Département MIDO

Master 1 Informatique des Organisations

Parcours Miage

Projet de Représentation des Connaissances

Le problème de Fred

BOUHMADI Kacem

GOLETTO Michael

LIN Jiedi

PAGNIEZ Claire

Sommaire

Introduction

La représentation des connaissances est utilisée depuis la nuit des temps par les hommes pour faciliter la transmission du savoir, la compréhension des techniques et des sciences, et a permis à l'homme d'évoluer plus vite et plus loin sans perdre de vue ses acquis. Dans une époque plus récente, la représentation des connaissances trouve de nouvelles applications dans des domaines liés à l'intelligence artificielle et aux systèmes experts. Comment sont organisées les connaissances de l'homme, quels sont les liens qui les relient entre elles, et surtout comment les représenter de façon à ce qu'une machine puisse les comprendre de la même façon et être aussi efficace qu'un homme dans ses raisonnements ?

Les réseaux Bayesiens sont un des outils permettant de modéliser les connaissances de façon logique et précise. De tels graphes permettent de représenter différentes variables ainsi que les liens les reliant entre elles, mais également le degré d'incidence qu'elles ont les unes par rapport aux autres, puisqu'elles sont chacune liées à des probabilités. Ainsi, on peut calculer la probabilité que se produise un événement si l'on connait ses parents, mais on peut également déterminer le ou les faits ayant le plus probablement entrainé tel événement, en remontant dans les nœuds du réseau.

Le projet dont il est question dans ce dossier a pour base un problème lié à une situation concrète, que nous avons du modéliser sous la forme d'un réseau Bayesien. Pour cela, il a fallu déterminer les différentes variables et en quoi elles s'influençaient les unes par rapport aux autres, et également associer à chacune d'elle des probabilités, selon les données du problème et notre propre compréhension des informations. Ces différentes étapes sont exposées et développées dans la suite du rapport. Dans un deuxième temps, nous avons mis au point une application permettant à l'utilisateur de calculer pour chaque événement sa probabilité selon tel ou tel autre fait. Le développement de cette application sera également détaillé dans la suite du dossier.

L'application que nous avons réalisée est disponible à l'adresse suivante : kacem.bouhmadi.fr/master/rpc

I. Réseau Bayésien

1. Identification des variables

La première variable qui nous intéresse est la variable**interpreteur LISP**. En effet le problème de Fred consiste principalement à savoir si celui-ci fonctionne ou pas : *« Fred est en train de déboguer un programme LISP, […] et maintenant l'interpréteur LISP ne répond pas à tout autre frappe de clavier. »*

Il y a également la variable **invite de commande** qui est présentée dans le texte comme le signe que l'interpréteur fonctionne. Le fonctionnement de la variable **interpreteur LISP** influe directement sur la variable **invite de commande**. *« Fred ne peut pas voir l'invite de commande qui habituellement indique que l'interpréteur est en attente d'une entrée supplémentaire. »*

On peut ensuite d'après le texte mettre en avant les variables **hardware** et **programme**. En effet, d'après l'énoncé, ce sont les deux principaux faits qui peuvent avoir influé sur l'**interpreteur LISP** et causé le problème de Fred. *« Il y a seulement deux solutions qui pourraient causer l'arrêt de l'interpréteur : il y a des problèmes avec le hardware de l'ordinateur, il y a une erreur dans le code de Fred »*.

Enfin la dernière partie du texte nous permet de complexifier notre réseau en y introduisant deux autres variables : **editeur** et **curseur clignotant**. La variable **curseur** est une variable dont Fred peut immédiatement et simplement contrôler le fonctionnement. Sachant que la variable **editeur de texte** influe sur la variable **curseur clignotant** et que la variable **hardware** influe sur la variable**editeur de texte**, la variable **curseur clignotant** donne une très bonne indication à Fred de la cause du problème. *« Si le hardware fonctionne correctement alors l'éditeur de texte doit fonctionner. Si l'éditeur fonctionne, le curseur de l'éditeur doit clignoter. »*

En résumé nous pouvons mettre en avant les variables suivantes :

Variable Interpréteur LISP

Variable Invite de commande

Variable Programme

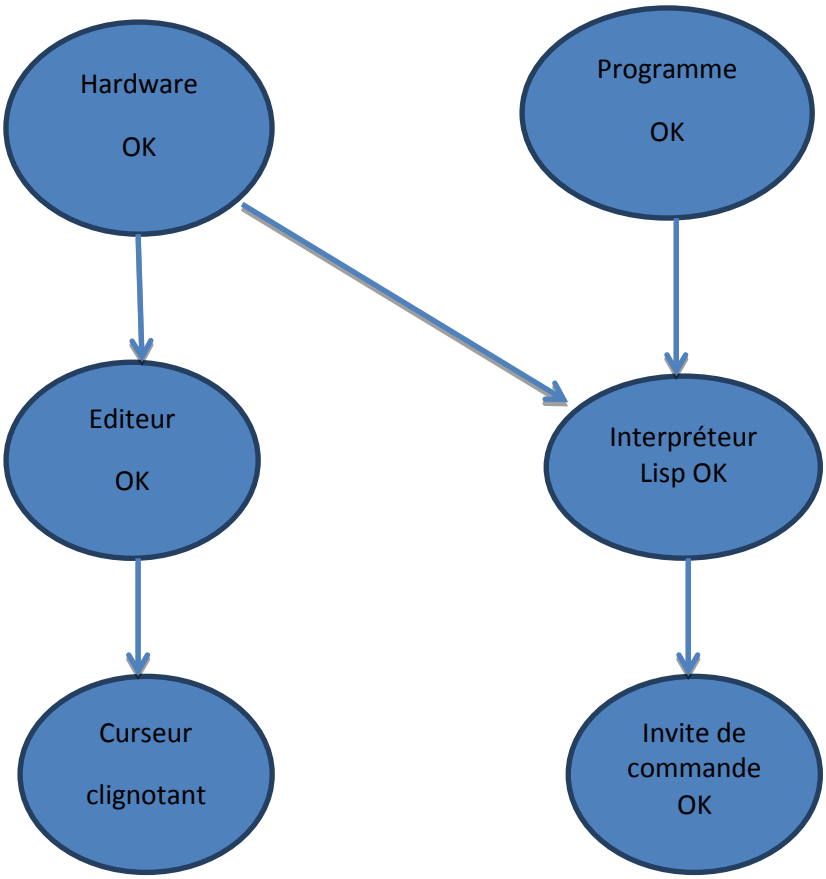
Variable Hardware

Variable Editeur de texte

Variable Curseur clignotant

2. Réseau Bayésien représentant le problème

Les dépendances directes entres les variables sont représentées dans le réseau Bayésien ci-dessous :



Nous avons supposé ici que Fred lance toujours l'éditeur de code à chaque fois qu'il veut programmer, c'est-à-dire que le dysfonctionnement de l'éditeur ne dépend que de l'état du hardware (et des éventuels bugs internes…).

3. Table de probabilité

La table des probabilités représente l'aspect quantitatif des connaissances liées au réseau Bayesien. Chaque variable se voit associée une probabilité, en fonction de ses parents (et uniquement ses parents). Dans le cadre de notre projet, certaines valeurs découlaient directement des données de l'énoncée. C'est le cas par exemple de la variable hardware (« *le hardware est assez fiable, et il fonctionne environ 99 fois sur 100* ») et de la variable Programme (« *le code LISP de Fred contient des erreurs 40 fois sur 100* »). Dans les autres cas, nous avons établis nous même les valeurs, selon ce qui nous semblait le plus logique et le plus cohérent.

Probabilité	Valeur en %
P (Programme OK)	60
P (Hardware OK)	99
P (Editeur OK / Hardware OK)	95
P (Editeur OK / ⌐Hardware OK)	10
P (Curseur clignotant/Editeur OK)	95
P (Curseur clignotant/ ⌐Editeur OK)	0
P (Interpréteur LispOK / Hardware OK ^ Programme OK)	97
P (Interpréteur LispOK / ⌐Hardware OK ^ Programme OK)	5
P (Interpréteur LispOK / Hardware OK ^⌐Programme OK)	80
P (Interpréteur LispOK / ⌐Hardware OK ^⌐Programme OK)	1
P (Invite de commande OK / Interpréteur Lisp OK)	80
P (Invite de commande OK / ⌐Interpréteur Lisp OK)	5

Explications :

Rappelons qu'on a supposé que Fred lance toujours l'éditeur de code avant de commencer à coder, c'est-à-dire que le fonctionnement de l'éditeur ne dépend pas de son lancement. On peut donc dire que l'état de l'éditeur dépend uniquement de l'état du hardware, qui est fiable à hauteur de 99%.

Sachant que le hardware fonctionne, seul un bug de l'éditeur lui-même pourrait le faire dysfonctionner. On peut donc supposer que l'éditeur fonctionnera 95 fois sur 100, laissant ainsi 5% de chance pour qu'un bug se produise (cela varie d'un éditeur à un autre) :

$$P(\text{Editeur OK} / \text{Hardware OK}) = 95\ \%$$

Si le hardware ne fonctionne pas alors l'éditeur ne devrait normalement pas fonctionner. Mais étant donné qu'un éditeur de code n'utilise pas forcément toutes les fonctionnalités du matériel, il y a une chance pour que celui-ci fonctionne quand même. Il reste bien entendu le cas où il y aurait des bugs dans l'éditeur lui-même, on a donc choisit une probabilité de 10% :

$$P(\text{Editeur OK} / \neg \text{Hardware OK}) = 10\ \%$$

Sachant que l'éditeur fonctionne, on a assigné une probabilité de 95% pour que le curseur clignote. On a supposé qu'il y a certaines opérations qui pourraient empêcher le clignotement du curseur comme la recherche ou la sauvegarde. A l'inverse, si l'éditeur ne fonctionne pas, alors d'après notre modèle le curseur n'a aucune chance de clignoter :

$$P(\text{Curseur clignotant/Editeur OK}) = 95\ \%$$
$$P(\text{Curseur clignotant/ } \neg \text{Editeur OK}) = 0\ \%$$

Si le hardware fonctionne et que le programme de Fred est sans bug, alors seul un bug de l'interpréteur pourrait l'arrêter. D'après le modèle qu'on a construit cela est tout à fait possible mais reste très peu probable vu qu'un interpréteur LISP est en général utilisé par beaucoup de programmeurs. On a donc opté pour une probabilité de 97 % pour que l'interpréteur fonctionne sachant que le hardware marche et que le programme est sans bugs :

$$P(\text{Interpréteur Lisp OK} / \text{Hardware OK} \wedge \text{Programme OK}) = 97\ \%$$

Si le programme est sans bug mais que le hardware ne fonctionne pas, alors il y a peu de chance que l'interpréteur fonctionne. Il reste cependant l'hypothèse que l'interpréteur n'utilise pas toutes les composantes du matériel, cependant il est certain qu'il en utilise plus qu'un éditeur de code et c'est pour cela que nous avons choisi une probabilité de 5% au lieu de 10% :

$$P(\text{Interpréteur Lisp OK} / \neg \text{Hardware OK} \wedge \text{Programme OK}) = 5\%$$

Si le hardware est fonctionnel mais que le programme est sans bugs, alors il est très probable que l'interpréteur fonctionne toujours s'il arrive à gérer les erreurs :

$$P(\text{Interpréteur Lisp OK} / \text{Hardware OK} \wedge \neg \text{Programme OK}) = 80\ \%$$

L'interpréteur Lisp a peu de chance de fonctionner correctement si le programme est bogué et que le hardware est dysfonctionnel :

$$P(\text{Interpréteur Lisp OK} / \neg \text{Hardware OK} \wedge \neg \text{Programme OK}) = 1\ \%$$

Si l'interpréteur Lisp fonctionne alors il est très probable que l'invite de commande soit affichée, à moins que l'interpréteur soit en train d'exécuter un code. L'exécution de code se produisant beaucoup moins fréquemment que l'édition, on a choisi d'assigner la valeur de 90% à cette probabilité conditionnelle :

$$P(\text{Invite de commande OK} / \text{Interpréteur Lisp OK}) = 90\%$$

Il est très peu probable que l'invite de commande soit fonctionnelle si l'interpréteur ne marche pas. Cependant on peut toujours considérer le cas où l'interpréteur puisse s'arrêter suite à un bug en laissant l'invite de commande ouverte :

$$P(\text{Invite de commande OK} / \neg\text{Interpréteur Lisp OK}) = 5\ \%$$

II. Implémentation

L'application que nous avons réalisée est disponible à l'adresse suivante : kacem.bouhmadi.fr/master/rpc

1. Théorie

Un réseau bayésien est vu comme un arbre dont les nœuds sont considérés comme des variables. Tous les ancêtres d'un nœud représentent les variables dont dépend la variable de ce nœud.

Une fois le réseau bayésien construit, on peut alors lui soumettre une requête. Cette requête consiste en la demande de la probabilité d'une variable, sachant une ou plusieurs autres variables.

Pour résoudre cette requête de façon optimale, nous avons décidé d'utiliser l'algorithme par élimination de variables, dont voici, en théorie, la procédure :

```
function ELIMINATION-ASK(X: the query variable,
        e:  evidence specified as an event
        bn:  Bayesian network specifying
             full joint distribution P(X1,...,Xn))
factors ← []
vars ← REVERSE(bn.vars)
for var in vars
   factors ← [MAKE-FACTOR(var,e)|factors]
   if HIDDEN(var),
      factors ← SUM-OUT(var,factors)
return NORMALISE(POINTWISE-PRODUCT(factors))
```

Nous avons suivis la même logique, mais nous avons modifié un peu l'ordre des opérations pour que ça correspondent bien à notre modélisation.

2. Implémentation

Nous avons implémenté ce programme en utilisant le langage PHP. Certes ce n'est pas le langage le plus adéquat pour résoudre ce genre de problématique, mais nous l'avons choisi pour des raisons pratiques (il est facile à appréhender par tout le monde, et permet d'utiliser certaines notions de la programmation objet).

1. Modélisation

Pour représenter un réseau bayésien, nous avons implémenté trois classes :

- **Variable** :
  La classe Variable permet de représenter un nœud du réseau bayésien. Sa structure est très simple, elle est composée d'un attribut nom et d'une liste d'objets de type Variable pour représenter les parents du nœud.

- **MatriceProbabilite** :
  Cette classe permet de modéliser un « tableau de probabilité » de ce type :

V_1	V_2	…	…	V_{n-1}	Valeurs
T/F	T/F	…	…	T/F	0.78
T/F	T/F	…	…	T/F	0.05
…	…	…	…	…	…
T/F	T/F	…	…	T/F	…
T/F	T/F	…	…	T/F	1

  Où V_1…$V_n$sont des variables qui prennent des valeurs booléennes. La dernière colonne représente la valeur de chaque ligne. Nous avons remarqué que ce tableau peut aussi servir à modéliser une requête en prenant comme convention le fait que la première variable soit la variable requête, et le reste des variables observables.
  Par exemple le tableau ci-dessous est la modélisation de la requête P (V_1 / V_2^¬V_3^ V_4) :

V1	V2	V3	V4	Valeur
True	True	False	True	--

  Ce tableau est modélisé à l'aide de trois variables. Une première variable pour représenter les variables, une deuxième pour représenter la matrice des valeurs booléennes et enfin la dernière un vecteur pour les valeurs correspondantes.

- **ReseauBayesien** :
  Cette classe permet de stocker les probabilités correspondantes au réseau bayésien dans une liste de type MatriceProbabilite, nommée data.
  L'utilité de cette classe réside dans ses différentes méthodes, nous allons ici décrire les plus importantes d'entre elles :

  - **variableCachees** : Permet de trouver les variables cachées par rapport à une requête donnée.
  - **Ancetres** : retourne tous les ancêtres d'une variable dans un réseau bayésien. C'est une méthode static et récursive qui remonte l'arbre jusqu'au nœud parent ne possédant aucun parent.
  - **eliminationAsk** : C'est la méthode la plus importante de la classe. Elle prend en paramètre une requête et retourne son résultat dans une variable de type MatriceProbabilite.
  - **Normalise** : Méthode static qui permet de normaliser le vecteur de valeurs contenu dans une variable de type MatriceProbabilite, passée en paramètre.

- **sumOut** : Permet de calculer la somme partielle d'un tableau de données « factor », par rapport à une variable « item ».
- **makeFactor** : Cette méthode permet de trouver la probabilité d'un nœud sachant ses parents. Elle retourne donc une variable de type MatriceProbabilite.
- **eliminerVarObs** : Permet de supprimer les lignes qui ne servent à rien dans un tableau où il y a des variables observables (dont on connait la valeur). Par exemple si on sait qu'une variable **V_2 vaut false**, le tableau suivant :

V1	V2	Valeurs
True	True	a
True	False	b
False	True	c
False	False	d

Deviendra :

V1	V2	Valeurs
True	False	b
False	False	d

- **product** : Cette méthode permet de multiplier deux tableaux de type MatriceProbabilite, tout en prenant en compte les variables communes aux deux.

2. Listing du code source

Pou rappel, l'application que nous avons réalisée est disponible à cette adresse : kacem.boumahdi.fr/master/rpc

Page index.php

```php
<?php
    // Initialisation du réseau bayésien :
    include('BayesianNetwork.v4.class.php');

    $rb = new ReseauBayesien();

    $variables = array();
    $variables['h'] = new Variable('h'); // hardware
    $variables['p'] = new Variable('p'); // Code
    $variables['e'] = new Variable('e'); // Editeur
    $variables['i'] = new Variable('i'); // Interpréteur
    $variables['c'] = new Variable('c'); // Curseur
    $variables['ic'] = new Variable('ic'); // Invite de commande

    $rb->liste_variables = array_values($variables);
```

```
    $variables['h']->liste_fils = array($variables['e'], $variables['i']);
    $variables['p']->liste_fils = array($variables['i']);
    $variables['e']->liste_fils = array($variables['c']);
    $variables['i']->liste_fils = array($variables['ic']);

    $variables['e']->liste_parents = array($variables['h']);
    $variables['i']->liste_parents = array($variables['h'],
$variables['p']);
    $variables['c']->liste_parents = array($variables['e']);
    $variables['ic']->liste_parents = array($variables['i']);

    $data = array();
    $data[] = new MatriceProbabilite(1, array($variables['p']),
array(array(true)), array(0.6)); // P (Programme OK)
    $data[] = new MatriceProbabilite(1, array($variables['h']),
array(array(true)), array(0.99)); // P (Hardware OK)
    $data[] = new MatriceProbabilite(2, array($variables['e'],
$variables['h']), array(array(true, true)), array(0.95)); // P (Editeur OK
/ Hardware OK)
    $data[] = new MatriceProbabilite(2, array($variables['e'],
$variables['h']), array(array(true, false)), array(0.1)); // P (Editeur OK
/ ¬Hardware OK)
    $data[] = new MatriceProbabilite(2, array($variables['c'],
$variables['e']), array(array(true, true)), array(0.95)); // P (Curseur
clignotant/Editeur OK)
    $data[] = new MatriceProbabilite(2, array($variables['c'],
$variables['e']), array(array(true, false)), array(0)); // P (Curseur
clignotant/ ¬Editeur OK)
    //P (Interpréteur LispOK / Hardware OK ∧ Programme OK)
    $data[] = new MatriceProbabilite(3, array($variables['i'],
$variables['h'], $variables['p']), array(array(true, true, true)),
array(0.97));
    //P (Interpréteur LispOK / ¬Hardware OK ∧ Programme OK)
    $data[] = new MatriceProbabilite(3, array($variables['i'],
$variables['h'], $variables['p']), array(array(true, false, true)),
array(0.05));
    //P (Interpréteur LispOK / Hardware OK ∧ ¬Programme OK)
    $data[] = new MatriceProbabilite(3, array($variables['i'],
$variables['h'], $variables['p']), array(array(true, true, false)),
array(0.8));
    //P (Interpréteur LispOK / ¬Hardware OK ∧ ¬Programme OK)
    $data[] = new MatriceProbabilite(3, array($variables['i'],
$variables['h'], $variables['p']), array(array(true, false, false)),
array(0.01));
    // P (Invite de commande OK / Interpréteur Lisp OK)
    $data[] = new MatriceProbabilite(2, array($variables['ic'],
$variables['i']), array(array(true, true)), array(0.8));
    // P (Invite de commande OK / ¬Interpréteur Lisp OK)
    $data[] = new MatriceProbabilite(2, array($variables['ic'],
$variables['i']), array(array(true, false)), array(0.05));

    $rb->data = $data;
    $rb->initData();
?>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Projet RPC</title>
```

```
        <link rel="stylesheet" href="rpc.css" />
    </head>

    <body>
    <table id = "entete">
    <tr><td></td><td style="text-align:right;"><img src="images/logo-
dauphine.jpg" alt="universitee Paris Dauphine"/> </td></tr>
    </table>
    <h1>Projet de représentation des connaissances : le problème de
Fred</h1>
    <div id="resume">
    <h2>Résumé </h2>
    <div id = "enonce">
        <p>Fred est en train de débugger un programme LISP. Il vient de
taper une expression pour l'interpréteur LISP, et mainteannt l'intérpréteur
LISP ne répond plus à aucune frappe du clavier. Fred ne peut pas voir
l'invite de commande qui indique habituellement que l'interpréteur est en
attende d'une entrée supplémentaire. D'après ce que sait Fred, il y a
seulement deux situations qui pourraient causer l'arret de l'interpréteur :
</p>
        <p>(1) Il y a des problèmes avec le harware de l'ordinateur</p>
        <p>(2) Il y a une erreur dans le code de Fred</p>
        <p> Fred est également en train d'utiliser un éditeur pour écrire
et modifier son code. Si le harware fonctionne correctement, alors
l'éditeur doit fonctionner. Si l'éditeur fonctionne, son curseur doit
clignoter. on sait de plus que le hardware est assez fiable, et qu'il
fonctionne 99 fois sur 100, alors que le code Lisp de Fred contient des
erreurs 40 fois sur 100.</p>
        </div>
        <div id = "arbre">
        <p style = "text-align:center;"><img src="images/arbre.jpg"
alt="résau bayesien"  /></p>
        </div>
        <div id = "probas">
            <table  id="tabprobas">
                <tr>
                    <td>P(Code)</td>
                    <td>60%</td>
                </tr>

                <tr>
                    <td>P(H)</td>
                    <td>99%</td>
                </tr>


                <tr>
                <td>P(E|H)</td>
                <td>95%</td>
                </tr>
                <tr>
                <td>P(E|¬H)</td>
                <td>10%</td>
                </tr>

                <tr>
                <td>P(C|E)</td>
                <td>95%</td>
                </tr>

                <tr>
```

```
                <td>P(C|⌐E)</td>
                <td>0%</td>
                </tr>

                <tr>
                <td>P(I|H^C)</td>
                <td>97%</td>
                </tr>
                <tr>
                <td>P(I|H^⌐C)</td>
                <td>5%</td>
                </tr>
                <tr>
                <td>P(I|⌐H^C)</td>
                <td>80%</td>
                </tr>
                <tr>
                <td>P(I|⌐H^⌐C)</td>
                <td>1%</td>
                </tr>

                <tr>
                <td>P(IC|I)</td>
                <td>80%</td>
                </tr>
                <tr>
                <td>P(IC|I)</td>
                <td>5%</td>
                </tr>
            </table>
        </div>
        <div id="legende">
            <p>Variable H : représente la probabilité que le hardware
fonctionne.</p>
            <p>Variable Code : représente la probabilité que le code de
Fred ne contienne pas d'erreur.</p>
            <p>Variable E : représente la probabilité que l'éditeur
fonctionne.</p>
            <p>Variable I : représente la probabilité que l'interpréteur
fonctionne.</p>
            <p>Variable C : représente la probabilité que le curseur
clignote.</p>
            <p>Variable IC : représente la probabilité que l'invite de
commande fonctionne</p>
            </div>
        </div>
        <br/>
        <br/>
        <div id = "listevariables">
        <h2>Application</h2>
            <form method="post" action="#">
                <table id="tabihm">
                    <tr><td >Quelle probabilité voulez vous
évaluer?</td></tr>

                    <tr><td>
                        <select name="requete" id="requete">
                        <option value="h">Hardware</option>
                        <option value="e">Editeur</option>
                        <option value="c">Curseur</option>
                        <option value="p">Code</option>
```

```
                        <option value="i">Interpreteur</option>
                        <option value="ic">Invite de commande</option>
                    </td><td>

                            est :
                            <input type="radio" name="reqval" value="vraie"
id="reqvalvraie" checked="checked" /> <label for="vraie">Vraie</label>
                            <input type="radio" name="reqval" value="faux"
id="reqvalfaux" /> <label for="faux">Fausse</label>

                    </td></tr>

                </table>
                <br/>
                <table>

                    <tr><td>Variable hardware</td>
                    <td>
                        <select name="h" id="hardware">
                        <option value="inconnue">Inconnue</option>
                        <option value="vraie">Vrai</option>
                        <option value="faux">Faux</option>
                    </td>

                    <tr><td>Variable editeur</td>
                    <td>
                        <select name="e" id="editeur">
                        <option value="inconnue">Inconnue</option>
                        <option value="vraie">Vrai</option>
                        <option value="faux">Faux</option>
                    </td></tr>

                    <tr><td>Variable curseur</td>
                    <td>
                        <select name="c" id="curseur">
                        <option value="inconnue">Inconnue</option>
                        <option value="vraie">Vrai</option>
                        <option value="faux">Faux</option>
                    </td></tr>

                    <tr><td>Variable code</td>
                    <td>
                        <select name="p" id="code">
                        <option value="inconnue">Inconnue</option>
                        <option value="vraie">Vrai</option>
                        <option value="faux">Faux</option>
                    </td></tr>

                    <tr><td>Variable interpreteur</td>
                    <td>
                        <select name="i" id="interpreteur">
                        <option value="inconnue">Inconnue</option>
                        <option value="vraie">Vrai</option>
                        <option value="faux">Faux</option>
                    </td></tr>

                    <tr><td>Variable invite de commande</td>
                    <td>
                        <select name="ic" id="invite">
                        <option value="inconnue">Inconnue</option>
                        <option value="vraie">Vrai</option>
```

```
                        <option value="faux">Faux</option>
                    </td></tr>
                    <tr><td>      </td></tr>
                    <tr>
                        <td><input type="submit" name="envoyer"
value="Faire la requete !" /></td>
                    </tr>
                </table>

            </form>
            <br/>
            <div id="resultats">

            <?php
                function varEmpty($var)
                {
                    return empty($_POST[$var]);
                }
                function varUnknown($var)
                {
                    return $_POST[$var]=='inconnue';
                }

                if(isset($_POST['envoyer']))
                {
                    $err = array();

                    $vars = array('h', 'e', 'c', 'p', 'i', 'ic');

                    // On teste les valeurs récupérées ...
                    if(empty($_POST['requete']) || empty($_POST['reqval']))
                        $err[] = 'Vous devez spécifier une variable de
requête et sa valeur';
                    if(in_array(true, array_map('varEmpty', $vars)))
                        $err[] = 'Vous devez renseigner au moins une
variable observée';

                    $isVarUnknown = array_map('varUnknown', $vars);
                    if(!in_array(false, $isVarUnknown))
                        $err[] = 'Vous devez renseigner au moins une
variable observée';

                    // On commence le traitement
                    if(count($err) != 0)
                        var_dump($err);
                    else
                    {
                        $var_req = $variables[$_POST['requete']];
                        $val_var_req = $_POST['reqval'] == 'vraie' ? true :
false;;

                        $var_obs = array();
                        $val_var_obs = array();
                        for($i=0; $i < count($isVarUnknown); $i++)
                        {
                            if(!$isVarUnknown[$i])
                            {
                                $var_obs[] = $variables[$vars[$i]];
                                $val_var_obs[] =
$_POST[(string)$variables[$vars[$i]]] == 'vraie' ? true : false;
                            }
```

```
                        }
                        $req = new MatriceProbabilite(count($var_obs)+1,
array_merge(array($var_req), $var_obs), array_merge(array($val_var_req),
$val_var_obs), array());

                        echo '<h2>Résultats</h2>';
                        $result = $rb->eliminationAsk($req);
                        $j = 0;

                        foreach($result->matrice as $ligne)
                        {
                            $keys = array_keys($ligne);
                            echo 'Proba que '.$keys[0].' soit
'.($ligne[$keys[0]] == true ? ' vraie ' : ' faux ').', sachant ';
                            for($i=1; $i < count($ligne); $i++)
                                echo $keys[$i].' soit '.
($ligne[$keys[$i]]== true ? ' vraie ' : ' faux ') .' et ';

                            echo ' = '. $result->valeurs[$j++].' <br />';
                        }
                    }
                }

            ?>
        </div>
        <br/>
            </div>
    </body>
</html>
```

Classe php

```
<?php

    class ReseauBayesien
    {

        public $data;
        public function __construct()
        {
            $this->data = array();
        }

        // A partir du réseau, il suffit de lui donner la var requete, et
les variables observées pour sortir les variables cachées
        public function variableCachees($var_req, $var_observees)
        {
            if(!is_array($var_observees) || is_array($var_req)) return
array();

            $var_cachees = array();
            $var_cachees = $this->ancetres($var_req);
            foreach($var_observees as $vo)
                $var_cachees = array_merge($var_cachees, $this-
>ancetres($vo));
                        $var_cachees = array_diff($var_cachees,
$var_observees);
            $var_cachees = array_diff($var_cachees, array($var_req));
            return array_unique($var_cachees);
        }
```

```
        public static function ancetres($variable)
        {
            $ret = array();
            if($variable->liste_parents != null || count($variable-
>liste_parents) > 0)
            {
                foreach($variable->liste_parents as $parent)
                {
                    $ret[] = $parent;
                    $ret = array_merge($ret,
ReseauBayesien::ancetres($parent));
                }
            }
            return $ret;
        }

        public function initData()
        {
            $data1 = array();
            foreach($this->data as $item)
            {
                $temp = new MatriceProbabilite($item->nb_variables, $item-
>variables, $item->matrice, $item->valeurs);
                $temp->matrice[0][0] = !$temp->matrice[0][0];
                $temp->valeurs[0] = 1 - $temp->valeurs[0];
                $data1[] = $temp;
            }
            $this->data = array_merge($this->data, $data1);


            foreach($this->data as $item)
            {
                for($j=0; $j < count($item->matrice); $j++)
                {
                    $i=0;
                    foreach ($item->matrice[$j] as $k => $v) {
                        if($i == count($item->matrice[$j])) break;
                        unset ($item->matrice[$j][$k]);
                        $new_key =  $item->variables[$i]->nom;
                        $item->matrice[$j][$new_key] = $v;
                        $i++;
                    }
                }
            }
        }

        public function eliminationAsk($req) // param : MatriceProbabilite
        {
            $var_req = $req->variables[0];
            $var_observees = array_diff_key($req->variables, array(0));

            $factors = array();
            $var_cachees = $this->variableCachees($var_req,
$var_observees);
            $i = 0;

            $union_vars = array_merge(array_merge(array($var_req ),
$var_observees), $var_cachees);

            if(in_array($var_req, $var_observees))
            {
```

```
                $temp = array_search($var_req, $var_observees);
                return new MatriceProbabilite(count($req->variables), $req-
>variables, $req->matrice, array((int) ($req->matrice[0] == $req-
>matrice[$temp])));
            }

            foreach($union_vars as $item)
            {
                $factors[$i] = $this->makeFactor($item, $req-
>matrice[array_search($item, $req->variables)] , $var_observees, $req);

                $i++;
            }

            $product = $factors[0];
            for($j=1; $j < count($factors); $j++)
                $product = $this->product($product, $factors[$j]);

            foreach($union_vars as $item)
            {
                if(in_array($item, $var_cachees))
                {
                    $product = $this->sumOut($item, $product);
                }
            }

            return $this->normalise($product);
        }

        public static function normalise($product)
        {
            $somme = 0;
            $product->valeurs = array_values($product->valeurs );
            foreach($product->valeurs as $val)
                $somme += $val;

            if($somme == 0) return 0;

            for($i=0; $i < count($product->valeurs); $i++)
                $product->valeurs[$i] /= $somme;
            return $product;
        }


        public function sumOut($item, $factor)
        {
            if(!in_array($item, $factor->variables))
            {

                return 0;
            }

            if($factor->nb_variables == 1) return 1;

            $matrix = new MatriceProbabilite($factor->nb_variables-1, null,
null, null);
            $matrix->variables = array_diff($factor->variables,
array($item));
            $matrix->matrice = array();

            $indice_var = array_search($item, $factor->variables);
```

```
            $i = 0;

            foreach($factor->matrice as $ligne)
            {
                $ligne_temp = array_diff_assoc($ligne,
array((string)$factor->variables[$indice_var] => $ligne[(string)$factor-
>variables[$indice_var]]));


                if(in_array($ligne_temp, $matrix->matrice))
                {
                    $indice_ligne = array_search($ligne_temp, $matrix-
>matrice);
                                        $matrix->valeurs[$indice_ligne] +=
$factor->valeurs[$i];
                }
                else
                {

                    $matrix->matrice[] = $ligne_temp;

                    $matrix->valeurs[] = $factor->valeurs[$i];


                $i++;
            }

            return $matrix;
        }

        public function makeFactor($noeud, $val_noeud, $var_observees,
$req)
        {
            $matrix = new MatriceProbabilite(0, null, null, null);

            $matrix->variables[] = $noeud;
            $matrix->variables = array_unique(array_merge($matrix-
>variables, $noeud->liste_parents));


            $matrix->nb_variables = count($matrix->variables);
            $i=0;
            foreach($this->data as $item)
            {
                $i++;
                if(count($item->variables) != count($matrix->variables))
continue;
                if($item->variables[0] != $matrix->variables[0]) continue;

                $pass = false;
                for($i = 1; $i < count($matrix->variables); $i++)
                {
                    if(!in_array($matrix->variables[$i], $item->variables))
                    {
                        $pass = true;
                        break;
                    }
                }
                if(!$pass)
                {
```

```
                    if((in_array($noeud, $var_observees) &&
array_key_exists((string)$noeud, $item->matrice[0]) &&
                        $val_noeud == $item->matrice[0][(string)$noeud])
                        || !in array($noeud, $var observees))
                    {
                        if(!is_array($matrix->matrice)) $matrix->matrice =
array();
                        $matrix->matrice = array_merge($matrix->matrice,
$item->matrice);
                        foreach($item->valeurs as $value)
                            $matrix->valeurs[] = $value;
                    }
                }
            }

                $this->eliminerVarObs($matrix, $var_observees, $req);

            return $matrix;
        }

        public function eliminerVarObs($matrix, $var_observees, $req)
        {
            $var_observees = array_values($var_observees);

            $val_var_observees = $to_del = array();

            for($i=0; $i < count($var_observees); $i++)
            {
                if(in_array($var_observees[$i], $matrix->variables))
                    $val_var_observees[] = $req-
>matrice[array_search($var_observees[$i], $req->variables)];
                else
                    $to_del[] = $i;
            }
            foreach($to_del as $indice)
                unset($var_observees[$indice]);

            $var_observees = array_values($var_observees);

            ///*
            $to_del = array();
            $j = 0;
            for($i=0; $i < count($var_observees); $i++)
            {
                foreach($matrix->matrice as $ligne)
                {
                    if($ligne[(string)$var_observees[$i]] !=
$val_var_observees[$i])
                        $to_del[] = $j;
                    $j++;
                }
                foreach($to_del as $indice)
                {
                    unset($matrix->matrice[$indice]);
                    unset($matrix->valeurs[$indice]);
                }
            }

        }

        public function eliminerVarObs1($matrix, $var_observees, $req)
```

```
        {
            // On élimine les lignes qui ne servent à rien par rapport aux
variables observées :

            $temp = array_diff($matrix->variables, $var_observees);
            if(count($temp) == 0)
            {
                $i=0;
                $val_var_observees = array();
                foreach($var_observees as $var)
                    $val_var_observees[] = $req->matrice[array_search($var,
$req->variables)];

                foreach($this->data as $item)
                {
                    if($matrix->variables == $item->variables)
                    {
                        $break = false;
                        foreach($matrix->matrice as $ligne)
                        {
                            for($j=0; $j < count($ligne); $j++)
                            {
                                if($ligne[$j] !=
$val_var_observees[array_search($matrix->variables[$j], $var_observees)])
                                    $break = true;
                            }
                            if(!$break)
                            {
                                $matrix->matrice = array($ligne);
                                $matrix->valeurs = array($matrix-
>valeurs[$i]);
                                return $matrix;
                            }
                            $i++;
                        }
                    }
                }
            }

            $matrix->variables = array_values($matrix->variables);

            $indices_var_obs = array();
            $num_var_obs = array();
            $valeus_var_obs = array();
            for($i=0; $i < count($matrix->variables); $i++)
            {
                if(in_array($matrix->variables[$i], $var_observees))
                {
                    $indices_var_obs[] = (string)$matrix->variables[$i];
                    $num_var_obs[] = $i;
                    $valeus_var_obs[] = $req->matrice[array_search($matrix-
>variables[$i], $req->variables)];
                }
            }
            $to_del = array();
            for($k=0; $k < count($indices_var_obs); $k++)
            {
                for($i=0; $i < count($matrix->matrice); $i++)
                {
                    $ligne = $matrix->matrice[$i];
```

```
                    if($ligne[$indices_var_obs[$k]] != $valeus_var_obs[$k])
                        $to_del[] = $i;
                    unset($matrix->matrice[$i][$indices_var_obs[$k]]);
                }
            }
            for($k=0; $k < count($indices_var_obs); $k++)
            {
                foreach($to_del as $indice)
                {
                    unset($matrix->matrice[$indice]);
                    unset($matrix->valeurs[$indice]);
                }
                unset($matrix->variables[$num_var_obs[$k]]);
            }
            if($matrix->matrice != null)
            {
            $matrix->variables = array_values($matrix->variables);
            $matrix->nb_variables = count($matrix->variables);
            $matrix->matrice = array_values($matrix->matrice);
            $matrix->valeurs = array_values($matrix->valeurs);
            }
            return $matrix;
        }


        public function product($m1, $m2)
        {
            // on récupère les colonnes en commun
            $var_commun = array_intersect($m1->variables, $m2->variables);

            // On trie les deux matrices par rapport aux colonnes communes
            $i = 0;
            $c1 = array();
            foreach ($m1->matrice as $key => $row)
            {
                foreach($var_commun as $item)
                    $c1[$i++][$key]  = $row["$item"];
                $i = 0;
            }
            foreach($c1 as $item)
                array_multisort($item, SORT_DESC, $m1->matrice, $m1-
>valeurs);


            $i = 0;
            $c2 = array();
            foreach ($m2->matrice as $key => $row)
            {
                foreach($var_commun as $item)
                    $c2[$i++][$key]  = $row["$item"];
                $i = 0;
            }
            foreach($c2 as $item)
                array_multisort($item, SORT_DESC, $m2->matrice, $m2-
>valeurs);


            $product = new MatriceProbabilite($m1->nb_variables + $m2-
>nb_variables - count($var_commun));
            $product->variables = array_unique(array_merge($m1->variables,
$m2->variables));
```

```
            $m1->valeurs = array_values($m1->valeurs);
            $m2->valeurs = array_values($m2->valeurs);
            $i = $j = 0;
            foreach($m1->matrice as $item1)
            {
                $j = 0;
                foreach($m2->matrice as $item2)
                {
                    $pass = false;
                    foreach($var_commun as $comm)
                    {
                        if($item1["$comm"] != $item2["$comm"]) $pass =
true;
                    }
                    if(!$pass)
                    {
                        $product->matrice[] = array_merge($item1, $item2);
                        $product->valeurs[] = $m1->valeurs[$i] * $m2-
>valeurs[$j];

                    }
                    $j++;
                }
                $i++;
            }

            return $product;
        }
    }

    class MatriceProbabilite
    {
        public $nb_variables; //int
        public $variables; // array
        public $matrice; // array
        public $valeurs; // array

        public function __construct($nb_variables, $variables=null,
$matrice=null, $valeurs=null)
        {
            $this->nb_variables = $nb_variables;

            $this->valeurs = $valeurs; // de taille 2^nb_var
            $this->matrice = $matrice; // de taille 3 x (2^nb_var)
            $this->variables = $variables;
        }

        // retourne un jeu de test...
        public static function fillMatrix()
        {
            $m1 = new MatriceProbabilite(2);
            $m1->variables = array('v1', 'v2');

            $m1->matrice[] = array($m1->variables[0] => true, $m1-
>variables[1] => true);
            $m1->matrice[] = array($m1->variables[0] => true, $m1-
>variables[1] => false);
            $m1->matrice[] = array($m1->variables[0] => false, $m1-
>variables[1] => true);
```

```
            $m1->matrice[] = array($m1->variables[0] => false, $m1-
>variables[1] => false);

            $m1->valeurs = array(0.1, 0.2, 0.3, 0.4);

            $m2 = new MatriceProbabilite(3);
            $m2->variables = array('u3', 'u1', 'u2');

            $m2->matrice[] = array($m2->variables[0] => false, $m2-
>variables[1] => false, $m2->variables[2] => false);
            $m2->matrice[] = array($m2->variables[0] => false, $m2-
>variables[1] => false, $m2->variables[2] => true);
            $m2->matrice[] = array($m2->variables[0] => false, $m2-
>variables[1] => true, $m2->variables[2] => false);
            $m2->matrice[] = array($m2->variables[0] => false, $m2-
>variables[1] => true, $m2->variables[2] => true);
            $m2->matrice[] = array($m2->variables[0] => true, $m2-
>variables[1] => false, $m2->variables[2] => false);
            $m2->matrice[] = array($m2->variables[0] => true, $m2-
>variables[1] => false, $m2->variables[2] => true);
            $m2->matrice[] = array($m2->variables[0] => true, $m2-
>variables[1] => true, $m2->variables[2] => false);
            $m2->matrice[] = array($m2->variables[0] => true, $m2-
>variables[1] => true, $m2->variables[2] => true);

            $m2->valeurs = array(0.5, 0.6, 0.7, 0.8, 0.9, 0.11, 0.12,
0.13);

            return array($m1, $m2);
        }

    }
    class Variable
    {
        public $nom; //string
        public $liste_parents; //List<Variable>

        public function __construct($nom)
        {
            $this->nom = $nom;
            $this->liste_parents = array();
        }

        public function __toString()
        {
            return $this->nom;
        }
    }

?>
```