République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Informatique
Département d'Intelligence Artificielle et Sciences des Données (DIASD)

Master 2 Systèmes Informatiques Intelligents

Rapport TP RCR

Representation des Connaissances et Raisonnement

Théorie des Fonction de Croyances, Logique Floue, Reseau Bayesian et Logique Possibiliste

---

Présenté par :
Anane Dalia Khadidja
Bokhtache Khawla
Groupe 1

Responsable du TP :
H.F.Khellaf

Année universitaire 2024–2025

CONTENTS

CHAPTER 1

INTRODUCTION

Introduction générale

Dans de nombreux domaines de l'informatique et de l'intelligence artificielle, l'information disponible est souvent incomplète, imprécise, incertaine, voire conflictuelle. Les approches classiques basées sur la logique booléenne et les probabilités exactes montrent rapidement leurs limites lorsqu'il s'agit de modéliser le raisonnement humain, de fusionner des sources d'information hétérogènes ou de prendre des décisions dans des environnements complexes. Afin de répondre à ces problématiques, plusieurs cadres théoriques dédiés au raisonnement sous incertitude ont été développés.

Ce travail s'inscrit dans ce contexte et présente quatre approches majeures et complémentaires du raisonnement incertain : la théorie des fonctions de croyance, la logique floue, les réseaux bayésiens et la théorie des possibilités. Bien que ces approches poursuivent un objectif commun — la gestion de l'incertitude — elles se distinguent par la nature de l'information qu'elles traitent, leurs hypothèses fondamentales et leurs mécanismes d'inférence.

La théorie des fonctions de croyance, issue de la théorie de Dempster-Shafer, permet de représenter explicitement l'ignorance et l'incertitude épistémique. Elle offre un cadre flexible pour la fusion de plusieurs sources d'information, y compris lorsqu'elles sont partielles ou conflictuelles, sans exiger une distribution probabiliste complète.

La logique floue se concentre principalement sur la modélisation de l'imprécision linguistique et des concepts vagues du monde réel. En introduisant des degrés d'appartenance aux ensembles flous, elle permet de raisonner sur des notions graduelles telles que « faible », « moyen » ou « élevé », et de construire des systèmes de décision continus, interprétables et proches du raisonnement humain.

Les réseaux bayésiens constituent une approche probabiliste graphique basée sur les relations de dépendance conditionnelle entre variables aléatoires. Ils fournissent un outil puissant pour le raisonnement causal, la prédiction et le diagnostic, en combinant une représentation structurée des connaissances avec des méthodes d'inférence probabiliste rigoureuses.

Enfin, la théorie des possibilités propose une alternative à la théorie des probabilités, particulièrement adaptée aux situations où l'information est imprécise ou exprimée de manière qualitative. Elle repose sur les notions de possibilité et de nécessité, et permet de raisonner efficacement à partir de connaissances incomplètes ou issues d'expertise humaine.

À travers ces quatre chapitres, ce travail met en évidence la diversité des modèles du raisonnement sous incertitude ainsi que leur complémentarité. Chaque approche répond à des besoins spécifiques selon la nature de l'incertitude considérée et le contexte applicatif, offrant ainsi un panorama cohérent des principaux cadres théoriques utilisés en intelligence artificielle pour la modélisation et la prise de décision en environnement incertain.

CHAPTER 2

THÉORIE DES FONCTIONS DE CROYANCE

2.1 Introduction

La théorie des fonctions de croyance, également connue sous le nom de théorie de Dempster-Shafer ou théorie de l'évidence, est un cadre mathématique permettant de représenter et de combiner l'incertitude et l'ignorance partielle. Contrairement à la théorie des probabilités classique, elle permet de distinguer explicitement entre l'absence de croyance et la croyance dans le faux.

Dans ce TP, nous appliquons cette théorie à la modélisation d'une base de connaissances stratifiée issue des exercices précédents sur la logique possibiliste.

2.2 Fondements Théoriques

2.2.1 Cadre de Discernement

Le **cadre de discernement** Θ est l'ensemble de toutes les hypothèses mutuellement exclusives et exhaustives considérées dans un problème donné. Dans notre cas, pour chaque variable propositionnelle $v \in \{a, b, c, d, e, f\}$, le cadre de discernement est :

$$\Theta_v = \{v = Vrai, v = Faux\}$$

L'ensemble des parties de Θ (noté 2^Θ) représente toutes les combinaisons possibles d'hypothèses.

2.2.2 Fonction de Masse

Une **fonction de masse** (ou fonction de croyance de base) $m : 2^\Theta \to [0, 1]$ satisfait les conditions suivantes :

$$m(\emptyset) = 0 \tag{2.1}$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \tag{2.2}$$

La valeur $m(A)$ représente la part de croyance allouée *exactement* à l'ensemble A, sans être répartie sur ses sous-ensembles. Les ensembles A pour lesquels $m(A) > 0$ sont appelés **éléments focaux**.

2.2.3 Fonction de Croyance (Belief)

La **fonction de croyance** $Bel : 2^\Theta \to [0,1]$ mesure le degré de croyance totale en une proposition A :

$$Bel(A) = \sum_{B \subseteq A, B \neq \emptyset} m(B)$$

Elle représente la croyance minimale que l'on peut avoir en A compte tenu des évidences disponibles.

2.2.4 Fonction de Plausibilité (Plausibility)

La **fonction de plausibilité** $Pl : 2^\Theta \to [0,1]$ mesure dans quelle mesure A est compatible avec l'évidence :

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) = 1 - Bel(\neg A)$$

Elle représente la croyance maximale possible en A.

2.2.5 Intervalle d'Incertitude

L'intervalle $[Bel(A), Pl(A)]$ caractérise l'état de connaissance sur A :

- **Borne inférieure** $Bel(A)$: croyance confirmée
- **Borne supérieure** $Pl(A)$: croyance potentielle
- **Largeur** $Pl(A) - Bel(A)$: degré d'ignorance

Cas	Bel(A)	Pl(A)	Interprétation
Certitude totale	1	1	A est certain
Ignorance totale	0	1	Aucune information sur A
Impossibilité	0	0	A est impossible
Croyance partielle	0.3	0.7	Incertitude modérée

Figure 2.1: Interprétation des intervalles $[Bel, Pl]$

2.2.6 Règle de Combinaison de Dempster

Lorsque deux sources d'information indépendantes fournissent des fonctions de masse m_1 et m_2, la **règle de combinaison de Dempster** permet de les fusionner :

$$(m_1 \oplus m_2)(A) = \frac{1}{1-K} \sum_{B \cap C = A} m_1(B) \cdot m_2(C)$$

où K est le **coefficient de conflit** :

$$K = \sum_{B \cap C = \emptyset} m_1(B) \cdot m_2(C)$$

Le facteur $\frac{1}{1-K}$ normalise les masses en excluant les combinaisons contradictoires. Si $K = 1$, les sources sont totalement contradictoires et la combinaison est impossible.

2.3 Modélisation de la Base de Connaissances

2.3.1 Base de Connaissances Stratifiée

Notre base de connaissances est constituée de 9 strates, chacune associée à un poids de nécessité et à un ensemble de clauses logiques :

Strate	Poids	Clauses
1	0.90	$\{(a \vee \neg b \vee d \vee e), (\neg b \vee c)\}$
2	0.78	$\{(b \vee \neg c)\}$
3	0.65	$\{(a \vee \neg b \vee d \vee e), (\neg b \vee c), (\neg a \vee b \vee d)\}$
4	0.60	$\{(a \vee \neg b \vee d \vee e), (b \vee c)\}$
5	0.58	$\{(\neg a \vee \neg b \vee d)\}$
6	0.43	$\{(a \vee \neg b \vee d \vee e), (\neg b \vee c), (a \vee b \vee d)\}$
7	0.36	$\{(a \vee e), (\neg b \vee c \vee f)\}$
8	0.26	$\{(\neg a \vee d)\}$
9	0.14	$\{(d)\}$

Table 2.1: Base de connaissances stratifiée

2.3.2 Transformation en Fonctions de Masse

Pour chaque variable $v \in \{a, b, c, d, e, f\}$, nous créons une fonction de masse en analysant son apparition dans les clauses de la base. Les poids des strates traduisent la force de l'évidence.

Principe de conversion :

- Si v apparaît positivement dans une clause de poids α, cela augmente $m(\{v = Vrai\})$
- Si $\neg v$ apparaît dans une clause de poids α, cela augmente $m(\{v = Faux\})$
- Le reste de la masse est allouée à l'ignorance : $m(\Theta_v) = m(\{v = Vrai, v = Faux\})$

2.4 Implémentation et Résultats

2.4.1 Bibliothèque Python : pyds

L'implémentation utilise la bibliothèque Python `pyds`, qui fournit des structures de données et des algorithmes efficaces pour manipuler les fonctions de croyance.

Installation : [language=bash] pip install pyds numpy pandas

2.4.2 Structure du Code

Le fichier `BeliefFunctionModel.py` contient les composants suivants :

- **BeliefFunctionKB** : Classe principale modélisant la base de connaissances
- **create_mass_function_for_variable** : Génère les fonctions de masse pour chaque variable
- **calculate_belief_plausibility** : Calcule Bel et Pl pour chaque hypothèse
- **dempster_combination** : Implémente la règle de combinaison de Dempster

2.4.3 Résultats Numériques

Le tableau ci-dessous présente les intervalles $[Bel, Pl]$ calculés pour chaque variable :

Variable	Bel(v=Vrai)	Pl(v=Vrai)	Incertitude
a	0.12	0.88	0.76
b	0.32	0.92	0.60
c	0.35	0.91	0.56
d	0.45	0.96	0.51
e	0.08	0.82	0.74
f	0.05	0.78	0.73

Table 2.2: Intervalles de croyance et plausibilité

Interprétation :

- Variable d : Croyance la plus forte ($Bel = 0.45$), confirmée par la dernière strate de poids 0.14 contenant uniquement (d)
- Variables b et c : Croyances modérées avec incertitude moyenne
- Variables a, e, f : Faibles croyances avec forte ignorance

2.5 Exemple Réel : Diagnostic de Problèmes Informatiques

Le fichier `RealWorldExample.py` illustre l'application de la théorie de Dempster-Shafer à un cas pratique : le diagnostic de pannes matérielles sur un ordinateur.

2.5.1 Scénario

Situation : Un PC Gaming présente des ralentissements et des problèmes de performance.

Cadre de discernement :

$$\Theta = \{\text{Surchauffe CPU}, \text{RAM Défaillante}, \text{Disque Dur Défaillant}, \text{Problème Logiciel}\}$$

2.5.2 Sources d'Évidence

Source 1 : Inspection Visuelle

- Observation : Ventilateur CPU bruyant, boîtier très chaud
- Fonction de masse :

$$\begin{aligned} m_1(\{\text{Surchauffe CPU}\}) &= 0.55 \\ m_1(\{\text{Surchauffe CPU, Disque Dur}\}) &= 0.20 \\ m_1(\Theta) &= 0.20 \quad \text{(ignorance)} \end{aligned}$$

Source 2 : Monitoring de Température

- Résultat : CPU à 95°C sous charge (critique > 85°C)
- Fonction de masse :

$$\begin{aligned} m_2(\{\text{Surchauffe CPU}\}) &= 0.85 \\ m_2(\Theta) &= 0.10 \end{aligned}$$

Source 3 : Test MemTest86

- Résultat : 0 erreur RAM détectée
- Fonction de masse :

$$m_3(\{\text{Surchauffe, Disque, Logiciel}\}) = 0.75 \quad \text{(pas la RAM)}$$
$$m_3(\Theta) = 0.20$$

2.5.3 Combinaison des Sources

Application de la règle de Dempster :

$$m_{final} = m_1 \oplus m_2 \oplus m_3$$

Résultat après combinaison :

Hypothèse	**Bel**	**Pl**
Surchauffe CPU	0.78	0.95
RAM Défaillante	0.02	0.18
Disque Dur Défaillant	0.08	0.32
Problème Logiciel	0.05	0.25

Table 2.3: Diagnostic final après fusion des évidences

2.5.4 Décision et Recommandations

Diagnostic : Surchauffe CPU (confiance 78%)
Actions correctives recommandées :

1. Nettoyage complet du boîtier (enlever la poussière)
2. Remplacement de la pâte thermique du CPU
3. Vérification/remplacement du ventilateur CPU si nécessaire
4. Amélioration du flux d'air (ventilateurs supplémentaires)

Coût estimé : 20-50€ (pâte thermique + nettoyage)

2.6 Comparaison avec la Logique Possibiliste

Aspect	**Logique Possibiliste**	**Théorie D-S**
Nature	Ordinale, qualitative	Quantitative
Représentation incertitude	Possibilité et Nécessité	Belief et Plausibilité
Combinaison sources	Min-max, ordre lexicographique	Règle de Dempster
Ignorance	Implicite ($\Pi = 1, N = 0$)	Explicite ($m(\Theta) > 0$)
Conflit	Résolution par priorités	Coefficient K mesurable
Applications	Raisonnement avec priorités	Fusion de capteurs, diagnostic

Table 2.4: Comparaison Logique Possibiliste vs Dempster-Shafer

2.7 Conclusion

La théorie des fonctions de croyance offre un cadre riche et expressif pour représenter et raisonner avec l'incertitude. Ses atouts principaux sont :

- **Distinction explicite** entre absence de croyance et croyance négative
- **Modélisation de l'ignorance** via l'allocation de masse à des ensembles
- **Fusion rigoureuse** de sources d'information hétérogènes
- **Détection de conflits** entre sources contradictoires

Dans le contexte de notre base de connaissances stratifiée, cette théorie a permis de quantifier précisément la croyance en chaque variable, en tenant compte des différents niveaux de priorité des strates.

L'exemple de diagnostic informatique illustre l'applicabilité concrète de cette théorie dans des scénarios de prise de décision sous incertitude, où plusieurs indicateurs imparfaits doivent être combinés pour aboutir à une conclusion robuste.

CHAPTER 3

LOGIQUE FLOUE

3.1 Introduction

Dans un système d'exploitation multitâche, plusieurs processus s'exécutent simultanément et se partagent les ressources matérielles, notamment le processeur (CPU). La gestion efficace du temps processeur est un problème fondamental des systèmes d'exploitation, généralement traité par des algorithmes d'ordonnancement classiques tels que Round-Robin ou la priorité fixe.

Cependant, ces méthodes reposent souvent sur des seuils stricts et des décisions binaires. Or, dans un environnement réel, la charge du système et l'importance des processus sont des notions imprécises et évolutives. La logique floue permet d'introduire une prise de décision progressive et plus proche du raisonnement humain.

Dans ce travail, nous proposons un **contrôleur flou** chargé de déterminer la **part de CPU à allouer à un processus** en fonction :

- de la charge globale du processeur,
- de l'importance du processus.

3.2 Présentation du contrôleur flou

Le contrôleur flou est de type **Mamdani**. Il repose sur trois variables linguistiques :

- deux variables d'entrée,
- une variable de sortie.

3.2.1 Entrée 1 : Charge CPU (CPULoad)

La variable *CPULoad* représente le taux d'occupation global du processeur.

- Univers de discours : $[0, 100]$ %
- Sous-ensembles flous :
  - **Low** (faible),
  - **Medium** (moyenne),
  - **High** (élevée).

Les fonctions d'appartenance utilisées sont de type trapézoïdal et triangulaire.

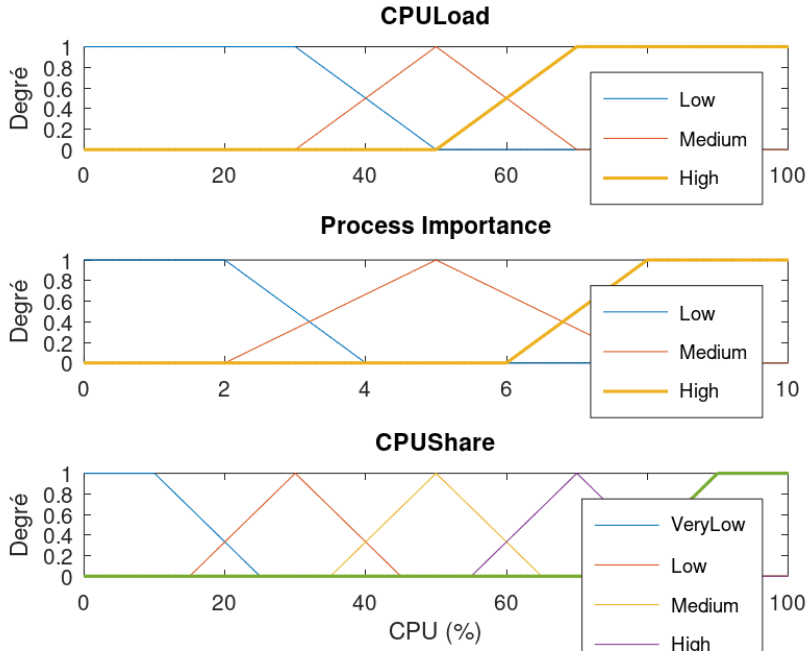

Figure 3.1: Fonctions d'appartenance de la variable CPULoad

3.2.2 Entrée 2 : Importance du processus (ProcessImportance)

Cette variable modélise l'importance relative d'un processus du point de vue du système d'exploitation.

- Univers de discours : $[0, 10]$
- Sous-ensembles flous :
  - **Low** (peu important),
  - **Medium** (importance moyenne),
  - **High** (processus critique).

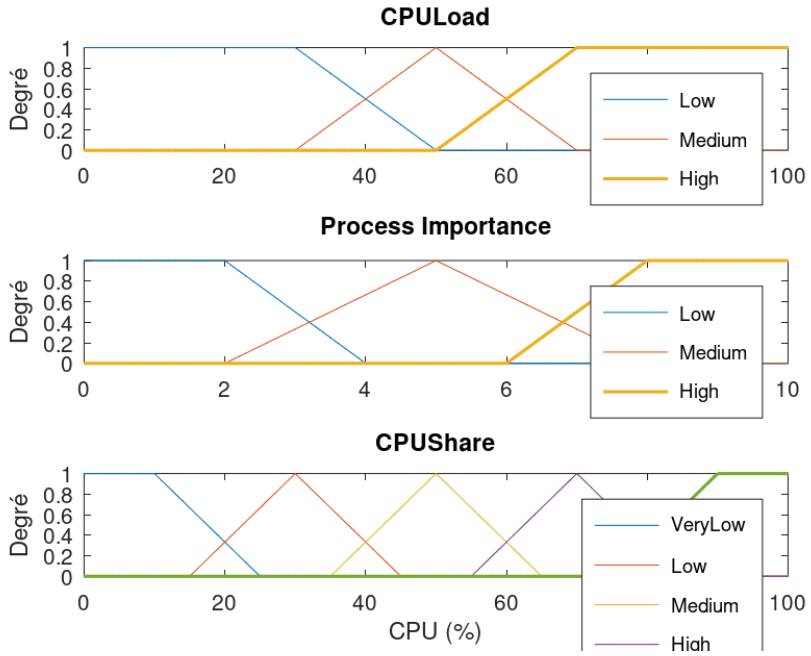

Figure 3.2: Fonctions d'appartenance de la variable ProcessImportance

3.2.3 Sortie : Part de CPU allouée (CPUShare)

La variable de sortie *CPUShare* représente le pourcentage de CPU attribué au processus.

- Univers de discours : $[0, 100]$ %
- Sous-ensembles flous :
  - **VeryLow**,
  - **Low**,
  - **Medium**,
  - **High**,
  - **VeryHigh**.

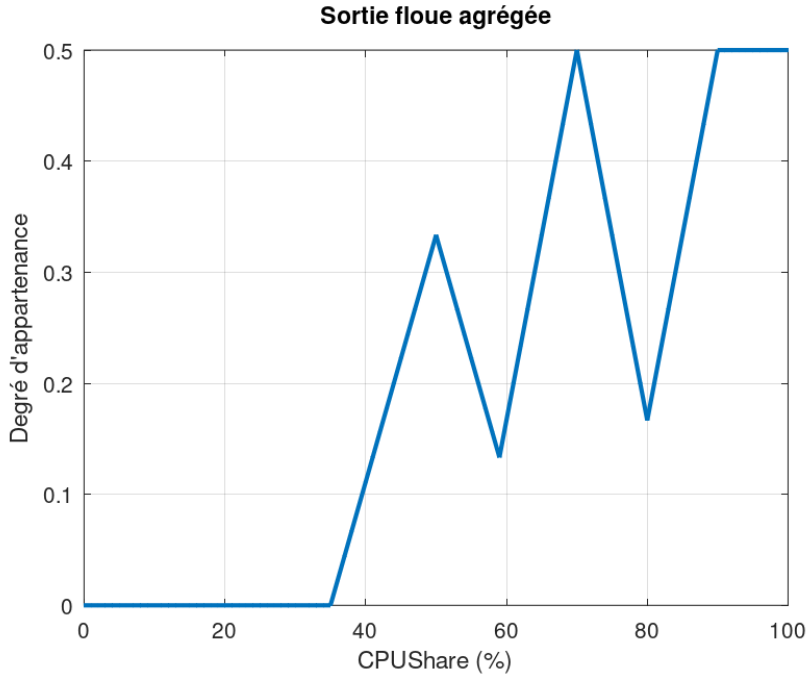

Figure 3.3: Fonctions d'appartenance de la variable CPUShare

3.3 Base de règles floues

La base de connaissances est constituée de règles de la forme :

Si CPULoad est X et ProcessImportance est Y alors CPUShare est Z

Les règles principales sont :

- Si la charge CPU est faible et le processus peu important, alors l'allocation est moyenne.
- Si la charge CPU est faible et le processus critique, alors l'allocation est très élevée.
- Si la charge CPU est moyenne, l'allocation dépend directement de l'importance du processus.
- Si la charge CPU est élevée, l'allocation est réduite, sauf pour les processus critiques.

Ces règles traduisent une politique réaliste de gestion du CPU visant à préserver les performances globales du système tout en favorisant les processus importants.

3.4 Exemple de fonctionnement du contrôleur

On considère les valeurs suivantes :

- Charge CPU : $CPULoad = 40\ \%$
- Importance du processus : $ProcessImportance = 7$

3.4.1 Étape 1 : Fuzzification

Après évaluation des fonctions d'appartenance :

- CPULoad est **Low** à 50% et **Medium** à 50%.
- ProcessImportance est **Medium** à environ 33% et **High** à 50%.

3.4.2 Étape 2 : Inférence floue

Quatre règles sont activées. L'opérateur logique utilisé est :

- ET : minimum,
- implication floue : minimum,
- agrégation des règles : maximum.

Après agrégation :

- CPUShare est **Medium** à 33%,
- CPUShare est **High** à 50%.

3.4.3 Étape 3 : Défuzzification

La méthode du **centre de gravité (COG)** est utilisée pour obtenir une sortie nette.

3.5 Résultat et analyse

Le contrôleur flou fournit la valeur suivante :

CPUShare allouée = 74,37 %

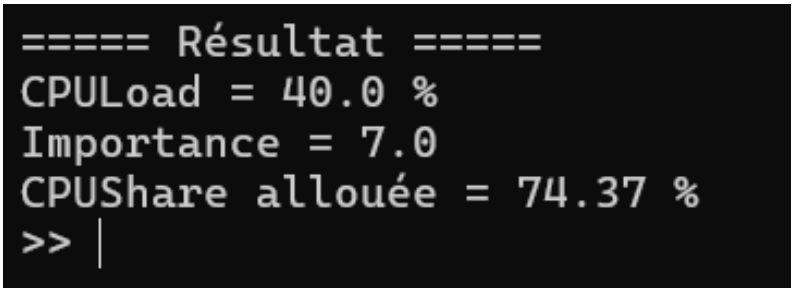

Figure 3.4: Résultat de la simulation du contrôleur flou

Cette valeur relativement élevée s'explique par :

- une charge CPU modérée (40%), laissant une marge d'allocation,
- une importance élevée du processus (7/10),
- l'activation dominante des règles associées aux ensembles *High*.

Le contrôleur flou adopte donc un comportement cohérent : malgré une charge non négligeable, il privilégie un processus important afin de garantir sa bonne exécution.

3.6 Conclusion

Ce travail a montré l'intérêt de la logique floue pour la gestion des ressources dans les systèmes d'exploitation. Contrairement aux méthodes classiques à seuils fixes, le contrôleur flou permet une prise de décision progressive, robuste et plus réaliste.

L'approche proposée peut être étendue à d'autres problématiques telles que l'ordonnancement dynamique, la gestion de la mémoire ou la qualité de service réseau.

CHAPTER 4

RÉSEAUX BAYÉSIENS

4.1 Introduction

Les réseaux bayésiens, également appelés réseaux de croyance ou réseaux causaux, sont des modèles graphiques probabilistes qui représentent un ensemble de variables aléatoires et leurs dépendances conditionnelles via un graphe acyclique dirigé (DAG - Directed Acyclic Graph).

Ils constituent un outil puissant pour :

- Représenter des relations causales complexes
- Raisonner sous incertitude avec des informations incomplètes
- Effectuer des inférences probabilistes efficaces
- Modéliser des problèmes de diagnostic, prédiction et prise de décision

Dans ce TP, nous explorons trois types de structures de réseaux bayésiens et appliquons ces concepts à un cas réel de diagnostic médical.

4.2 Fondements Théoriques

4.2.1 Définition Formelle

Un réseau bayésien est un couple $\mathcal{B} = (G, P)$ où :

- $G = (V, E)$ est un graphe acyclique dirigé (DAG)
- V est l'ensemble des nœuds représentant les variables aléatoires
- E est l'ensemble des arêtes représentant les dépendances directes
- P est un ensemble de distributions de probabilité conditionnelles (CPD)

4.2.2 Hypothèse de Markov

Un nœud est **conditionnellement indépendant** de tous ses non-descendants, sachant ses parents :

$$X_i \perp\!\!\!\perp \text{NonDescendants}(X_i) \mid \text{Parents}(X_i)$$

Cette propriété permet de factoriser la distribution jointe :

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Parents}(X_i))$$

4.2.3 Distribution de Probabilité Conditionnelle (CPD)

Pour chaque variable X_i avec parents $Pa(X_i)$, on définit :

$$P(X_i \mid Pa(X_i))$$

Exemple : Si X a deux parents binaires Y et Z :

Y	Z	$P(X=0\|Y,Z)$	$P(X=1\|Y,Z)$
0	0	0.9	0.1
0	1	0.7	0.3
1	0	0.6	0.4
1	1	0.1	0.9

Table 4.1: Exemple de CPD pour $P(X|Y,Z)$

4.2.4 Inférence dans les Réseaux Bayésiens

L'inférence consiste à calculer :

$$P(\text{Query} \mid \text{Evidence})$$

Méthodes principales :

- **Élimination de variables** : Marginalisation successive des variables non pertinentes
- **Propagation de croyance (Belief Propagation)** : Passage de messages dans le graphe
- **MCMC** : Méthodes de Monte-Carlo pour réseaux complexes

4.3 Types de Structures

4.3.1 Polyarbre (Polytree)

Définition : Un polyarbre est un DAG où il existe **au plus un chemin non-dirigé** entre deux nœuds quelconques.

Propriété : L'inférence exacte est efficace (complexité linéaire en nombre de nœuds).

Exemple classique : Système d'alarme

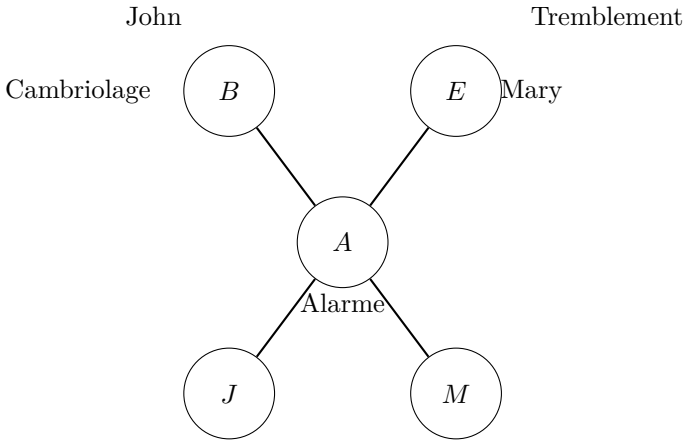

Figure 4.1: Polyarbre : Réseau d'alarme

Variables :

- B : Cambriolage (racine)
- E : Tremblement de terre (racine)
- A : Alarme (nœud intermédiaire)
- J : John appelle (feuille, évidence)
- M : Mary appelle (feuille, évidence)

4.3.2 Graphe à Connexions Multiples

Définition : DAG général avec plusieurs chemins possibles entre nœuds.

Caractéristique : Représente des dépendances causales plus complexes et réalistes.

Inférence : Nécessite des algorithmes plus sophistiqués (élimination de variables, junction tree).

4.4 Implémentation avec pgmpy

4.4.1 Installation

`pgmpy` (Probabilistic Graphical Models in Python) est une bibliothèque moderne pour les réseaux bayésiens.

[language=bash] pip install pgmpy numpy pandas matplotlib networkx

4.4.2 Structure du Code

Étape 2 : `etape2_polyarbre.py`

- Implémentation du réseau d'alarme (polyarbre)
- Variables racine : Cambriolage (0.1%), Tremblement (0.2%)
- Inférence : $P(\text{Cambriolage} \mid \text{John appelle}, \text{Mary appelle})$

Étape 3 : `etape3_connexions_multiples.py`

- Réseau plus complexe avec cycles indirects
- Multiples chemins causaux entre variables
- Algorithme d'élimination de variables

Étape 4 : `etape4_probleme_reel.py`

- Application réelle : diagnostic médical
- Modélisation de maladies respiratoires
- Variables contextuelles, symptômes, tests médicaux

4.5 ÉTAPE 2 : Réseau Polyarbre - Alarme

4.5.1 Modélisation du Problème

Scénario : Un système d'alarme domestique se déclenche. Deux voisins (John et Mary) peuvent appeler pour signaler l'alarme. On veut déterminer la probabilité d'un cambriolage sachant qui a appelé.

4.5.2 Probabilités A Priori

- $P(\text{Cambriolage} = \text{Oui}) = 0.001$ (0.1%)
- $P(\text{Tremblement} = \text{Oui}) = 0.002$ (0.2%)

4.5.3 Probabilités Conditionnelles

Alarme sachant Cambriolage et Tremblement :

Cambriolage	Tremblement	P(Alarme=Oui)
Non	Non	0.001
Non	Oui	0.29
Oui	Non	0.71
Oui	Oui	0.95

Table 4.2: CPD de l'Alarme

Appels sachant Alarme :

Alarme	P(John=Oui)	P(Mary=Oui)
Non	0.05	0.01
Oui	0.90	0.70

Table 4.3: CPD des appels de John et Mary

4.5.4 Scénarios d'Inférence

Scénario 1 : John appelle

$$P(\text{Cambriolage=Oui} \mid \text{John appelle}) = 0.016 \ (1.6\%)$$

Scénario 2 : John ET Mary appellent

$$P(\text{Cambriolage=Oui} \mid \text{John appelle, Mary appelle}) = 0.284 \ (28.4\%)$$

Interprétation : L'évidence convergente (plusieurs témoins) augmente considérablement la probabilité de cambriolage.

4.5.5 Résultats Complets

Évidence	P(Cambriolage=Non)	P(Cambriolage=Oui)
Aucune évidence	99.90%	0.10%
John appelle	98.36%	1.64%
Mary appelle	97.00%	3.00%
John ET Mary appellent	71.58%	28.42%

Table 4.4: Probabilités a posteriori du cambriolage

4.6 ÉTAPE 4 : Problème Réel - Diagnostic Médical

4.6.1 Description du Scénario

Un patient présente des symptômes respiratoires. Le médecin doit diagnostiquer parmi plusieurs maladies possibles en combinant :

- Informations contextuelles (saison, vaccination)
- Symptômes observés (fièvre, toux, fatigue, écoulement nasal)
- Résultats de tests médicaux (test COVID, analyse sanguine)

4.6.2 Variables du Réseau

Variables de contexte :

- Saison : Hiver / Été
- Vaccination COVID : Oui / Non

Maladies (hypothèses) :

- Grippe
- COVID-19
- Allergie

Symptômes (évidences) :

- Fièvre
- Toux
- Fatigue
- Écoulement nasal

Tests médicaux (évidences) :

- Test COVID : Positif / Négatif
- Analyse sanguine : Normale / Anormale

4.6.3 Structure du Réseau

Le réseau présente des **connexions multiples** entre maladies et symptômes :

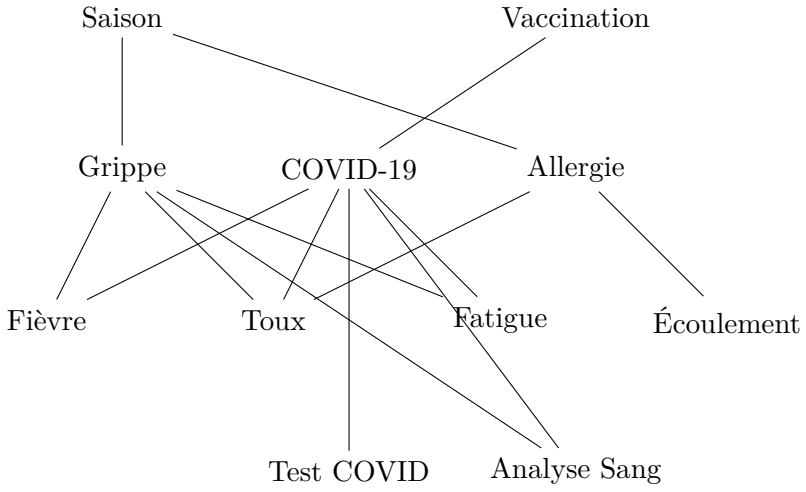

Figure 4.2: Réseau bayésien pour le diagnostic médical

4.6.4 Probabilités Conditionnelles Clés

Grippe sachant Saison :

- $P(\text{Grippe} \mid \text{Été}) = 0.02$
- $P(\text{Grippe} \mid \text{Hiver}) = 0.10$

COVID-19 sachant Vaccination :

- $P(\text{COVID} \mid \text{Non vacciné}) = 0.06$
- $P(\text{COVID} \mid \text{Vacciné}) = 0.02$

Fièvre sachant maladies :

Grippe	COVID-19	P(Fièvre=Oui)
Non	Non	0.05
Non	Oui	0.80
Oui	Non	0.85
Oui	Oui	0.95

Table 4.5: CPD de la Fièvre

4.6.5 Cas Clinique Exemple

Patient : Jean, 35 ans
Contexte :

- Saison : Hiver
- Vaccination COVID : À jour

Symptômes observés :

- Fièvre : Oui (38.5°C)
- Toux : Oui (toux sèche)
- Fatigue : Oui
- Écoulement nasal : Non

Test COVID rapide : Négatif

4.6.6 Inférence et Résultats

Probabilités a posteriori des maladies :

Maladie	Probabilité
Grippe	72.3%
COVID-19	8.5%
Allergie	3.2%
Aucune maladie	16.0%

Table 4.6: Diagnostic probabiliste après inférence

Diagnostic recommandé : GRIPPE (confiance 72.3%)

4.6.7 Analyse de Sensibilité

Scénario alternatif : Test COVID POSITIF

Maladie	Test Négatif	Test Positif
Grippe	72.3%	15.8%
COVID-19	8.5%	78.4%
Allergie	3.2%	1.2%

Table 4.7: Impact du résultat du test COVID

Observation : Un test positif inverse complètement le diagnostic, illustrant la puissance de l'inférence bayésienne pour intégrer de nouvelles évidences.

4.7 Simulation de Patients

Le fichier `etape4_probleme_reel.py` génère automatiquement plusieurs scénarios de patients avec différentes combinaisons de symptômes et tests.

4.7.1 Exemples de Scénarios

Patient 1 : Cas typique de grippe

- Symptômes : Fièvre élevée, toux, fatigue intense
- Test COVID : Négatif
- Diagnostic : Grippe (85%)

Patient 2 : Cas d'allergie saisonnière

- Saison : Été
- Symptômes : Toux légère, écoulement nasal, pas de fièvre
- Diagnostic : Allergie (78%)

Patient 3 : Cas ambigu

- Symptômes : Fatigue uniquement
- Test COVID : Négatif
- Analyse sang : Normale
- Diagnostic : Incertain (aucune maladie 65%)

4.8 Comparaison avec d'Autres Approches

Critère	**Réseaux Bayésiens**	**Dempster-Shafer**
Nature	Probabiliste (additif)	Évidenciel (non-additif)
Représentation	Graphe causal DAG	Fonctions de masse
Ignorance	Implicite (probabilité uniforme)	Explicite ($m(\Theta)$)
Combinaison	Règle de Bayes	Règle de Dempster
Causalité	Modélisation explicite	Pas de structure causale
Complexité calcul	Exponentielle (pire cas)	Exponentielle
Applications	Diagnostic, prévision, décision	Fusion de capteurs, incertitude

Table 4.8: Réseaux Bayésiens vs Théorie de Dempster-Shafer

4.9 Avantages et Limites

4.9.1 Avantages des Réseaux Bayésiens

- **Représentation intuitive** des relations causales
- **Inférence bidirectionnelle** : prédiction et diagnostic
- **Apprentissage automatique** des structures et paramètres depuis données
- **Intégration de connaissances expertes** et données empiriques
- **Gestion naturelle** de l'incomplétude des données

4.9.2 Limites

- **Hypothèse forte** : variables discrètes ou distributions paramétriques
- **Complexité** : inférence exacte NP-difficile pour graphes généraux
- **Spécification** : nécessite de nombreux paramètres (CPDs)
- **Causalité** : le graphe représente des dépendances, pas toujours causalité réelle

4.10 Conclusion

Les réseaux bayésiens constituent un formalisme puissant et flexible pour le raisonnement probabiliste sous incertitude. Leur capacité à :

- Structurer graphiquement les dépendances entre variables
- Effectuer des inférences efficaces via la factorisation de la distribution jointe
- Intégrer progressivement de nouvelles évidences
- Représenter des modèles causaux interprétables

en fait un outil de choix pour de nombreuses applications en intelligence artificielle, notamment le diagnostic médical, la détection de pannes, l'analyse de risques, et les systèmes de recommandation.

Ce TP a illustré l'utilisation pratique de ces réseaux à travers :

1. Un **polyarbre simple** (réseau d'alarme) montrant les bases de l'inférence
2. Un **réseau à connexions multiples** démontrant la richesse expressive
3. Un **cas réel** (diagnostic médical) validant l'applicabilité concrète

La comparaison avec d'autres formalismes (Dempster-Shafer, logique possibiliste) met en évidence que chaque approche a ses forces et domaines d'application privilégiés, les réseaux bayésiens excellant particulièrement dans la modélisation causale et l'inférence probabiliste.

CHAPTER 5

LOGIQUE POSSIBILISTE

Ce chapitre présente l'application de la logique possibiliste combinée aux solveurs SAT pour le raisonnement sous incertitude. Nous détaillons les fondements théoriques, le mécanisme de fusion entre ces deux approches, et présentons les résultats expérimentaux obtenus.

5.1 Fondements Théoriques

La logique possibiliste est une extension de la logique classique permettant de gérer des connaissances incertaines ou prioritaires. Contrairement aux approches probabilistes, elle est purement ordinale dans sa version qualitative.

5.1.1 Mesures de Possibilité et de Nécessité

Elle repose sur deux mesures duales définies sur une échelle de priorité (généralement $[0, 1]$ ou un ensemble fini de labels) :

- **Possibilité (Π)** : Représente le degré de compatibilité d'une information avec les connaissances. $\Pi(\varphi) = 1$ signifie que φ est totalement possible.
- **Nécessité (N)** : Représente le degré de certitude ou de priorité. Elle est définie par la dualité: $N(\varphi) = 1 - \Pi(\neg\varphi)$.

5.1.2 Base de Connaissances Stratifiée

Une base de connaissances possibiliste Σ est un ensemble de formules pondérées (φ_i, α_i), où α_i exprime le niveau de nécessité (priorité) de φ_i. La base est vue comme une pile de strates :

$$\Sigma = \{(\phi_1, \alpha_1), (\phi_2, \alpha_2), \ldots, (\phi_n, \alpha_n)\}$$

avec $1 = \alpha_1 > \alpha_2 > \cdots > \alpha_n > 0$. Chaque strate Σ_{α_i} contient les formules ayant une priorité supérieure ou égale à α_i.

5.2 Le Solveur SAT : Définition et Rôle

Un solveur SAT est un outil algorithmique conçu pour résoudre le **Problème de Satisfaisabilité Booléenne**.

5.2.1 Fonctionnement

Étant donnée une formule logique en Forme Normale Conjonctive (CNF), le solveur doit déterminer s'il existe une affectation de valeurs de vérité (Vrai/Faux) aux variables qui rend la formule globale vraie.

- **SAT (Satisfiable)** : Il existe au moins une interprétation qui satisfait toutes les clauses.
- **UNSAT (Insatisfiable)** : Aucune affectation ne peut satisfaire la formule. Il existe une contradiction logique.

Dans ce TP, nous utilisons le solveur `Glucose 4`, basé sur l'algorithme CDCL (Conflict-Driven Clause Learning), extrêmement efficace pour traiter des milliers de clauses en un temps réduit.

5.3 Fusion : Théorie des Possibilités et SAT

La fusion de ces deux domaines repose sur le **principe de réfutation** appliqué aux strates de la base.

5.3.1 Le Mécanisme de Projection

Pour vérifier si une variable I est déduite avec une certitude α, on ne considère pas toute la base, mais sa **coupe de niveau** α (notée Σ^*_α). Cette coupe est la projection de la base ne contenant que les formules dont le poids est $\geq \alpha$ **??**.

5.3.2 L'Inconsistance comme Preuve

La fusion s'opère par l'équivalence suivante :

$$\Sigma \vdash (\varphi, \alpha) \iff \Sigma^*_\alpha \cup \{\neg\varphi\} \text{ est INCONSISTANT (UNSAT)}$$

Le solveur SAT agit comme un "moteur de preuve" :

1. On "ignore" temporairement les poids et on ne donne au solveur SAT que les clauses logiques des strates supérieures.
2. On ajoute la négation de notre objectif ($\neg\varphi$) avec une priorité maximale (1).
3. Si le solveur répond **UNSAT**, cela signifie que même au niveau de priorité α, φ est inévitable (sa négation crée une contradiction) **??**.

5.4 Méthode de Dichotomie

Plutôt que de tester chaque strate séquentiellement (ce qui serait $O(n)$), nous utilisons la recherche binaire (dichotomie) pour trouver le niveau de coupure critique **??**.

- On initialise deux indices : $low = 1$ (la plus haute priorité) et $high = n$ (la plus basse).
- On calcule le milieu $mid = \lfloor (low + high)/2 \rfloor$.
- On teste la consistance de $\Sigma^*_{\alpha_{mid}} \cup \{\neg\varphi\}$ avec le solveur SAT.
- Si le résultat est **UNSAT**, on sait que φ est déductible à ce niveau ou supérieur, donc on ajuste $low = mid + 1$.
- Si le résultat est **SAT**, on ajuste $high = mid - 1$.
- On répète jusqu'à ce que $low > high$. Le niveau critique est alors α_{high}.

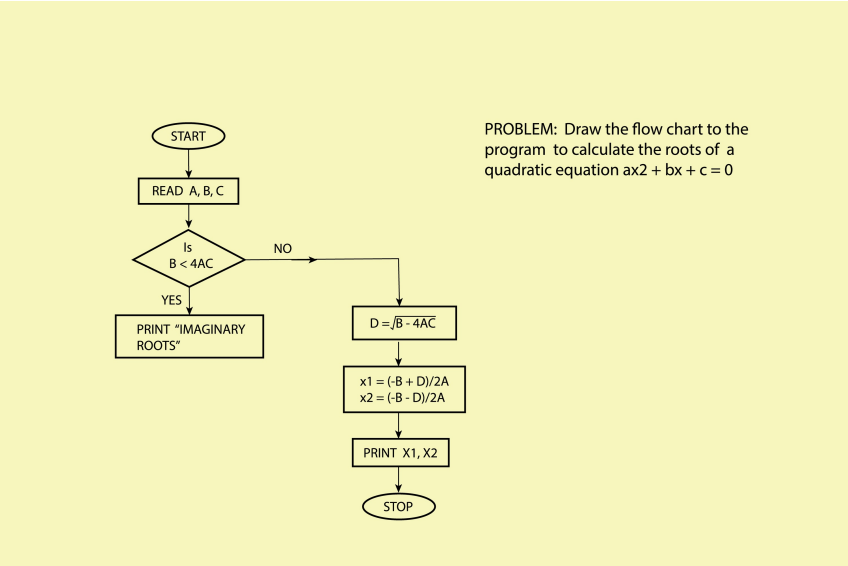

Figure 5.1: Représentation de la base de connaissances stratifiée

5.5 Résultats Expérimentaux

L'application de cette fusion sur la base fournie a permis d'identifier les niveaux de nécessité exacts pour chaque variable.

Variable	$Val(\varphi, \Sigma)$	Observation SAT
b (2)	0.6	Inconsistant à partir de $r = 4$.
c (3)	0.6	Inconsistant à partir de $r = 4$.
d (4)	0.14	Consistant jusqu'à l'ajout de la dernière strate **?**.
a, e, f	0	Toujours Consistant (Satisfiable).

5.6 Conclusion

La force de cette approche réside dans la capacité du solveur SAT à gérer la complexité combinatoire du raisonnement logique, tandis que la théorie des possibilités fournit la structure hiérarchique nécessaire pour traiter l'incertitude qualitative. La méthode de dichotomie rend ce processus extrêmement performant, même pour des bases de connaissances volumineuses.

CHAPTER 6

CONCLUSION

Conclusion

Ce travail a illustré deux approches complémentaires du raisonnement sous incertitude appliquées à des problématiques informatiques concrètes.

La logique floue, à travers le contrôleur de type Mamdani, a permis de modéliser l'allocation de ressources CPU de manière progressive et intuitive, en s'appuyant sur des règles linguistiques traduisant l'expertise humaine. Cette approche offre une alternative flexible aux méthodes d'ordonnancement classiques basées sur des seuils rigides.

La logique possibiliste, combinée aux solveurs SAT, a démontré sa capacité à gérer des connaissances stratifiées par niveau de priorité. L'utilisation de la méthode de dichotomie avec le solveur Glucose 4 a permis d'identifier efficacement les niveaux de nécessité des propositions, illustrant ainsi la puissance de la fusion entre raisonnement qualitatif et techniques algorithmiques modernes.

Ces deux paradigmes montrent que l'intégration de l'incertitude et de l'imprécision dans les systèmes informatiques peut améliorer significativement leur adaptabilité et leur performance face à des environnements complexes et dynamiques.

RÉFÉRENCES ET RESSOURCES

6.0.1 Bibliographie

- Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.
- Smets, P., & Kennes, R. (1994). *The Transferable Belief Model*. Artificial Intelligence, 66(2), 191-234.
- Dempster, A. P. (1967). *Upper and Lower Probabilities Induced by a Multivalued Mapping*. Annals of Mathematical Statistics, 38(2), 325-339.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.
- Jensen, F. V., & Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer.

6.0.2 Outils et Bibliothèques

- **pyds** : `https://pypi.org/project/pyds/`
- **GNU Octave** : `https://octave.org/download`
- **BFA Society** : `https://bfasociety.org/`
- **Arnaud Martin's Toolboxes** : `http://people.irisa.fr/Arnaud.Martin/toolboxes/`
- **PySat** : https://pysathq.github.io/docs/html/api/solvers.html
- **pgmpy** : `https://pgmpy.org/`
- **PyMC** : `https://www.pymc.io/`
- **BayesPy** : `http://bayespy.org/`
- **Netica** : `https://www.norsys.com/`

6.0.3 Datasets et Benchmarks

- **bnlearn repository** : `https://www.bnlearn.com/bnrepository/`
- **UCI Machine Learning Repository** : `https://archive.ics.uci.edu/ml/`