien évidemment embêtant, mais, bon, sans trop de gravité : le contribuable américain pouvait se permettre de gaspiller quelques millions de dollars. Mais arriva un moment où la NASA voulut envoyer des hommes dans l'espace et, un peu plus tard, où elle décida de créer un engin qui, contrairement aux fusées conventionnelles, pourrait effectuer plusieurs vols spatiaux : la navette spatiale. Et là, un problème de propulsion signifiait non seulement une perte de milliards de dollars, mais aussi une perte humaine, ce qui n'était plus du tout admissible.

Aussi monsieur NASA réfléchit-il à un moyen de gérer *rapidement* tous les incidents des propulseurs de la navette. Impossible de laisser des experts traiter ces problèmes en temp réel : les temps de réaction humains sont beaucoup trop lents. Monsieur NASA décida donc de confier cette tâche à un système informatique.

Il alla voir le mathématicien et lui tint à peu près ce langage :
– Bonjour monsieur le mathématicien, j'aimerais réaliser un logiciel qui me permette de prédire les pannes des propulseurs de ma navette spatiale. Nombreuses sont les pannes qui ont à peu près les mêmes conséquences. Si j'arrivais à diagnostiquer quelle est la panne la plus probable, je pourrais prendre les mesures appropriées.

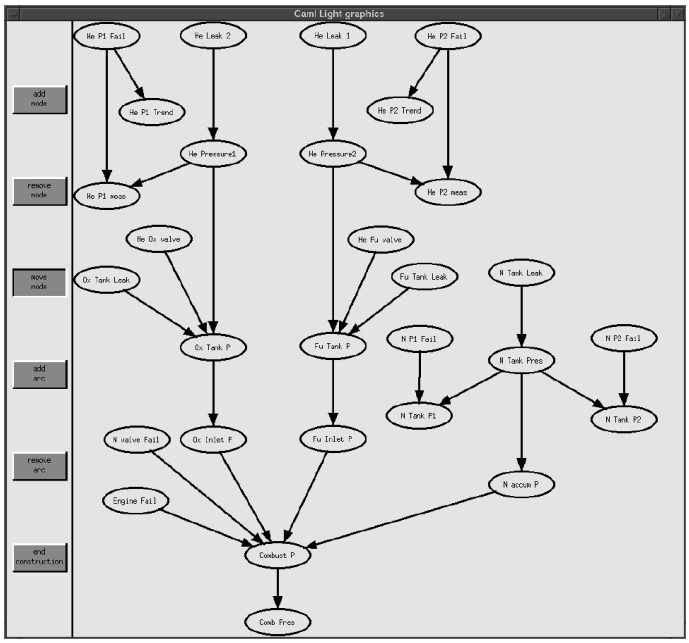


FIG. 1: Aspect de l'écran à la fin de la partie I.

Est-ce que vous auriez un outil pour faire ça ?
– Vous l'avez déjà cité : vous pouvez utiliser des probabilités.
– Mais comment ça marche ?
– C'est simple, dans les année 30, Kolmogorov a défini les probabilités de la manière suivante :

Définition 1 (Probabilité) *Soit Ω un ensemble fini* ***d'événements (élémentaires)****. Une probabilité $P(\cdot)$ est une fonction de l'ensemble des sous-ensembles de Ω (les événements) dans $[0,1]$, qui vérifie les trois propriétés suivantes :*

1. $\forall A,\ P(A) \geq 0$.
2. $P(\Omega) = 1$ *et* $P(\emptyset) = 0$ *(la probabilité de l'événement qui arrivera de manière certaine est égale à 1, et celle de l'événement qui n'arrivera jamais est égale à 0).*
3. $\forall A, B$ *tels que* $A \cap B = \emptyset$, $P(A \cup B) = P(A) + P(B)$.

L'idée des probabilités, c'est tout simplement d'associer un nombre à chaque événement, de telle sorte que plus ce nombre est élevé, plus l'événement a de chances de se réaliser.
– Quelle est la différence entre un événement élémentaire et un événement qui ne l'est pas ?
– Tous les événements sont des sous-ensembles de Ω, mais ceux qui sont élémentaires ne peuvent être décomposés en une union de sous-ensembles plus petits. Prenez l'exemple du jet d'un dé à six faces. Ω représente l'ensemble des faces sur lesquelles peut retomber le dé : $\Omega = \{$un,deux,trois,quatre,cinq,six$\}$. Dans ce cas, $P($un $\cup$ deux $\cup$ trois$)$ représente le nombre de chances que le dé a de tomber sur la face un, la face deux ou la face trois. Ce n'est pas un événement élémentaire parce qu'on peut le décomposer en trois événements : un, deux et trois. Par contre trois est un événement élémentaire parce qu'on ne peut pas le décomposer.
– C'est bien joli votre théorie, mais si Ω a 6 éléments, il a $2^6 = 64$ sous-ensembles possibles. Si je dois choisir un nombre à affecter à chacun d'eux, je ne pourrai jamais utiliser vos probabilités dans des situations complexes, par exemple lorsque Ω a 100 éléments !
– Non, en fait, grâce à la propriété 3, il suffit de déterminer les probabilités des 6 événements élémentaires de Ω pour que les probabilités de tous les autres événements soient déterminés. Par exemple, $P($un $\cup$ deux $\cup$ trois$) = P($un$) + P($deux$) + P($trois$)$.
– Quand je veux calculer la probabilité d'un événement, je le décompose en une union d'événements élémentaires et je calcule la somme de leurs probabilités, c'est bien ça ?
– Oui, c'est tout à fait cela !
– Et si je voulais utiliser des probabilités pour décrire l'incertitude sur le résultat des jets de 2 dés à 6 faces ?
– Eh bien dans ce cas, Ω est égal à l'ensemble de tous les couples (i, j), où i représente le résultat de premier dé, et j le résultat du second. Il faut bien que vous compreniez que ce qui arrive après le jet des 2 dés doit

forcément pouvoir être décrit à l'aide **d'un seul** événement élémentaire.
Introduisons un peu de vocabulaire : $P(\text{trois}) = \frac{1}{6}$ représente la probabilité que l'événement trois arrive, autrement dit que le jet de dé ait pour résultat trois. En maths, on pourra formuler cela autrement : $P(\text{«résultat du jet»} = \text{trois}) = \frac{1}{6}$ et on désignera «résultat du jet» sous le nom de variable aléatoire. Si vous voulez, une variable aléatoire est une synthèse de plusieurs événements : par exemple, il est légitime de parler de la probabilité que la variable «résultat du jet» prenne les valeurs un, deux ou trois. On pourra noter cela de la manière suivante : $P(\text{«résultat du jet»} = \text{un, deux ou trois}) = \frac{1}{6}$. Remarquez que les événements ne peuvent prendre que des valeurs booléennes : «vrai» s'ils se réalisent et «faux» sinon, alors que les variables aléatoires peuvent prendre plus de deux valeurs. Au lieu de valeurs, pour faire plus chic, on parlera de modalités et, si X est une variable aléatoire, on notera $|X|$ le nombre de ses modalités. Par exemple, $|\text{«résultat d'un jet de dé»}| = 6$.

Sur ces mots, monsieur NASA quitta satisfait le mathématicien en se disant qu'après tout, rentrer seulement des chiffres pour chaque événement élémentaire ne devrait pas lui demander trop de temps, et qu'ensuite il pourrait facilement calculer les probabilités qui l'intéressaient. Il essaya donc de déterminer tous les éléments de Ω pour son problème de navette. Et là, quelle ne fut pas sa surprise de s'apercevoir qu'il ne pouvait tous les déterminer parce qu'il y en avait beaucoup trop. En effet, les pannes suivantes peuvent se produire simultanément : fuites des réservoirs d'oxygène, d'hélium, d'azote, de fuel, problèmes de 2 valves d'admission, de pressions dans 4 réservoirs, de trop pleins... Autrement dit, Ω devait être constitué de l'ensemble des n-uplets de toutes les pannes possibles. Supposons que ces pannes soient évaluées sur une échelle de 0 à 9 en fonction de leur gravité. Dans ce cas, Ω doit avoir 10^{11} =100 milliards d'éléments. Voyant cela, monsieur NASA retourna voir le mathématicien.
– Dites, je me suis livré à un petit calcul qui m'a laissé entrevoir que Ω devait avoir beaucoup trop d'éléments pour être utilisable par mes ordinateurs.
– Ah, moi je n'y peux rien. Ma théorie est au point. Ce n'est pas de ma faute si votre ordinateur n'est pas assez puissant. Allez demander conseil auprès de l'informaticien.

2.1.2 Monsieur NASA découvre les réseaux bayésiens

Monsieur NASA alla voir l'informaticien et lui exposa son problème. Celui-ci lui répondit en substance :
– J'ai justement l'outil qu'il vous faut. Ça s'appelle un réseau bayésien. Reprenons votre exemple du jet des 2 dés. Selon le mathématicien, $\Omega = \{(x, y), 1 \leq x, y \leq 6\}$ a 36 éléments, et vous devez affecter à chacun de ces éléments une probabilité. Appelons X la variable aléatoire représentant le résultat du premier jet et Y celui du deuxième jet. X et Y peuvent prendre chacun 6 valeurs. Remarquez que les résultats des deux jets de dés ne dépendent pas l'un de l'autre. Dans ce cas, on dit que les variables aléatoires X et Y sont indépendantes. Elles vérifient alors la propriété suivante :

Définition 2 (Indépendance (marginale)) *Deux variables aléatoires X et Y sont **indépendantes** si $P(X = x, Y = y) = P(X = x) \times P(Y = y)\ \forall x, y$.*

Autrement dit, il est inutile de stocker les probabilités des 36 événements élémentaires, il suffit de stocker les probabilités suivantes :

$$\begin{array}{cccccc} P(X=1) & P(X=2) & P(X=3) & P(X=4) & P(X=5) & P(X=6) \\ P(Y=1) & P(Y=2) & P(Y=3) & P(Y=4) & P(Y=5) & P(Y=6) \end{array}$$

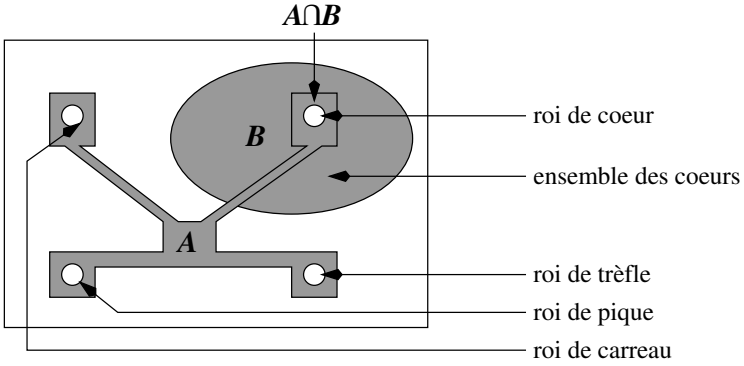
et d'utiliser la formule de la définition 2 pour recalculer toutes les probabilités $P(X = x, Y = y)$ qui vous intéressent.
– Oui, mais pour mon problème de navette, les variables aléatoires «fuites des réservoirs d'oxygène», «d'hélium», «d'azote», «de fuel», «problèmes de valves d'admission», «de pressions dans les réservoirs», «de trop pleins»... ne sont pas indépendantes les unes des autres. Je ne peux donc pas appliquer l'indépendance !
– C'est vrai, dans la pratique, il est assez rare d'avoir des variables totalement indépendantes, c'est pourquoi une propriété moins forte a été introduite par les mathématiciens : l'indépendance conditionnelle. Avant de la montrer, je dois vous expliquer le concept de probabilité conditionnelle :

Définition 3 (Probabilité conditionnelle) *La probabilité d'un événement A **conditionnellement** à un événement B, que l'on note $P(A|B)$, est la probabilité que A se produise sachant que B s'est produit.*

Ce qui est intéressant, c'est que $P(A|B)$ peut se calculer très facilement. D'abord, remarquez que $P(A|\Omega) = P(A)$. En effet, $P(A|\Omega)$ représente la probabilité que A soit réalisé conditionnellement au fait que l'événement certain le soit aussi. Or on sait que cette condition est toujours vérifiée puisque $P(\Omega) = 1$. Donc conditionner

par Ω ne diminue en rien ou n'augmente en rien les chances que A a de se réaliser. L'exemple suivant va vous permettre de comprendre comment $P(A|B)$ se calcule pour un événement B quelconque : tirons au hasard une carte parmi un jeu de 32 cartes. $\Omega = \{32 \text{ cartes}\}$. Considérons les événements $A =$ tirer un roi et $B =$ tirer un coeur. On veut calculer $P(A|B)$, c'est-à-dire la probabilité de tirer un roi sachant que l'on a tiré un coeur.

Si A se produit, sachant que B s'est aussi produit, cela signifie que $A \cap B$ se produit. Autrement dit, $P(A|B) = P(A \cap B|B)$, ou encore la probabilité de tirer un roi sachant que l'on a tiré un coeur est égale à la probabilité de tirer le roi de coeur sachant que l'on a tiré un coeur. Cela se voit bien sur la figure ci-après : on cherche la probabilité que l'événement A (c'est-à-dire l'un des quatre cercles) soit réalisé, sachant que l'événement B l'est (c'est-à-dire que l'on est dans l'ellipse). Il n'y a donc qu'un seul cercle qui peut être réalisé, et celui-ci correspond à $A \cap B$.



Cette petite manipulation est très intéressante. En effet, on cherche la probabilité que le cercle $A \cap B$, qui est inclus dans B, soit réalisé, sachant que l'événement B est réalisé, c'est-à-dire que $P(B) = 1$. Cela suggère de ne plus se placer dans l'univers (l'ensemble des événements élémentaires) Ω, mais plutôt dans l'univers B, et, en se servant de la remarque ci-dessus ($P(A|\Omega) = P(A)$), de calculer dans cet univers $P(A \cap B|B) = P(A \cap B)$. Là, on peut appliquer une formule classique des probabilités :

$$P(A \cap B) = \frac{\text{nombre de rois de coeur}}{\text{nombre de cartes dans } B} = \frac{\text{Card}(A \cap B)}{\text{Card}(B)}.$$

On peut alors en déduire :

$$P(A|B) = \frac{\text{Card}(A \cap B)}{\text{Card}(\Omega)} \times \frac{\text{Card}(\Omega)}{\text{Card}(B)} = \frac{P(A \cap B)}{P(B)}.$$

Cela donne lieu au théorème suivant :

Théorème 1 (Probabilité conditionnelle) *Soient A et B deux événements. Si $P(B) > 0$ alors*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Revenons à ce qui nous intéresse : l'indépendance conditionnelle :

Définition 4 (indépendance conditionnelle) *Soient X_1, X_2, X_3 des ensembles de variables aléatoires. X_1 et X_2 sont **indépendants conditionnellement** à X_3 si les propriétés suivantes (qui sont équivalentes) sont vérifiées :*

1. $P(X_1 = x_1|X_2 = x_2, X_3 = x_3) = P(X_1 = x_1|X_3 = x_3) \quad \forall x_1, x_2, x_3,$
2. $P(X_1 = x_1, X_2 = x_2|X_3 = x_3) = P(X_1 = x_1|X_3 = x_3) \times P(X_2 = x_2|X_3 = x_3) \quad \forall x_1, x_2, x_3.$

La barre verticale sépare les variables conditionnées des variables conditionnantes. Ainsi $P(X_1 = x_1|X_2 = x_2, X_3 = x_3) = P(X_1 = x_1|(X_2 = x_2, X_3 = x_3))$ et $P(X_1 = x_1, X_2 = x_2|X_3 = x_3) = P((X_1 = x_1, X_2 = x_2)|X_3 = x_3)$.

Traduction de la propriété 1 : $P(X_1 = x_1|X_3 = x_3)$ représente le nombre de chances que X_1 prenne la valeur x_1 sachant l'information que X_3 a pris la valeur x_3. $P(X_1 = x_1|X_2 = x_2, X_3 = x_3)$ représente le nombre de chances que X_1 prenne la valeur x_1 connaissant les informations X_3 a pris la valeur x_3 et X_2 a pris la valeur x_2. Autrement dit, la propriété 1 traduit simplement le fait que la connaissance de la valeur prise par X_2 n'apporte aucune information sur la valeur de X_1 quand on connaît déjà la valeur de X_3.

La propriété 2 traduit, elle, le fait que, connaissant la valeur de X_3, X_1 et X_2 sont indépendantes.
Notez que deux variables X_1, X_2 peuvent très bien être indépendantes conditionnellement à une troisième, et ne pas être indépendantes marginalement. Exemple : en général, tousser provoque des maux de tête. Donc $X_1 =$ «tousser» et $X_2 =$ «mal de tête» ne sont pas indépendants marginalement. Par contre, sachant que vous avez attrapé la grippe (X_3), les variables X_1 et X_2 deviennent indépendantes car toux et céphalées sont deux symptômes du virus, qui peuvent apparaître séparément.
Autre illustration de l'indépendance conditionnelle : certains immeubles sont dotés d'alarmes incendie. Vous admettrez sans trop de difficulté qu'il existe un lien entre le déclenchement de ces alarmes et le fait qu'il y ait réellement un incendie dans l'immeuble. Donc les variables aléatoires «incendie» et «alarme» ne sont pas indépendantes marginalement. De même, vous conviendrez que les variables «incendie» et «fumée» ne sont pas non plus indépendantes, de même que «fumée» et «alarme» puisqu'en principe ces dernières se déclenchent lorsqu'il y a un certain taux de fumée dans l'atmosphère. Par contre, à partir du moment où l'appareil a détecté de la fumée, il se déclenche, qu'il y ait ou non un incendie. Donc «incendie» et «alarme» sont indépendants conditionnellement à la variable «fumée».
Maintenant, quel est l'intérêt de l'indépendance conditionnelle? Eh bien, elle permet de réduire la place nécessaire au stockage des probabilités. En effet, si $X_1, \ldots, X_n$ sont des variables aléatoires, alors on peut démontrer aisément par récurrence grâce au théorème 1 ($P(A, B) = P(A|B) \times P(B)$) que :

$$P(X_1, \ldots, X_n) = P(X_1) \times P(X_2|X_1) \times P(X_3|X_1, X_2) \times \cdots \times P(X_n|X_1, \ldots, X_{n-1}). \quad (1)$$

L'équation ci-dessus devient particulièrement intéressante lorsqu'on l'utilise conjointement avec la propriété 1 de la définition 4 (cf. page 4) : si $\{i_1, \ldots, i_k\}$ et $\{i_{k+1}, \ldots, i_{j-1}\}$ forment une partition de $\{1, \ldots, j-1\}$ et si X_j est indépendant de $X_{i_{k+1}}, \ldots, X_{i_{j-1}}$ conditionnellement à $X_{i_1}, \ldots, X_{i_k}$, alors $P(X_j|X_1, \ldots, X_{j-1}) = P(X_j|X_{i_1}, \ldots, X_{i_k})$. L'équation (1) peut alors se simplifier énormément, et, d'une manière générale, le nombre de probabilités conditionnelles à rentrer dans l'ordinateur (et à stocker en mémoire) est souvent très inférieur à celui qu'on aurait dû rentrer si l'on avait suivi le mathématicien. Cela permet de traiter des problèmes qui, *a priori*, devraient être impossibles à traiter.
L'idée des réseaux bayésiens consiste donc :

1. à recenser toutes les variables aléatoires. Prenons un exemple : la dyspnée, une maladie respiratoire, peut être engendrée par une tuberculose, un cancer des poumons, une bronchite, par plusieurs de ces maladies, ou bien par aucune. Un séjour récent en Asie augmente les chances de tuberculose, tandis que fumer augmente les risques de cancer des poumons. Un patient se rend à l'hopital parce qu'il éprouve des difficultés à respirer. Dans quelle mesure peut-on dire qu'il est atteint de dyspnée? Les variables aléatoires de ce problème sont donc «dyspnée» (le patient est-il atteint de dyspnée? oui/non), «tuberculose» (a-t-il la tuberculose? oui/non), «cancer des poumons» (oui/non), «bronchite» (oui/non), «séjour en Asie» (oui/non), «fumer» (oui/non).
2. à choisir un ordre pour ces variables. Prenons par exemple $X_1 =$ «séjour en Asie», $X_2 =$ «fumer», $X_3 =$ «tuberculose», $X_4 =$ «cancer des poumons», $X_5 =$ «bronchite», $X_6 =$ «dyspnée». En fait, on peut ranger les variables dans n'importe quel ordre. Cependant, certains ordres sont plus avantageux que d'autres parce qu'ils vont permettre plus de simplifications des probabilités conditionnelles. En règle générale, on obtient plus de simplifications lorsque l'on place les variables correspondant à des causes avant celles correspondant aux conséquences (par exemple, on a intérêt à placer «fumer» avant «cancer des poumons»).
3. à simplifier les probabilités conditionnelles. On ne peut guère simplifier $P(X_1)$. *A priori*, on peut considérer que le fait d'avoir séjourné en Asie est indépendant du fait que l'on fume (quoique...). Donc $P(X_2|X_1) = P(X_2)$. Il n'y a aucun lien entre le fait de fumer et celui d'avoir la tuberculose; par contre il y en a un avec un éventuel séjour en Asie. Donc $P(X_3|X_1, X_2) = P(X_3|X_1)$. Un cancer du poumon n'a pas de rapport avec un séjour en Asie, ni avec la tuberculose. Donc $P(X_4|X_1, X_2, X_3) = P(X_4|X_2)$. De même avec la bronchite, et comme «fumer» ne provoque pas de bronchite, $P(X_5|X_1, X_2, X_3, X_4) = P(X_5)$. Enfin la dyspnée n'est provoquée que par la tuberculose, la bronchite et le cancer des poumons, donc $P(X_6|X_1, X_2, X_3, X_4, X_5) = P(X_6|X_3, X_4, X_5)$. On peut objecter que X_6 devrait aussi dépendre de X_2 puisque fumer augmente les risques de cancer du poumon. Mais sachant si l'on a (ou non) ce cancer, le fait de fumer ne modifie pas les chances d'avoir la dyspnée. Donc, connaissant X_3, X_4, et X_5, X_6 est indépendante de X_1 et de X_2. Ainsi, l'équation (1) se réduit-elle à :

$$P(X_1, X_2, X_3, X_4, X_5, X_6) = P(X_1) \times P(X_2) \times P(X_3|X_1) \times P(X_4|X_2) \times P(X_5) \times P(X_6|X_3, X_4, X_5). \quad (2)$$

Notez que si l'on avait saisi les probabilités des événements élémentaires comme le préconisait le

mathématicien, on aurait du saisir $2^6 = 64$ nombres, tandis qu'avec l'équation (2), on a besoin de saisir uniquement $2 + 2 + 4 + 4 + 2 + 16 = 30$ nombres.

4. à utiliser les formules de probabilités ci-dessus pour calculer les probabilités cherchées.

Afin de mieux visualiser les simplifications des probabilités conditionnelles décrites dans l'équation (2), on construit un graphe dans lequel les noeuds représentent les variables aléatoires et dans lequel on crée des arcs allant des variables conditionnantes vers les variables conditionnées : $P(X_6|X_3, X_4, X_5)$ sera ainsi représentée grâce aux arcs (X_3, X_6), (X_4, X_6) et (X_5, X_6). On obtient alors le graphe de la figure 2.

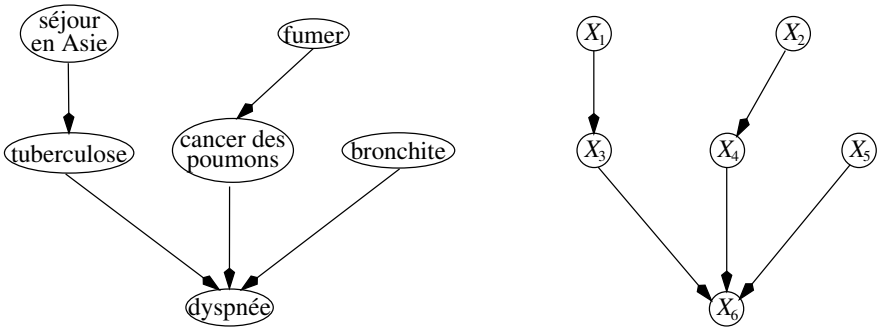


FIG. 2: Le réseau bayésien de la dyspnée

Conclusion : un réseau bayésien est constitué d'une part d'un graphe tel que celui ci-dessus et, d'autre part d'un ensemble de probabilités conditionnelles. Il est d'usage de considérer que chaque noeud stocke sa probabilité conditionnellement à ses parents.

2.2 Travail demandé dans la partie I : la saisie de la partie graphique du réseau

Toute l'interface utilisateur et les fonctions graphiques nécessaires à l'obtention de la figure 1, page 2, vous sont données (fichiers `graph.ml` et `interface.ml`). Ce que l'on vous demande dans cette partie, c'est de compléter le fichier `partie_1.ml` en réalisant les fonctions gérant la structure interne du réseau bayésien. La structure retenue pour modéliser le réseau dans cette partie du TP est la suivante :

```
(* === représentation d'un noeud === *)
type Graph_Node =
    {nom : string;    (* un identifiant du noeud *)
     x   : int ref;   (* abscisse du noeud dans la fenêtre camlgraph *)
     y   : int ref};; (* ordonnée du noeud *)

(* === représentation d'un arc === *)
type Graph_Arc =
    {head : string;   (* nom du noeud correspondant à la pointe de l'arc *)
     tail : string};; (* nom du noeud correspondant à la queue de l'arc *)
```

Un réseau sera représenté par une liste de noeuds de type `Graph_Node` et une liste d'arcs de type `Graph_Arc`. Cette structure est adéquate car elle permet de rajouter et de supprimer des noeuds et des arcs très simplement. Afin de gérer des erreurs éventuelles, les exceptions suivantes pourront être levées dans vos fonctions :

```
exception Multiply_Defined_Node of string;; (* === plusieurs noeuds ont le même nom  === *)
exception Unknown_Arc of string;;           (* === utilisation d'un arc non défini   === *)
exception Unknown_Node of string;;          (* === utilisation d'un noeud non défini === *)
exception Cycle_Detected of string;;        (* === le réseau contient des cycles     === *)
```

Vous pourrez retrouver toutes ces déclarations dans le fichier d'interface `partie_1.mli`.

Question 1 Définissez une fonction `add_node` de type `Graph_Node list -> string -> int -> int -> Graph_Node list` qui prend en arguments une liste de noeuds `liste_node`, le nom d'un noeud `name`, ainsi que ses coordonnées `x` et `y` dans la fenêtre CAMLGRAPH. Cette fonction teste s'il existe une noeud dans la liste `liste_node` ayant pour nom `name`. Le cas échéant, elle lève l'exception `Multiply_Defined_Node "add_node"`. Sinon, elle ajoute en tête de `liste_node` le noeud de nom `name` et renvoie cette nouvelle liste.

Question 2 Définissez une fonction `find_adjacent_arcs` de type `Graph_Node -> Graph_Arc list -> Graph_Arc list` qui prend en arguments un noeud `N` et une liste d'arcs `L`. Cette fonction renvoie la sous-liste de `L` constituée des arcs adjacents au noeud `N`, c'est-à-dire la liste des arcs dont l'une des extrémités correspond au noeud `N`.

Question 3 Définissez une fonction `find_non_adjacent_arcs` de type `Graph_Node -> Graph_Arc list -> Graph_Arc list` qui prend en arguments un noeud `N` et une liste d'arcs `L`. Cette fonction renvoie la sous-liste de `L` constituée des arcs non adjacents à `N`.

Question 4 Nous avons vu que les probabilités se calculent correctement en l'absence de cycles dans le réseau. Quand le réseau contient des cycles, ce n'est plus vrai. Il faut donc empêcher l'utilisateur de créer des cycles. Pour cela, vous allez définir une fonction `verif_cycle : Graph_Node list -> Graph_Arc list -> bool` qui prend en arguments une liste de noeuds `L_N` et une liste d'arcs `L_A`, et qui renvoie `false` s'il existe un cycle et `true sinon`.

On montre par récurrence que lorsqu'un graphe est sans cycles, soit il n'a aucun noeud, soit il contient un noeud ayant au plus un arc adjacent. Ce noeud ne peut alors faire partie d'un cycle. Pour détecter si le graphe possède des cycles on procède donc de la façon suivante :
On teste si `L_N` est vide, auquel cas le graphe passé en argument de la fonction est bien sans cycle. Dans le cas contraire, on détermine la liste `L_N_ext` des noeuds ayant au plus un arc adjacent et la liste `L_A_ext` des arcs adjacents à ces noeuds. Si `L_N_ext` $= \emptyset$, puisque `L_N` $\neq \emptyset$, (`L_N, L_A`) est un graphe comprenant au moins un noeud, mais ne comprenant aucun noeud ayant au plus un arc adjacent. Dans ce cas, (`L_N, L_A`) contient des cycles. Si, au contraire, `L_N_ext` $\neq \emptyset$, alors le graphe (`L_N, L_A`) est sans cycle si et seulement si le graphe (`L_N`$\backslash$`L_N_ext, L_A`$\backslash$`L_A_ext`), où `L`$\backslash$`M` représente l'ensemble `L` privé de `M`, l'est aussi. Il suffit donc de tester si ce dernier graphe possède des cycles.

Question 5 Définissez une fonction `add_arc` de type `Graph_Node list -> Graph_Arc list -> string -> string -> Graph_Arc list` qui prend en arguments une liste de noeuds, une liste d'arcs `liste_arc`, ainsi que les noms des deux extrémités d'un nouvel arc. Cette fonction teste si l'ajout du nouvel arc crée un cycle. Le cas échéant, elle lève une exception `Cycle_Detected "add_arc"`. Sinon, elle ajoute en tête de `liste_arc` le nouvel arc et elle renvoie cette nouvelle liste.

Question 6 Définissez une fonction `remove_arc : Graph_Arc list -> string -> string -> Graph_Arc list` qui prend en arguments une liste d'arcs `liste_arc`, le nom `N1` du noeud en queue d'un arc, et le nom `N2` à la tête de l'arc. Cette fonction recherche l'arc (`N1,N2`) dans `liste_arc`. Si elle ne le trouve pas, elle lève l'exception `Unknown_Arc "remove_arc"`; sinon, elle renvoie la liste `liste_arc` privée de l'arc (`N1,N2`).

Question 7 Définissez une fonction `remove_node : Graph_Node list -> Graph_Arc list -> string -> Graph_Node list * Graph_Arc list` qui prend en arguments une liste de noeuds, une liste d'arcs et le nom d'un noeud à supprimer du graphe. Si le noeud n'appartient pas à la liste de noeuds, la fonction lève l'exception `Unknown_Node "remove_node"`. Sinon, elle renvoie un couple formée de la liste de noeuds privée du noeud à supprimer, et de la liste des arcs non adjacents à ce noeud.

Question 8 Définissez une fonction `get_node : Graph_Node list -> string -> Graph_Node` qui prend en arguments une liste de noeuds et le nom d'un noeud. La fonction recherche si la liste contient un noeud de même nom. Le cas échéant, elle renvoie ce noeud. Sinon, elle lève l'exception `Unknown_Node "get_node"`.

Question 9 Définissez une fonction `get_arc : Graph_Arc list -> string -> string -> Graph_Arc` qui prend en arguments une liste d'arcs, et les noms des extrémités d'un arc (on ne sait pas si la tête ou la queue de l'arc correspond au premier nom ou au deuxième). La fonction recherche si la liste contient un arc avec des extrémités de mêmes noms. Le cas échéant, elle renvoie cet arc. Sinon, elle lève l'exception `Unknown_Arc "get_arc"`.

Vous pouvez tester vos fonctions grâce à la fonction `partie_1()` qui se trouve dans le fichier `interface.ml`. Si vous voulez obtenir un exécutable de la partie 1, tapez la commande :

```
camlc -custom -o partie_1 unix.zo graphics.zo partie_1.ml graph.zo interface.zo
      -lgraph -lunix -lX11
```

3 Partie II du TP

Le but de cette partie est, d'une part, de permettre à l'utilisateur de rentrer les probabilités de chaque noeud du réseau bayésien conditionnellement à ses parents et, d'autre part, d'adapter la structure de données

représentant le réseau afin de faciliter les calculs de la partie III. Dans la partie II, vous interrogerez l'utilisateur, non plus dans la fenêtre CAMLGRAPH, mais dans le shell qui vous a servi à lancer votre fenêtre (ou bien dans le toplevel si vous travaillez en interactif). À l'issue de cette partie, vous devriez obtenir un écran similaire à celui de la figure 2 : l'ordinateur vous demande de saisir toutes les probabilités dont il a besoin.

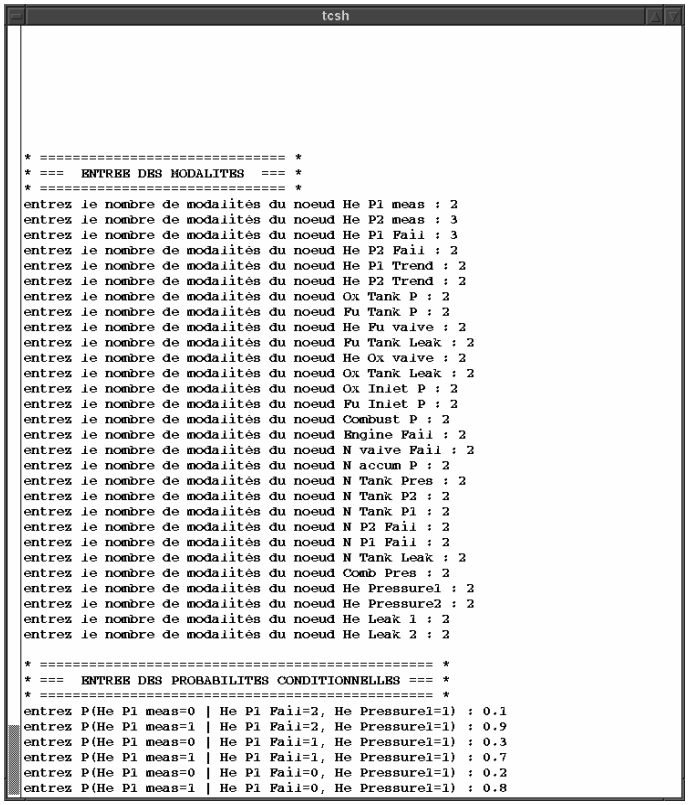


FIG. 3: Aspect de l'écran à la fin de la partie II.

3.1 Choix de la structure de données

Dans la partie I, il était assez pratique d'utiliser des listes de noeuds et d'arcs pour représenter la partie graphique du réseau bayésien car cela permettait de rajouter ou de supprimer des noeuds très simplement. Lorsque l'on aborde la partie II, le graphe est défini une fois pour toutes, on ne le modifiera plus. Pour accélérer les calculs, il devient alors plus intéressant d'utiliser des vecteurs que des listes. C'est pourquoi nous représenterons le réseau bayésien comme un vecteur de noeuds.

Le point épineux consiste maintenant à trouver une structure pour représenter les probabilités conditionnelles stockées dans chaque noeud du réseau. Si nous devons stocker $P(A|B)$, il nous faudra sauvegarder un nombre $(P(A = a_i|B = b_j))$ pour chaque valeur de A et pour chaque valeur de B. Cela fait furieusement penser à utiliser une matrice. Les noeuds pouvant avoir plusieurs parents, il nous faudra des «matrices» à plus de 2 dimensions : on appelle **hypermatrice** une «matrice» ayant un nombre de dimensions quelconque. En particulier, une hypermatrice de dimension 1 s'appelle un vecteur, une hypermatrice de dimension 2 s'appelle une matrice.

Nous allons donc définir un type hypermatrice pour représenter les probabilités conditionnelles. L'idée consiste à dire que si une matrice peut être représentée comme un vecteur de vecteurs, une hypermatrice peut l'être comme un vecteur de vecteurs de vecteurs de vecteurs... Il nous faut toutefois un mécanisme pour repérer à quel noeud correspond chacune des dimensions. Puisque le réseau bayésien est représenté comme un vecteur de noeuds, il suffit de préciser pour chacune des dimensions de l'hypermatrice l'indice du noeud correspondant dans le vecteur représentant le réseau. On obtient alors le type suivant :

```
type Hypermatrice = Vect of int * float vect | Mat of int * Hypermatrice vect;;
```

où le type `int` correspond à un indice de noeud dans le vecteur de noeuds représentant le réseau.
Exemple : Considérons deux noeuds A et B, ayant respectivement pour indices 2 et 10. A peut prendre 5 valeurs, $0, 1, 2, 3, 4$, et B peut en prendre 3 : $0, 1, 2$. Supposons que l'on veuille stocker la probabilité conditionnelle $P(A|B)$

suivante :

$$\begin{array}{lll} P(A=0|B=0)=0.2 & P(A=0|B=1)=0.3 & P(A=0|B=2)=0.1 \\ P(A=1|B=0)=0.4 & P(A=1|B=1)=0.2 & P(A=1|B=2)=0.1 \\ P(A=2|B=0)=0.1 & P(A=2|B=1)=0.3 & P(A=2|B=2)=0.2 \\ P(A=3|B=0)=0.2 & P(A=3|B=1)=0.1 & P(A=3|B=2)=0.1 \\ P(A=4|B=0)=0.1 & P(A=4|B=1)=0.1 & P(A=4|B=2)=0.5 \end{array}$$

Alors on peut représenter $P(A|B)$ par :

```
Mat (2, [| Vect (10, [|0.2; 0.3; 0.1|]);
           Vect (10, [|0.1; 0.3; 0.2|]);
           Vect (10, [|0.2; 0.1; 0.1|]);
           Vect (10, [|0.1; 0.1; 0.5|]) |]);;
```

ou bien encore par :

```
Mat (10, [| Vect (2, [|0.2; 0.4; 0.1; 0.2; 0.1|]);
            Vect (2, [|0.3; 0.2; 0.3; 0.1; 0.1|]);
            Vect (2, [|0.1; 0.1; 0.2; 0.1; 0.5|]) |]);;
```

Nous verrons dans la partie III que pendant les calculs de probabilités, les noeuds reçoivent des messages de leurs voisins dans le graphe. La structure suivante va nous permettre de réaliser cela :

```
type Voisin =
    {voisin_num     : int;          (* indice du voisin dans le vecteur de noeuds *)
                                    (* représentant le réseau *)
     message        : float vect};; (* le message transmis par le voisin *)
```

La taille du message est égale au nombre de modalités du voisin si celui-ci est un parent, et est égale au nombre de modalité du noeud lui-même si le voisin est un enfant.

Enfin, voici la structure qui nous permet de manipuler les noeuds :

```
type Node =
   {node_num   : int;            (* indice du noeud dans le vecteur représentant le réseau *)
    node_name  : string;         (* nom du noeud *)
    node_x     : int;            (* abscisse du noeud dans la fenêtre camlgraph *)
    node_y     : int;            (* ordonnée du noeud dans la fenêtre camlgraph *)
    parent     : Voisin list;    (* liste des parents du noeud *)
    enfant     : Voisin list;    (* liste des enfants du noeud *)
    evidence   : float vect;     (* le vecteur d'evidence E_x (cf. partie III) *)
    proba_marg : float vect;     (* la probabilité a priori ou a posteriori du noeud *)
    proba_cond : Hypermatrice};; (* la matrice de probabilité conditionnelle *)
```

Les tailles des vecteurs `proba_marg` et `evidence` sont égales au nombre de modalités du noeud.

3.2 Travail demandé dans la Partie II : la saisie des matrices de probabilité conditionnelle

Dans cette partie, vous allez compléter le fichier `partie_2.ml`.

Question 10 Définissez une fonction `get_modalites : Graph_Node list -> int vect` qui prend en argument une liste de noeuds au format de la partie I. Cette fonction demande, pour chaque noeud de la liste, à l'utilisateur de saisir son nombre de modalités. Elle renvoie un vecteur contenant ces nombres dans l'ordre suivant : le $i^{\text{ème}}$ élément correspond à la modalité du $i^{\text{ème}}$ noeud de la liste passée en argument.

Question 11 Définissez une fonction `separate_orphans` de type `Graph_Node list -> Graph_Arc list -> Graph_Node list * Graph_Node list` qui prend en argument une liste de noeuds et une liste d'arcs aux formats de la partie I. Cette fonction renvoie un couple dont le premier élément est la liste des noeuds qui, d'après la liste d'arcs, n'ont pas de parents, et dont le deuxième élément est la liste des noeuds qui, eux, ont des parents.

Question 12 Définissez dans un style fonctionnel une fonction `get_node_num : Graph_Node list -> string -> int` qui recherche l'indice dans la liste du noeud dont le nom est passé en deuxième argument. La fonction retourne cet indice si elle trouve le noeud, et lève une exception `Unknown_Node "get_node_num"` sinon. On pose que le premier élément d'une liste correspond à l'indice 0.
Définissez, toujours dans le plus pur style fonctionnel, la fonction `get_node_num_in_vect : Node vect -> string -> int` qui fait exactement la même chose, mais en recherchant le noeud dans un vecteur de noeuds au nouveau format.

Question 13 Définissez la fonction `create_proba_cond : Graph_Node list -> Graph_Node -> string list -> int vect -> Hypermatrice` qui prend en arguments une liste de noeuds `liste_node`, un noeud `node`, la liste `liste_parents` des noms des parents de `node`, et un vecteur contenant les modalités de tous les noeuds du réseau. Cette fonction demande à l'utilisateur de saisir l'ensemble de l'hypermatrice de probabilité conditionnelle du noeud `node`. Elle renvoie cette hypermatrice.
Une hypermatrice est soit un vecteur de `float`, soit un vecteur d'hypermatrices. Lorsque vous créerez $[\![P(x_i|y_1,\ldots,y_n)]\!]_{|X||Y_1|\ldots|Y_n|}$, vous vous arrangerez pour que X corresponde à la dernière dimension de l'hypermatrice. Exemple : reconsidérons nos deux noeuds A et B, ayant respectivement pour indices 2 et 10, et leur probabilité conditionnelle $P(A|B)$:

$$\begin{array}{lll} P(A=0|B=0)=0.2 & P(A=0|B=1)=0.3 & P(A=0|B=2)=0.1 \\ P(A=1|B=0)=0.4 & P(A=1|B=1)=0.2 & P(A=1|B=2)=0.1 \\ P(A=2|B=0)=0.1 & P(A=2|B=1)=0.3 & P(A=2|B=2)=0.2 \\ P(A=3|B=0)=0.2 & P(A=3|B=1)=0.1 & P(A=3|B=2)=0.1 \\ P(A=4|B=0)=0.1 & P(A=4|B=1)=0.1 & P(A=4|B=2)=0.5 \end{array}$$

Alors $P(A|B)$ sera représenté par :

```
Mat (10, [| Vect (2, [|0.2; 0.4; 0.1; 0.2; 0.1|]);
            Vect (2, [|0.3; 0.2; 0.3; 0.1; 0.1|]);
            Vect (2, [|0.1; 0.1; 0.2; 0.1; 0.5|]) |]);;
```

Question 14 Définissez la fonction `make_graph : Graph_Node list -> Graph_Arc list -> Node vect` qui transforme un réseau bayésien stocké au format de la partie I, c'est-à-dire une liste de noeuds et une liste d'arcs, en un vecteur de noeuds au format `node`. On fera particulièrement attention à respecter les bonnes dimensions pour les vecteurs représentant les messages envoyés par les voisins, pour ceux représentant E_X, pour les probabilités marginales et conditionnelles. Ces vecteurs pourront être initialisés avec des 0.

4 Partie III du TP

Dans cette partie, vous allez effectuer des calculs de probabilités. À l'issue de la troisième partie, vous pourrez tester la fonction `partie_3()` qui vérifie si vos calculs sont exacts (cette fonction se trouve dans le fichier `fin.zo`). Le cas échéant, vous verrez sur la droite de la fenêtre CAMLGRAPH un réseau bayésien qui correspond à peu de choses près à celui utilisé par la NASA pour gérer les incidents du système de propulsion de la navette spatiale. Vous verrez alors les messages transiter dans le réseau, les flèches rouges symbolisant des messages et les flèches vertes des demandes de messages de certains noeuds à leurs voisins. Vous verrez de plus la navette spatiale s'élever normalement dans la partie gauche de l'écran (cf. figure 4). Si les calculs sont incorrects, CAML vous avertira que vous venez de sacrifier une navette spatiale.

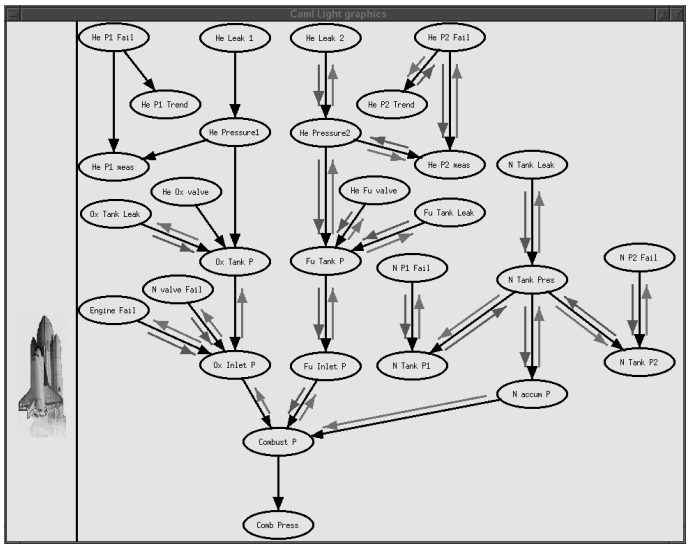


FIG. 4: Aspect de l'écran à la fin de la partie III.

La section ci-dessous explique en détails comment se déroulent les calculs dans le réseau bayésien. La lecture de la section 4.1.2 suffit pour répondre aux questions 15 à 20. Les autres sections, plus exactement les algorithmes 2 et 3, devraient vous permettre de comprendre l'algorithme de calcul de probabilité, ce qui vous permettra de répondre aux trois dernières questions de la partie III.

4.1 Monsieur NASA cauchemarde sur les calculs à effectuer dans le réseau

Monsieur NASA fut convaincu assez rapidement que les réseaux bayésiens pouvaient lui permettre de stocker toutes les informations nécessaires aux calculs de probabilités qu'il envisageait d'effectuer. Restait maintenant à se préoccuper desdits calculs...

4.1.1 Calcul des probabilités *a priori*

La première chose que l'on peut calculer, ce sont les probabilités marginales *a priori*, c'est-à-dire les $P(X_i)$. Reprenons l'exemple de la dyspnée. $P(X_1)$, $P(X_2)$ et $P(X_5)$ ont déjà été saisies dans l'ordinateur, donc pas besoin de calcul pour les déterminer. En généralisant, on peut dire que cela correspond aux probabilités des noeuds sans parents dans le graphe (les noeuds orphelins). En appliquant la propriété 3 de la définition 1,

$$P(X_3 = x_3) = \sum_{i=1}^{|X_1|} P(X_3 = x_3, X_1 = x_1^i),$$

et, en appliquant le théorème 1, on obtient :

$$P(X_3 = x_3) = \sum_{i=1}^{|X_1|} P(X_3 = x_3 | X_1 = x_1^i) \times P(X_1 = x_1^i).$$

De la même manière,

$$P(X_4 = x_4) = \sum_{i=1}^{|X_2|} P(X_4 = x_4 | X_2 = x_2^i) \times P(X_2 = x_2^i).$$

Enfin, pour calculer $P(X_6)$, on remarque que :

$$\begin{aligned} P(X_6 = x_6) &= \sum_{i=1}^{|X_3|} \sum_{j=1}^{|X_4|} \sum_{k=1}^{|X_5|} P(X_6 = x_6, X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k), \\ &= \sum_{i=1}^{|X_3|} \sum_{j=1}^{|X_4|} \sum_{k=1}^{|X_5|} P(X_6 = x_6 | X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k) \times P(X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k). \quad (3) \end{aligned}$$

Il reste maintenant à calculer $P(X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k)$. Pour cela, on reprend la propriété 3 de la définition 1 :

$$\begin{aligned} P(X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k) &= \sum_{a=1}^{|X_1|} \sum_{b=1}^{|X_2|} \sum_{c=1}^{|X_6|} P(X_1 = x_1^a, X_2 = x_2^b, X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k, X_6 = x_6^c), \\ &= \sum_{a=1}^{|X_1|} \sum_{b=1}^{|X_2|} \sum_{c=1}^{|X_6|} P(X_1 = x_1^a) \times P(X_2 = x_2^b) \times P(X_3 = x_3^i | X_1 = x_1^a) \times P(X_4 = x_4^j | X_2 = x_2^b) \\ &\qquad \times P(X_5 = x_5^k) \times P(X_6 = x_6^c | X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k) \end{aligned}$$

et on lui applique l'équation (2) :

$$\begin{aligned} P(X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k) &= \left[\sum_{a=1}^{|X_1|} P(X_3 = x_3^i | X_1 = x_1^a) \times P(X_1 = x_1^a) \right] \times \left[\sum_{b=1}^{|X_2|} P(X_4 = x_4^j | X_2 = x_2^b) \times P(X_2 = x_2^b) \right] \\ &\quad \times \left[\sum_{c=1}^{|X_6|} P(X_6 = x_6^c | X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k) \right] \times P(X_5 = x_5^k) \\ &= P(X_3 = x_3^i) \times P(X_4 = x_4^j) \times P(X_5 = x_5^k). \end{aligned}$$

Par conséquent, d'après ce qui précède et l'équation 3, on peut calculer $P(X_6 = x_6)$ de la manière suivante :

$$P(X_6 = x_6) = \sum_{i=1}^{|X_3|} \sum_{j=1}^{|X_4|} \sum_{k=1}^{|X_5|} P(X_6 = x_6 | X_3 = x_3^i, X_4 = x_4^j, X_5 = x_5^k) \times P(X_3 = x_3^i) \times P(X_4 = x_4^j) \times P(X_5 = x_5^k).$$

Les calculs que nous venons de mener nous permettent d'en déduire l'algorithme de calcul des probabilités a priori pour des graphes quelconques[1] :

Les probabilités des noeuds orphelins (sans parents dans le graphe) sont déjà calculées. Pour tous les noeuds X_i dont tous les parents $Y_1, \ldots, Y_n$ ont été calculés, $P(X_i = x_i)$ se calcule de la manière suivante :

$$P(X_i = x_i) = \sum_{j_1=1}^{|Y_1|} \sum_{j_2=1}^{|Y_2|} \cdots \sum_{j_n=1}^{|Y_n|} P(X_i = x_i | Y_1 = y_1^{j_1}, \ldots, Y_n = y_n^{j_n}) \times P(Y_1 = y_1^{j_1}) \times \cdots \times P(Y_n = y_n^{j_n}).$$

En l'absence de cycle dans le graphe, une récursion sur le nombre de noeuds non calculés montre que soit i) il existe un noeud dont tous les parents ont déjà été calculés, auquel cas on peut appliquer l'équation ci-dessus ; soit ii) tous les noeuds ont été calculés.

4.1.2 Les calculs selon une notation matricielle

Nous allons reprendre les calculs ci-dessus, mais en utilisant des opérations matricielles plutôt que des opérations sur des probabilités. Pour mieux comprendre ce qui suit, nous allons systématiquement indicer les vecteurs, les matrices et hypermatrices par leurs dimensions respectives. Ainsi V_n désignera un vecteur ayant n éléments, M_{pn} désignera une matrice ayant p lignes et n colonnes, soit $p \times n$ éléments. De manière analogue, $M_{i_1 i_2 \ldots i_p}$ représentera une hypermatrice ayant $i_1 \times i_2 \times \cdots \times i_p$ éléments. De plus, nous désignerons dans les calculs les vecteurs, matrices et hypermatrices par leurs termes génériques de la manière suivante : un vecteur $V_n = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$ sera noté $[\![v_i]\!]_n$, où v_i représente l'élément du vecteur sur la $i^{\text{ème}}$ ligne ; une matrice $M_{pn} = \begin{pmatrix} m_{11} & \ldots. & m_{1j} & \ldots. & m_{1n} \\ \vdots & & \vdots & & \vdots \\ m_{i1} & \ldots. & m_{ij} & \ldots. & m_{in} \\ \vdots & & \vdots & & \vdots \\ m_{p1} & \ldots. & m_{pj} & \ldots. & m_{pn} \end{pmatrix}$ sera notée $[\![m_{ij}]\!]_{pn}$, et une hypermatrice $M_{i_1 i_2 \ldots i_p}$ sera notée $[\![m_{j_1 j_2 \ldots j_p}]\!]_{i_1 i_2 \ldots i_p}$.

Somme de deux hypermatrices :

Soient $M_{i_1 i_2 \ldots i_p}$ et $N_{i_1 i_2 \ldots i_p}$ deux hypermatrices de mêmes dimensions. Alors la somme de ces deux hypermatrices est l'hypermatrice suivante : $M_{i_1 i_2 \ldots i_p} + N_{i_1 i_2 \ldots i_p} = [\![m_{j_1 j_2 \ldots j_p} + m_{j_1 j_2 \ldots j_p}]\!]_{i_1 i_2 \ldots i_p}$.

Produit d'une hypermatrice avec un scalaire :

Soient λ un nombre réel quelconque et $M_{np} = [\![m_{ij}]\!]_{np}$ une matrice quelconque. On sait que le produit de M_{np} par λ est égal à la matrice $\lambda M_{np} = [\![\lambda\ m_{ij}]\!]_{np}$. On généralise très simplement ce produit aux hypermatrices : soit $M_{i_1 i_2 \ldots i_p} = [\![m_{j_1 j_2 \ldots j_p}]\!]_{i_1 i_2 \ldots i_p}$, alors $\lambda M_{i_1 i_2 \ldots i_p} = [\![\lambda\ m_{j_1 j_2 \ldots j_p}]\!]_{i_1 i_2 \ldots i_p}$.

[1] En fait, si l'on pousse les calculs jusqu'au bout, cet algorithme ne fonctionne que pour des graphes **sans cycles**, c'est-à-dire dans lequel il n'existe pas d'ensemble de noeuds $\{X_{i_1}, \ldots, X_{i_k}\}$ tel que le graphe contienne soit l'arc (X_{i_1}, X_{i_k}) soit l'arc (X_{i_k}, X_{i_1}), et, pour tout $j < k$, soit $(X_{i_j}, X_{i_{j+1}})$ soit l'arc $(X_{i_{j+1}}, X_{i_j})$

Produit d'une hypermatrice avec un vecteur :

Soient $V_n = [\![v_n]\!]$ un vecteur de taille n, et $M_{pn} = [\![m_{ij}]\!]_{pn}$ une matrice. Notons $\otimes$ le produit en question. On sait que le produit de la matrice M_{pn} par le vecteur V_n est égal au vecteur

$$N_p = M_{pn} \otimes V_n = \begin{pmatrix} \sum_{i=1}^{n} v_j \times m_{1j} \\ \vdots \\ \sum_{i=1}^{n} v_j \times m_{pj} \end{pmatrix} = [\![\sum_{j=1}^{n} v_j \times m_{ij}]\!]_p.$$

On peut généraliser ce produit à des hypermatrices de la manière suivante : Soit $M_{i_1 i_2 \ldots i_p}$ une hypermatrice soit V_{i_k} un vecteur. Alors le produit de l'hypermatrice hypermatrice avec le vecteur est égal à $M_{i_1 i_2 \ldots i_p} \otimes V_{i_k} = [\![\sum_{i=1}^{i_k} v_i \times m_{j_1 j_2 \ldots j_{k-1} i j_{k+1} \ldots j_p}]\!]_{i_1 i_2 \ldots i_{k-1} i_{k+1} \ldots i_p}$.

Produit tensoriel (ou terme à terme) entre deux vecteurs :

Soient deux vecteurs $V_k = [\![v_i]\!]_k$ et $W_k = [\![w_j]\!]_k$. Alors le produit tensoriel de V_k et de W_k est égal à $V_k \odot W_k = [\![v_i \times w_i]\!]_k$.

4.1.3 Calcul des probabilités *a priori* :

Revenons maintenant aux calculs qui nous intéressent. Considérons un noeud X dont on connaît les probabilités *a priori* de ses parents, $Y_1, \ldots, Y_n$. On a vu que $P(X = x_j)$ se calcule en théorie de la manière suivante :

$$P(X = x_j) = \sum_{j_1=1}^{|Y_1|} \sum_{j_2=1}^{|Y_2|} \cdots \sum_{j_n=1}^{|Y_n|} P(X = x_j | Y_1 = y_1^{j_1}, Y_2 = y_2^{j_2}, \ldots, Y_n = y_n^{j_n}) \times P(Y_1 = y_1^{j_1}) \times \cdots \times P(Y_n = y_n^{j_n}).$$

En pratique, nous allons stocker dans l'ordinateur la probabilité conditionnelle $P(X = x_i | Y_1 = y_1^{j_1}, \ldots, Y_n = y_n^{j_n})$ pour l'ensemble des valeurs $x_i, y_1^{j_1}, \ldots, y_n^{j_n}$, grâce à une hypermatrice : $[\![P(x_i | y_1^{j_1}, \ldots, y_n^{j_n})]\!]_{|X||Y_1|\ldots|Y_n|}$. De la même manière, nous allons stocker les probabilités des parents sous forme de vecteurs $[\![P(y_k^{j_k})]\!]_{|Y_k|}$. D'après les formules qui précèdent, il est relativement évident que l'équation ci-dessus correspond à la ligne x_j du vecteur $[\![P(x_i)]\!]_{|X|}$ défini par :

$$[\![P(x_i)]\!]_{|X_i|} = \Big(\cdots \Big(\Big([\![P(x_i | y_1^{j_1}, \ldots, y_n^{j_n})]\!]_{|X||Y_1|\ldots|Y_n|} \otimes [\![P(y_1^{j_1})]\!]_{|Y_1|} \Big) \otimes [\![P(y_2^{j_2})]\!]_{|Y_2|} \Big) \cdots \Big) \otimes [\![P(y_n^{j_n})]\!]_{|Y_n|}.$$

Autrement dit, le calcul des probabilités *a priori* se ramène à l'algorithme suivant :

Algorithme 1 (Calcul des probabilités *a priori*) *Les probabilités des noeuds orphelins (sans parents dans le graphe) sont déjà calculées. Pour tous les noeuds X dont tous les parents $Y_1, \ldots, Y_n$ ont été calculés, $[\![P(X = x_i)]\!]_{|X|}$ se calcule de la manière suivante :*

$$[\![P(x_i)]\!]_{|X|} = \Big(\cdots \Big(\Big([\![P(x_i | y_1^{j_1}, \ldots, y_n^{j_n})]\!]_{|X||Y_1|\ldots|Y_n|} \otimes [\![P(y_1^{j_1})]\!]_{|Y_1|} \Big) \otimes [\![P(y_2^{j_2})]\!]_{|Y_2|} \Big) \cdots \Big) \otimes [\![P(y_n^{j_n})]\!]_{|Y_n|}.$$

En l'absence de cycle dans le graphe, soit i) il existe un noeud dont tous les parents ont déjà été calculés, auquel cas on peut appliquer l'équation ci-dessus ; soit ii) tous les noeuds ont été calculés.

4.1.4 Calcul des probabilités *a posteriori*

L'intérêt des réseaux bayésiens ne réside pas seulement dans le calcul des probabilités a priori ; heureusement, sinon nous aurions développé un outil bien compliqué pour l'utilité qu'il procurerait ! Non, l'intérêt principal des réseaux bayésiens réside dans le calcul de probabilités *a posteriori*. Reprenons l'exemple de la dyspnée. Que nous indiquent les probabilités *a priori* ? Eh bien tout simplement la probabilité qu'une personne choisie au hasard dans la population fume ($P(X_2)$), ou bien qu'elle soit allée en Asie ($P(X_1)$), ou bien encore la probabilité qu'une personne choisie au hasard ait une dyspnée ($P(X_6)$). Cependant, ce n'est pas ce qui intéresse le médecin pour faire son diagnostic : lui sait que son patient fume et qu'il n'a jamais séjourné en Asie. Donc ce qui l'intéresse, c'est la probabilité que le patient ait une dyspnée sachant ces informations, autrement dit $P(X_6 = \text{«oui»} | X_1 = \text{«non»}, X_2 = \text{«oui»})$. Une probabilité conditionnée par des informations (des observations) s'appelle une probabilité *a posteriori*. Grâce à un exemple simple et des équations aux dimensions, nous allons déduire comment l'on peut la calculer.

Lorsque les observations proviennent d'une racine

Considérons donc le réseau de la figure 5. On apprend par ailleurs que T, qui *a priori* pouvait prendre les

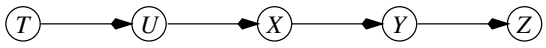


FIG. 5: Exemple de réseau bayésien

valeurs $t_1, t_2, \ldots, t_{|T|}$ ne peut plus prendre que deux valeurs t_1 et t_2. Appelons e cette information (e comme *evidence* en anglais). Quelle est maintenant la probabilité de chacune des variables aléatoires connaissant e ? Autrement dit, que valent $P(T|e)$, $P(U|e)$, $P(X|e)$, $P(Y|e)$, $P(Z|e)$?
Tout d'abord, il faut calculer $P(T = t_i|e)$ pour tout i. Bien évidemment, si $i \neq 1, 2$, $P(T = t_i|e) = 0$. Supposons que T représente le couple (âge,sexe) des étudiants dans un groupe de TD. Si t_0, t_1, t_2, t_3 correspondent respectivement aux couples (âge≤21 ans, masculin), (âge>21 ans, masculin), (âge≤21 ans, féminin), (âge>21 ans, féminin), et si en moyenne les groupes contiennent 23 hommes, dont 14 ont plus de 21 ans, et 7 femmes dont 3 ont plus de 21 ans, alors :

$$P(T = t_1) = \frac{9}{30}, \quad P(T = t_2) = \frac{14}{30}, \quad P(T = t_3) = \frac{4}{30}, \quad P(T = t_4) = \frac{3}{30}.$$

On stocke sur ordinateur la probabilité de T grâce au vecteur $[\![P(t_i)]\!]_{|T|} = (\frac{9}{30}, \frac{14}{30}, \frac{4}{30}, \frac{3}{30})$. Maintenant, on sélectionne un groupe et l'on s'aperçoit qu'il n'y a aucune femme dans ce groupe. Dans ce cas, on peut dire qu'en sélectionnant un individu au hasard dans le groupe, on a 14 chances sur 23 qu'il ait plus de 21 ans, et 9 chances sur 23 qu'il ait moins de 21 ans. Ainsi,

$$P(T = t_1|e) = \frac{9}{23}, \quad P(T = t_2|e) = \frac{14}{23}, \quad P(T = t_3|e) = 0, \quad P(T = t_4|e) = 0.$$

Notons que $\sum_{i=1}^{4} P(T = t_i|e) = 1$. Créons un vecteur représentant l'information sur T, $E_{|T|} = (1, 1, 0, 0)$, où les 1 représentent les valeurs t_i que peut prendre T et les 0 représentent les valeurs que T ne peut plus prendre, et effectuons le produit tensoriel de $[\![P(t_i)]\!]_{|T|}$ avec $E_{|T|}$. On obtient alors le vecteur $[\![P(t_i)]\!]_{|T|} \odot E_{|T|} = (\frac{9}{30}, \frac{14}{30}, 0, 0)$, qui n'est autre que le vecteur représentant $P(T|e)$ à un coefficient multiplicatif près. Ce dernier est visiblement égal à $\frac{30}{23}$, ce qui correspond en fait à $\frac{1}{P(e)}$ puisqu'on a 23 chances sur 30 de sélectionner un homme lorsque l'on choisit un individu au hasard dans un groupe choisi lui aussi au hasard. Cependant, nous calculerons ce coefficient de la manière suivante : on sait que $\sum_{i=1}^{4} P(T = t_i|e) = 1$, donc obligatoirement, le coefficient multiplicatif est l'inverse de la somme des éléments du vecteur $[\![P(t_i)]\!]_{|T|} \odot E_{|T|}$. Sous forme matricielle, cette somme peut s'écrire de la manière suivante : $\left([\![P(t_i)]\!]_{|T|} \odot E_{|T|}\right) \otimes \mathbb{1}_{|T|}$, où $\mathbb{1}_{|T|}$ désigne le vecteur de taille $|T|$ composé uniquement de 1. Donc

$$[\![P(t_i|e)]\!]_{|T|} = \frac{1}{P(e)} \left([\![P(t_i)]\!]_{|T|} \odot E_{|T|}\right) \quad \text{et} \quad P(e) = \left([\![P(t_i)]\!]_{|T|} \odot E_{|T|}\right) \otimes \mathbb{1}_{|T|}.$$

En principe, on ne calcule jamais le coefficient multiplicatif $P(e)$ car les calculs restent cohérents quels que soient ces coefficients. Lorsqu'on a besoin de montrer à l'utilisateur les probabilités calculées, il suffit de se rappeler que $\sum_i P(A = a_i|e) = 1$, quelle que soit la variable A.

Calculons maintenant $P(U|e)$.

$$P(U = u|e) = \frac{P(U = u, e)}{P(e)} = \frac{\sum_{i=1}^{4} P(U = u, e|T = t_i) \times P(T = t_i)}{P(e)}.$$

Notons que $P(U = u, e|T = t_3) = P(U = u, e|T = t_4) = 0$ puisque ce sont les probabilités que $U = u$ et $T = t_1$ ou t_2 sachant que $T = t_3$ ou t_4. De même $P(T = t_3, e) = P(T = t_4, e) = 0$. Par contre, $P(U = u, e|T = t_1) = P(U = u, e|T = t_2) = P(U = u|T = t_2)$, $P(T = t_1, e) = P(T = t_1)$, et $P(T = t_2, e) = P(T = t_2)$. Donc

$$P(U = u|e) = \frac{\sum_{i=1}^{4} P(U = u|T = t_i) \times P(T = t_i, e)}{P(e)} = \sum_{i=1}^{4} P(U = u|T = t_i) \times P(T = t_i|e).$$

Autrement dit, pour calculer $P(U = u|e)$, il suffit de faire le produit scalaire de la matrice $[\![P(u_i|t_j)]\!]_{|U||T|}$ avec le vecteur $[\![P(t_j|e)]\!]_{|T|}$. De la même manière, on voit bien que $[\![P(x_i|e)]\!]_{|X|} = [\![P(x_i|u_j)]\!]_{|X||U|} \otimes [\![P(u_j|e)]\!]_{|U|}$. Par récurrence,

Supposons que le réseau bayésien soit une chaîne et que l'information e provienne de la racine de cette chaîne. Soit B un noeud quelconque de la chaîne (excepté la racine), et soit A son père.
Alors $[\![P(b_i|e)]\!]_{|B|} = [\![P(b_i|a_j)]\!]_{|B||A|} \otimes [\![P(a_j|e)]\!]_{|A|}$.

D'une manière plus visuelle (cf. figure 6), voilà comment se passe la propagation de l'information $T = t_1$ ou t_2 dans le réseau :

1. T reçoit l'information, sous forme d'un vecteur $E_{|T|}$, qu'il ne peut plus prendre que certaines valeurs.
2. T met alors à jour sa probabilité : $[\![P(t_i|e)]\!]_{|T|} = \frac{1}{P(e)} \left([\![P(t_i)]\!]_{|T|} \odot E_{|T|} \right)$. Il envoie alors à son fils un message $\pi_{TU} = [\![P(t_i|e)]\!]_{|T|}$.
3. Lorsque U reçoit le message envoyé par T, il met à jour sa propre probabilité a posteriori : $[\![P(u_i|e)]\!]_{|U|} = [\![P(u_i|t_j)]\!]_{|U||T|} \otimes \pi_{TU}$. Il envoie ensuite à son fils un message $\pi_{UX} = [\![P(u_i|e)]\!]_{|U|}$.
4. lorsque X reçoit le message, il met à jour sa probabilité et émet un message π_{XY}, et ainsi de suite...

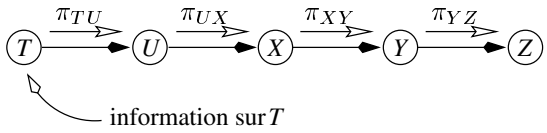


FIG. 6: Les messages à envoyer pour calculer les probabilités *a posteriori*

Lorsque les observations proviennent d'une feuille

Supposons maintenant que l'information e ne soit plus en T mais en Z : on apprend que Z peut seulement prendre les valeurs z_1 et z_2 au lieu des 4 valeurs z_1, z_2, z_3, z_4. On connaît déjà $P(Z)$, cette probabilité ayant été calculée dans les sections 4.1.1 et 4.1.2. De la même manière que l'on avait calculé $P(T|e)$, on peut calculer $P(Z|e)$:

$$[\![P(z_i|e)]\!]_{|Z|} = \frac{1}{P(e)} \left([\![P(z_i)]\!]_{|Z|} \odot E_{|Z|} \right) \quad \text{et} \quad P(e) = \left([\![P(z_i)]\!]_{|Z|} \odot E_{|Z|} \right) \otimes \mathbb{1}_{|Z|}.$$

Pour calculer $P(Y|e)$, nous allons utiliser le théorème de Bayes :

Théorème 2 (Théorème de Bayes) *Soient A et B deux variables aléatoires telles que $P(A) \neq 0$ et $P(B) \neq 0$. Alors :*

$$P(B|A) = P(A|B) \times \frac{P(B)}{P(A)}. \tag{4}$$

Appliquons le théorème :

$$P(Y|e) = P(e|Y) \times \frac{P(Y)}{P(e)}.$$

Maintenant $P(e|Y) = P(Z = z_1 \text{ ou } z_2|Y) = P(Z = z_1|Y) + P(Z = z_2|Y)$. En termes matriciels, on obtient :

$$[\![P(e|y_i)]\!]_{|Y|} = [\![P(z_j|y_i)]\!]_{|Z||Y|} \otimes E_{|Z|}.$$

Par conséquent,

$$[\![P(y_i|e)]\!]_{|Y|} = \frac{1}{P(e)} \left(\left([\![P(z_j|y_i)]\!]_{|Z||Y|} \otimes E_{|Z|} \right) \odot [\![P(y_i)]\!]_{|Y|} \right).$$

De la même manière,

$$P(X|e) = P(e|X) \times \frac{P(X)}{P(e)} = \sum_{i=1}^{|Y|} P(e|X, Y = y_i) \times P(Y = y_i|X) \times \frac{P(X)}{P(e)}.$$

Or, d'après la signification des arcs dans le graphe, le fait que l'arc (Y, Z) existe, mais que l'arc (X, Z) n'existe pas, signifie que Z est indépendant de X conditionnellement à Y. Autrement dit, $P(e|X, Y = y_i) = P(e|Y = y_i)$. Donc

$$P(X|e) = \sum_{i=1}^{|Y|} P(e|Y = y_i) \times P(Y = y_i|X) \times \frac{P(X)}{P(e)}.$$

En termes matriciels, on obtient :

$$[\![P(x_i|e)]\!]_{|X|} = \frac{1}{P(e)} \left([\![P(e|y_j)]\!]_{|Y|} \otimes [\![P(y_j|x_i)]\!]_{|Y||X|} \right) \odot [\![P(x_i)]\!]_{|X|}.$$

Par récurrence, on obtient la propriété suivante :

Supposons que le réseau bayésien soit une chaîne et que l'information e provienne de la feuille de cette chaîne. Soit un noeud B quelconque de la chaîne (excepté la feuille), et soit A son fils. Alors $[\![P(b_i|e)]\!]_{|B|} = \frac{1}{P(e)} \left([\![P(e|a_j)]\!]_{|A|} \otimes [\![P(a_j|b_i)]\!]_{|A||B|} \right) \odot [\![P(b_i)]\!]_{|B|}$. Là encore, le terme multiplicatif $\frac{1}{P(e)}$ n'a pas besoin d'être calculé explicitement puisqu'on sait que $[\![P(b_i|e)]\!]_{|B|} \odot \mathbb{1}_{|B|} = 1$.

D'une manière plus visuelle (cf. figure 7), voilà comment se passe la propagation de l'information $Z = z_1$ ou z_2 dans le réseau :

1. Z reçoit l'information sous forme d'un vecteur $E_{|Z|}$ qu'il ne peut plus prendre que certaines valeurs.
2. Z met alors à jour sa probabilité : $[\![P(z_i|e)]\!]_{|Z|} = \frac{1}{P(e)} \left([\![P(z_i)]\!]_{|Z|} \odot E_{|Z|} \right)$. Il envoie alors à son père un message $\lambda_{ZY} = [\![P(e|y_i)]\!]_{|Y|} = [\![P(z_j|y_i)]\!]_{|Z||Y|} \otimes E_{|Z|}$.
3. Lorsque Y reçoit le message envoyé par Z, il met à jour sa propre probabilité a posteriori : $[\![P(y_i|e)]\!]_{|Y|} = \frac{1}{P(e)} \left(\lambda_{ZY} \odot [\![P(y_i)]\!]_{|Y|} \right)$. Il envoie ensuite à son père un message $\lambda_{YX} = \lambda_{ZY} \otimes [\![P(y_j|x_i)]\!]_{|Y||X|}$.
4. lorsque X reçoit le message, il met à jour sa probabilité et emet un message λ_{XU}, et ainsi de suite...

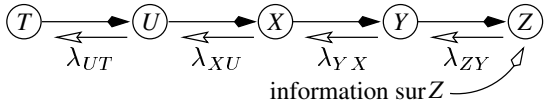


FIG. 7: Les messages à envoyer pour calculer les probabilités *a posteriori*

Lorsque les observations proviennent d'une racine et d'une feuille

Supposons maintenant qu'on ait eu une information e_T en T et une information e_Z en Z. Pour calculer les probabilités *a posteriori*, Dans ce cas, l'algorithme n'est pas compliqué, c'est en fait juste une petite généralisation de ce que nous avons fait précédemment :

1. On calcule les messages π et λ avec les formules données précédemment.
2. Pour les noeuds non observés, U, X, Y, si l'on note $\propto$ l'égalité à un coefficient multiplicatif près, les probabilités a posteriori sont égales à :

$$\begin{aligned}
[\![P(u_i|e_T, e_z)]\!]_{|U|} &\propto \left([\![P(u_i|t_j)]\!]_{|U||T|} \otimes \pi_{TU} \right) \odot \lambda_{XU}, \\
[\![P(x_i|e_T, e_z)]\!]_{|X|} &\propto \left([\![P(x_i|u_j)]\!]_{|X||U|} \otimes \pi_{UX} \right) \odot \lambda_{YX}, \\
[\![P(y_i|e_T, e_z)]\!]_{|Y|} &\propto \left([\![P(y_i|x_j)]\!]_{|Y||X|} \otimes \pi_{XY} \right) \odot \lambda_{ZY}.
\end{aligned}$$

3. Pour les noeuds observés, T et Z :

$$\begin{aligned}
[\![P(t_i|e_T, e_z)]\!]_{|T|} &\propto \left([\![P(t_j)]\!]_{|T|} \odot \lambda_{UT} \right) \odot E_T, \\
[\![P(z_i|e_T, e_z)]\!]_{|Z|} &\propto \left([\![P(z_i|y_j)]\!]_{|Z||Y|} \otimes \pi_{YZ} \right) \odot E_Z.
\end{aligned}$$

En fait, dans les calculs, on peut ne pas différencier les variables non observées de celles qui le sont si l'on rajoute à ces derniers un fils fictif qui enverrait un message λ égal au message E. Par exemple, pour T, on rajouterait un fils qui enverrait le message $E_{|T|}$.

Dans tout ce qui suit, on appellera les messages π lorsqu'ils sont envoyés des parents vers les enfants, et λ lorsqu'ils vont des enfants vers les parents.

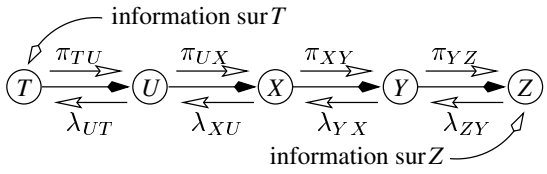


FIG. 8: Les messages à envoyer pour calculer les probabilités *a posteriori*

4.1.5 Redéfinition du calcul des probabilités *a priori* à l'aide des π et des λ

Si l'on examine attentivement l'algorithme donné page 13 pour calculer les probabilités *a priori* des variables du réseau, on se convaincra assez facilement que cet algorithme se transpose de la manière suivante :

Algorithme 2 (Calcul des probabilités *a priori*)

1. *À chaque noeud X du réseau, on associe un vecteur d'observation $E_{|X|} = \mathbb{1}_X$, et sur chaque arc (X, Y), on place un message $\lambda_{YX} = \mathbb{1}_X$.*
2. *Pour chaque noeud X dont les probabilités a priori de tous ses parents $Y_1, \ldots, Y_n$ ont été calculées, $[\![P(x_i)]\!]_{|X|}$ se calcule de la manière suivante :*

   $$[\![P(x_i)]\!]_{|X|} = \Big(\Big(\cdots\Big(\Big([\![P(x_i|y_1^{j_1},\ldots,y_n^{j_n})]\!]_{|X||Y_1|\ldots|Y_n|} \otimes \pi_{Y_1X}\Big) \otimes \pi_{Y_2X}\Big)\cdots\Big) \otimes \pi_{Y_nX}\Big) \odot \lambda_{Z_1X} \odot \cdots \odot \lambda_{Z_pX},$$

   où $Z_1, \ldots, Z_p$ sont les fils de X. Lorsque X a calculé sa probabilité a priori, il envoie un message $\pi_{XZ_j} = [\![P(x_i)]\!]_{|X|}$ à tous ses fils Z_j.
3. *En l'absence de cycle dans le graphe, soit i) il existe un noeud dont tous les parents ont déjà été calculés, auquel cas on peut appliquer le 2 ; soit ii) tous les noeuds ont été calculés.*

4.1.6 Calcul des probabilités *a posteriori* : le cas général

Comprendre l'algorithme ci-dessus se ramène globalement à comprendre le sens des arcs dans le réseau. L'existence d'un arc (X, Y) signifie que les variables X et Y ne sont pas indépendantes. Si l'on apprend quelque chose sur l'une de ces deux variables, il est donc normal de propager l'information à l'autre. Si, au contraire, il n'existe pas d'arc (X, Y), mais qu'il existe une suite d'arcs $(X, U_1), (U_1, U_2), \ldots, (U_{n-1}, U_n), (U_n, Y)$, alors une information en X doit avoir des répercussions sur U_1. Oui, mais maintenant, U_1 a reçu des informations, et donc il est normal qu'il en fasse profiter U_2, et ainsi de suite. Par récurrence, l'information va transiter jusqu'à Y. Maintenant, pourquoi les messages transmis à travers les arcs du réseau sont-ils suffisants pour calculer les probabilités *a posteriori* ? En fait, d'après la signification des arcs, si l'on n'a pas d'arc de X vers Y, cela signifie que conditionnellement aux variables $U_1, \ldots, U_n$, X et Y sont indépendants. Ainsi, à partir du moment où X a envoyé un message vers U_1 et que celui-ci a mis à jour sa probabilité, X devient indépendant de Y, et donc il n'a pas à envoyer d'autres messages vers Y pour que la probabilité de Y soit mise à jour proprement.

Conclusion : dans un réseau bayésien quelconque, on peut mettre à jour toutes les probabilités après l'arrivée d'observations uniquement en envoyant des messages à travers les arcs du réseau.

Encore faut-il savoir déterminer quels messages envoyer, et dans quel ordre. Pour répondre à la première question, il convient de comprendre quelle est la signification de ces messages. La figure 9 va nous y aider : l'arc (X, Y)

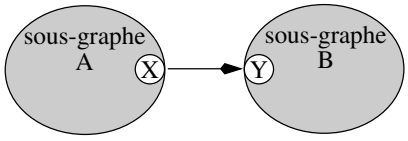


FIG. 9: De la signification du π et du λ.

sépare le réseau en deux sous-réseaux, désignés par les lettres A et B. L'ensemble des observations du sous-réseau A va avoir un certain impact sur X. Eh bien c'est cet impact que le vecteur π_{XY} propage vers Y. De même, λ_{YX} propage vers X l'impact des observations du sous-réseau B. Par conséquent, π_{XY} est totalement indépendant de λ_{YX}, et ces deux vecteurs peuvent même être calculés sur des machines fonctionnant en parallèle.

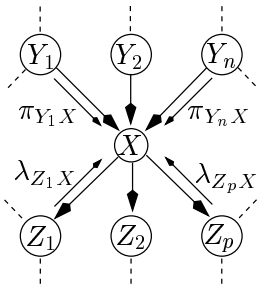


FIG. 10: Les messages à envoyer.

Considérons maintenant la figure 10 : un noeud X a pour parents $Y_1, \ldots, Y_n$ et pour enfants $Z_1, \ldots, Z_p$. Il désire envoyer un message λ_{XY_1} vers Y_1. Il doit donc considérer comme sous-réseau B la partie du réseau de la figure 10 en haut à gauche de Y_1, et comme sous-réseau A le reste du réseau de la figure 10. Donc, le message λ_{XY_1} doit impérativement renfermer les informations contenues dans tous les π_{Y_jX}, pour $j \geq 2$, car les Y_j ou leurs parents peuvent avoir été observées. Pour les mêmes raisons, λ_{XY_1} doit aussi renfermer les informations contenues dans tous les λ_{Z_iX}. Les équations aux dimensions vont nous montrer comment réaliser cela : les π_{Y_jX} ont pour taille $|Y_j|$. Or, X ne connaît a priori cette dimension que par l'intermédiaire de sa matrice de probabilité conditionnelle. Étant donné que λ_{XY_1} a pour dimension $|Y_1|$, il va falloir faire disparaître de la matrice de probabilité conditionnelle les dimensions en Y_j, $j \geq 2$. Le seul opérateur que l'on ait défini qui nous permette cela est le produit hypermatrice-vecteur. On devra donc calculer

$$\Big(\Big(\Big(\llbracket P(x|y_1, y_2, \ldots, y_n)\rrbracket_{|X||Y_1|\ldots|Y_n|} \otimes \pi_{Y_2X}\Big) \otimes \pi_{Y_3X}\Big) \otimes \cdots\Big) \otimes \pi_{Y_nX}.$$

Après calcul, on obtient une matrice de taille $|X||Y_1|$. Il faut donc encore éliminer la dimension en X (donc faire un produit matrice-vecteur) pour obtenir un message ayant la bonne taille. Tous les λ_{Z_iX} ainsi que E_X ont pour taille X. On va donc constituer un vecteur de taille X intégrant toutes leurs informations, et puis on va multiplier ce vecteur par la matrice de taille $|X||Y_1|$. On obtiendra alors un vecteur de taille Y_1, qui a en outre le bon goût de renfermer exactement les informations que l'on veut faire passer à Y_1. Donc :

$$\lambda_{XY_1} = \Big(\Big(\Big(\llbracket P(x|y_1, \ldots, y_n)\rrbracket_{|X||Y_1|\ldots|Y_n|} \otimes \pi_{Y_2X}\Big) \otimes \cdots\Big) \otimes \pi_{Y_nX}\Big) \otimes \left(\lambda_{Z_1X} \odot \lambda_{Z_2X} \odot \cdots \odot \lambda_{Z_pX} \odot E_X\right).$$

De la même manière, les équations aux dimensions nous montrent que

$$\pi_{XZ_1} = \Big(\Big(\Big(\llbracket P(x|y_1, \ldots, y_n)\rrbracket_{|X||Y_1|\ldots|Y_n|} \otimes \pi_{Y_1X}\Big) \otimes \cdots\Big) \otimes \pi_{Y_nX}\Big) \odot \lambda_{Z_2X} \odot \cdots \odot \lambda_{Z_pX} \odot E_X.$$

Maintenant, il reste à ordonner les envois de messages. D'après la sous-section précédente, sans perte de généralité, on peut supposer que la phase de calcul des probabilités *a priori* a établi des valeurs pour les λ et π de chaque arc du réseau. En partant de cette hypothèse, on peut montrer assez facilement par récurrence que l'algorithme suivant organise correctement les envois de messages :

Algorithme 3 (Calcul des probabilités *a posteriori*)

1. *On choisit un noeud X au hasard.*
2. *Ce noeud demande à ses voisins de lui envoyer un message. À leur tour, les voisins demandent à leurs voisins (sauf X) de leur envoyer des messages, et ainsi de suite : chaque noeud demande à tous ses voisins, excepté le noeud qui lui a demandé d'envoyer un message, de lui envoyer un message. Les noeuds aux extrémités du réseau ne peuvent plus demander de message à personne. Ils émettent alors leur message. Lorsqu'un noeud a reçu tous les messages qu'il attendait, il envoie alors le message qu'on lui avait demandé, et ainsi de suite jusqu'à X.*
3. *Lorsque X a reçu tous les messages qu'il avait demandé, il distribue des messages à tous ses voisins. À leur tour, ceux-ci distribuent des messages vers leurs voisins, excepté X, et ainsi de suite, jusqu'à ce qu'on ne puisse plus envoyer de message.*

Chaque message est calculé selon les formules données au bas de la page précédente. La phase 2 s'appelle une collecte d'informations et la phase 3 s'appelle une distribution d'informations.

Exemple : Considérons le graphe de la figure 11. Supposons que T, W, Y et Z aient été observés. Choisissons

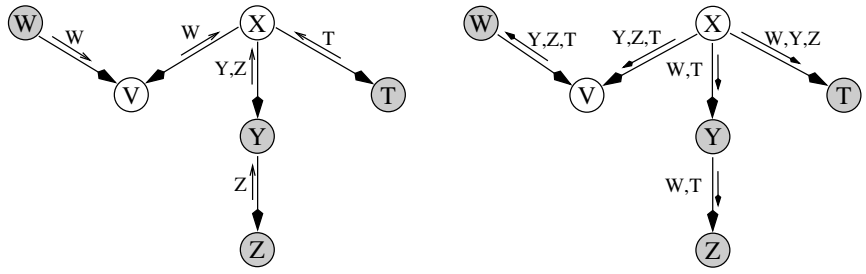


FIG. 11: Collecte d'informations.

un noeud au hasard, disons X.
collecte : X demande à V, Y et T de lui envoyer leurs messages. T envoie tout de suite son λ_{TX}, qui contient les informations sur l'observation de T. Y et V demandent quant à eux à Z et à W de leur envoyer des messages. W envoie π_{WV} à V. Ce message contient les informations sur l'observation de W. Z envoie λ_{ZY} pour que Y puisse connaître les informations sur l'observation de Z. V envoie à X un message λ_{VX} qui contient les informations sur l'observation de W. Enfin, Y envoie à X un message contenant à la fois les informations sur l'observation de Z et sur celle de Y. Ainsi, X connaît les impacts de toutes les informations.
diffusion : X envoie à V un message $\pi_{XV} = [\![P(x_i)]\!]_{|X|} \otimes (\lambda_{TX} \odot \lambda_{YX} \odot E_X)$. Autrement dit, le message π_{XV} renferme l'impact des observations sur T, Y et Z. Le noeud X envoie aussi les messages $\pi_{XY} = [\![P(x_i)]\!]_{|X|} \otimes (\lambda_{TX} \odot \lambda_{VX} \odot E_X)$ et $\pi_{XT} = [\![P(x_i)]\!]_{|X|} \otimes (\lambda_{YX} \odot \lambda_{VX} \odot E_X)$, qui contiennent respectivement les impacts des observations sur T, W, et sur Y, Z, W. Le noeud V envoie alors un message λ_{VW} qui contient l'impact des observations de Y, Z, T sur W. Enfin, Y envoie un message π_{YZ} qui contient l'impact de W, T sur Z.
Après la collecte et la diffusion, tous les noeuds ont reçu les impacts de chacune des observations. Une équation aux dimensions nous indique alors comment calculer les probabilités *a posteriori* des noeuds en fonction des messages qu'ils ont reçu. Un noeud X ayant pour parents $Y_1, \ldots, Y_n$ et pour enfants $Z_1, \ldots, Z_p$ connaît les dimensions de ses parents grâce à $[\![P(x_i|y_1, \ldots, y_n)]\!]_{|X||Y_1|\ldots|Y_n|}$. Pour faire disparaître les dimensions des parents, il va falloir faire des produits de la matrice de probabilité conditionnelle avec les messages π_{Y_iX}. On obtient alors un vecteur de taille $|X|$. Il suffit alors de faire des produits tensoriels avec E_X et avec les λ_{Z_jX} pour obtenir $[\![P(x_i|e)]\!]_{|X|}$:

Si e représente l'ensemble des observations que l'on veut propager dans le réseau, alors :

$$[\![P(x_i|e)]\!]_{|X|} = \Big(\Big(\Big([\![P(x|y_1, \ldots, y_n)]\!]_{|X||Y_1|\ldots|Y_n|} \otimes \pi_{Y_1X}\Big) \otimes \cdots\Big) \otimes \pi_{Y_nX}\Big) \odot \lambda_{Z_1X} \odot \cdots \odot \lambda_{Z_pX} \odot E_X.$$

4.2 Travail demandé dans la Partie III : les calculs dans le réseau

Question 15 Définissez la fonction `copy_hypermatrice : Hypermatrice -> Hypermatrice` qui prend en argument une hypermatrice et qui renvoie une copie de celle-ci. Les hypermatrices étant définies à partir du type `vect`, si l'on écrit `let hypermat1 = hypermat2 ; ;` les deux hypermatrices partageront leurs données, c'est-à-dire que si l'on modifie un élément dans `hypermat1`, il sera aussi modifié dans `hypermat2`, ce qui peut être gênant pour certains calculs. Le but de la présente fonction est donc d'empêcher ce partage des données. Vous utiliserez pour cela la fonction `copy_vect` du module `vect` (cf. le manuel de référence).

Question 16 Définissez la fonction `prod_tensoriel : Hypermatrice -> Hypermatrice -> unit` qui prend en argument deux hypermatrices et qui stocke dans la première le produit tensoriel des deux hypermatrices. Si ces dernières sont de dimensions différentes, elle lève une exception `Bad_Dimension "prod_tensoriel"`.

Question 17 Définissez la fonction `get_dim_hypermatrice : Hypermatrice -> (int * int) vect` qui prend en argument une hypermatrice et qui renvoie un vecteur dont chaque élément est un couple représentant des informations sur une dimension de l'hypermatrice. Le premier élément de ce couple correspond à l'indice du noeud correspondant à la dimension, dans le vecteur de noeuds représentant le réseau bayésien ; le deuxième est le nombre de modalités de ce noeud. Pour expliquer plus simplement, si, à une dimension de l'hypermatrice, on trouve `Vect (4, make_vect 5 1.)`, le couple renvoyé par la fonction est (4,5). Ainsi,

```
get_dim_hypermatrice (Mat(2, [|Vect (0, [|0.65; 1.3; 0.35|]);
                                Vect (0, [|0.2; 5.6; 0.8|])|]));;
```

renverra `[|(2,2); (0,3)|]`.

Question 18 Définissez la fonction `add_hypermatrice : Hypermatrice -> Hypermatrice -> unit` qui prend en argument deux hypermatrices et stocke dans la première la somme des deux. Si les matrices n'ont pas les mêmes dimensions, elle lève l'exception `Bad_Dimension "add_hypermatrice"`.

Question 19 Définissez la fonction `prod_hypermatrice_scalaire : Hypermatrice -> float -> unit` qui prend en arguments une hypermatrice et un scalaire, et qui stocke dans le premier le produit de l'hypermatrice par le scalaire.

Question 20 Définissez la fonction `rescale_proba : Hypermatrice -> unit` qui prend une hypermatrice représentant soit un message (π, λ), un vecteur de probabilité marginale, ou une hypermatrice de probabilité conditionnelle. La fonction modifie cette hypermatrice de manière à ce que la somme des éléments de chacun des vecteurs sur sa dernière dimension soit égale à 1. Par exemple, après la session

```
let M = Mat(2, [|Vect (0, [|0.65; 1.; 0.35|]);
                 Vect (0, [|0.2; 3.; 0.8|])|]);;
rescale_proba M;;
```

on obtient une hypermatrice `M` égale à :

```
M : Mat(2, [|Vect (0, [|0.325; .5; 0.175|]);
             Vect (0, [|0.05; .75; 0.2|])|])
```

Question 21 Les fonctions `collect_evidence` et `distribute_evidence` vous sont fournies dans le fichier `partie_3.ml`. Elles réalisent les collectes et distributions des messages décrits plus haut. Mais on ne voit pas s'afficher dans la fenêtre CAMLGRAPH les messages qu'elles font transiter dans le réseau.

Dans le fichier `partie_3.ml`, il existe justement deux fonctions, `draw_message : int * int -> int * int -> unit` et `draw_ask_message : int * int -> int * int -> unit`, qui dessinent précisément des flèches à côté des arcs sur lesquels transitent les messages. Les couples d'`int` passés en arguments de ces fonctions correspondent aux coodonnées x,y des noeuds aux extrémités des arcs de transition. Dans `draw_message`, les premières coordonnées sont celles du noeud qui envoie le message, et les dernières sont celles du noeud qui reçoit le message. Dans `draw_ask_message`, les premières coordonnées sont celles du noeud qui demande un message, et les dernières sont celles du noeud à qui il demande le message.

Afin de voir les messages s'afficher dans la fenêtre CAMLGRAPH, insérez dans les codes des fonctions `collect_evidence` et `distribute_evidence` des appels à ces fonctions.

Question 22 Les fonctions `collect_evidence` et `distribute_evidence` ne sont pas optimisées car elles effectuent plusieurs fois les mêmes calculs. En lisant attentivement ces fonctions, ou bien en les traçant (`trace "nom_de_fonction"`), essayer de trouver où sont réalisés ces calculs redondants, et trouvez une parade pour ne les faire qu'une seule fois.

Question 23 Est-il possible d'intervertir dans les fonctions `collect_evidence` et `distribute_evidence` les appels de `process_voisin` appliqué aux enfants du noeud courant et de `process_voisin` appliqué aux parents ? Expliquez pourquoi ?

Pour obtenir un exécutable de la partie III du TP, tapez : `camlc -custom -o fin_tp unix.zo graphics.zo partie_1.ml graph.zo interface.zo partie_2.ml partie_3.ml fin.zo -lgraph -lunix -lX11`