

Contents

1	Introduction	3
2	Réalisation	4
2.1	L'outil cygwin	4
2.2	Étape 1:	4
2.2.1	Utilisation du prod1vid.m	4
2.2.2	Réglage de paramètres	5
2.3	Étape 2	6
2.3.1	Explication	6
2.4	Étape 3	10
2.4.1	Génération automatique aléatoire des Graphes	10
2.5	Résultats	10
2.6	Observation pour chaque type	10
2.7	Observation et comparaison entre les différents types	10

List of Figures

2.1	Le changement des nb noeuds ainsi que nb parents max dans le code prodlevid.m	5
2.2	Une seule evidence	6
2.3	Deux evidences	6
2.4	exemple d'un graphe en sortie à l'exécution de prodlevid.m avec noeuds=80 et parents =10	7
2.5	exemple d'un graphe en sortie à l'exécution de prodlevid.m avec noeuds=80 et parents =10	8
2.6	Les commandes necessaires pour lancer l'inference	9
2.7	Lancement de passage et inference	9

Chapter 1

Introduction

bla bla for tp and le but of ir

Chapter 2

Réalisation

2.1 L'outil cygwin

why did we use it and all

2.2 Étape 1:

2.2.1 Utilisation du prod1vid.m

principalement prod1vid construit un réseau causal probabiliste basé sur le produit tel que les connexions entre les nœuds sont aléatoires , ainsi que les valeurs initiales attribués à la variable d'intérêt et l'évidence . Pour exécuter le programme il faut:

- sur Matlab taper : prod1vid

ce que le programme offre en sortie est environnement ou on peut voir toutes les variables et le graphe (matrice) crée. on peut alors afficher :

- la variable d'intérêt sachant l'évidence
- temps de la propagation
- type de graphe (multi-connected (multi-connectés) ou polytree (polyarbre))

Fonctionnement du programme

Après avoir étudié le programme on a pu résumer son fonctionnement dans les étapes qui suivent :

1. Initialisation du nombre de parents max globale et nombre de noeuds du graphe à construire
2. Création de liens de façon aléatoire entre les noeuds.
3. Utilisation de processus de fixation après la création aléatoire afin d'éviter les noeuds isolés et sous graphes isolés (les inconvénients de l'aléatoire)
4. Prise de considération des domaines des variables (représentés par les noeuds) cas binaire etc ...

5. Génération de la distribution aléatoire initiale du graphe crée (de possibilité initiales).
6. génération aléatoire d'une évidence : une évidence est une information nouvelle qui viens et à qui on aimerait calculer l'influence qu'elle aura sur la variable d'intérêt (évidente est comme une condition).
7. Détermination si le graphe est polytree ou multi-connected
8. Lancement de la propagation (algorithme de propagation)

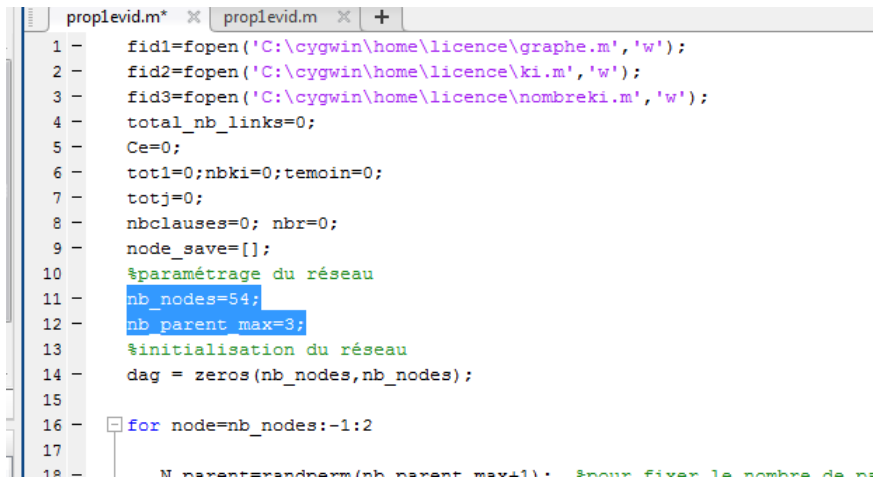
Concernant **Prodevid2** c'est le même fonctionnement à part qu'on a droit à deux évidences donc deux informations vont influencer notre réseau , en théorie on peut penser que la propagation des deux évidences prendra plus de temps que celle d'une seule , on testera ce cas dans ce qui suit .

2.2.2 Réglage de paramètres

Jeu de test

On a choisi de fixer le nombre de noeuds à :10
 et le nombre de parent max à :2
 ce qui est censé nous donnée un polytree .

Code



```

1 - fid1=fopen('C:\cygwin\home\licence\graphe.m','w');
2 - fid2=fopen('C:\cygwin\home\licence\ki.m','w');
3 - fid3=fopen('C:\cygwin\home\licence\nombreki.m','w');
4 - total_nb_links=0;
5 - Ce=0;
6 - totl=0;nbki=0;temoin=0;
7 - totj=0;
8 - nbclauses=0; nbr=0;
9 - node_save=[];
10 - %paramétrage du réseau
11 - nb_nodes=54;
12 - nb_parent_max=3;
13 - %initialisation du réseau
14 - dag = zeros(nb_nodes,nb_nodes);
15
16 - for node=nb_nodes:-1:2
17
18 - N_parents=randperm(nb_parent_max+1); %pour fixer le nombre de pa

```

Figure 2.1: Le changement des nb noeuds ainsi que nb parents max dans le code prodlevid.m

On a choisi de fixer le nombre de noeuds à : 54
 et le nombre de parents max à :3

Remarque Dans le code fourni Pour avoir le nombre de parents d'un nœud , le programme tire aléatoirement un nombre entre $[1, nb_parents_max + 1]$ donc concrètement le nombre de parents max est de 4 et non de 3 voir code:prodlevid.m.

Affichage

le résultat était le suivant :

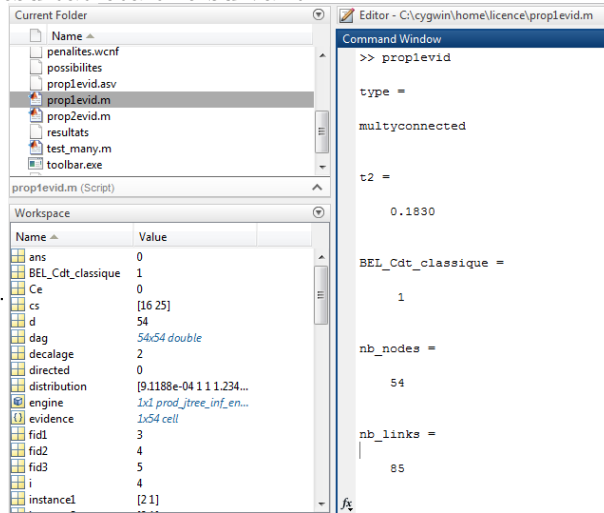


Figure 2.2: Une seule evidence

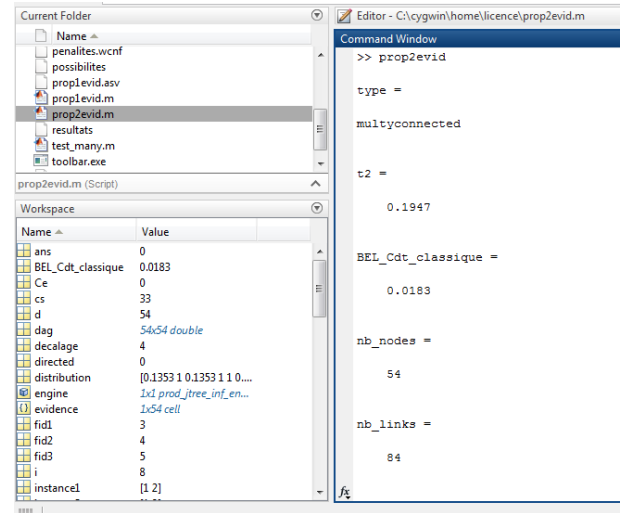


Figure 2.3: Deux evidences

nombre d'évidence	1	2
temps de propagation	0.18 secondes	0.19 secondes%
Bel variable d'intérêt	1	0.018

Explication et observation

Comme on a pu le deviner la propagation prend plus de temps avec deux évidences qu'avec une seule , car le calcul des nouvelles distributions de possibilités est plus complexe avec deux informations qui arrivent qu'avec une seule .

2.3 Étape 2

2.3.1 Explication

Dans cette étape il nous ai demandé d'utiliser deux programmes exécutables à fin de passer d'une modélisation graphique avec des algorithmes de propagation à une une représentation logique en clauses (max sat weighted) en utilisant le processus d'inférence. Prodevid1 donne en sortie un graphe (matrice en matlab) qui modélise le graphe crée

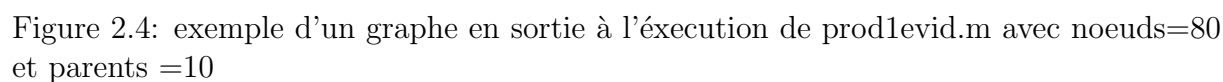


Figure 2.4: exemple d'un graphe en sortie à l'exécution de `prod1evid.m` avec `noeuds=80` et `parents=10`

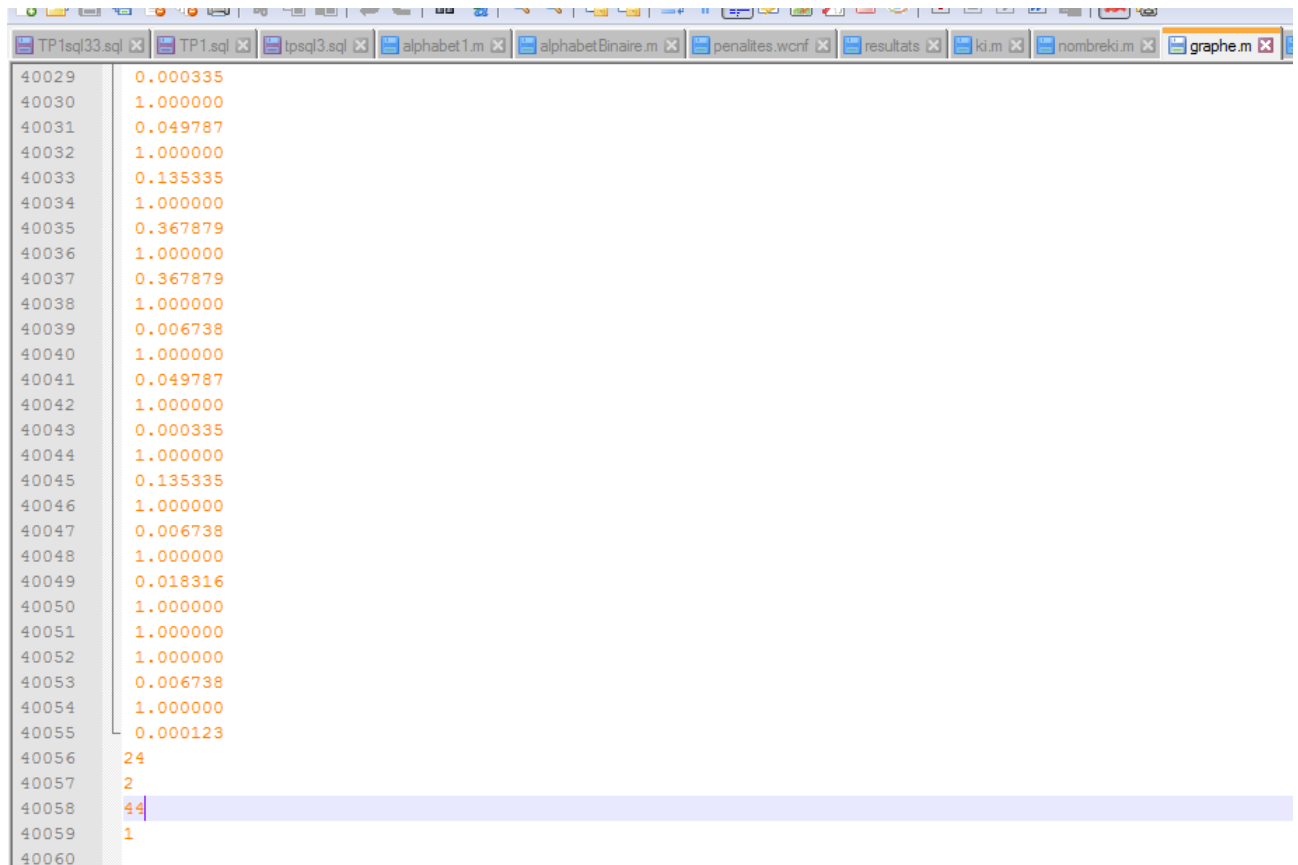


Figure 2.5: exemple d'un graphe en sortie à l'exécution de prod1evid.m avec noeuds=80 et parents =10

Tout d'abord on reprend l'exemple vu dans l'étape 1 avec un nombre de noeuds à 54 et nombre de parents max à 4 puis il faudra faire :

- Se placer dans le dossier licence sous terminal de cygwin .
- Exécuter le prod1evid sous matlab
- Exécuter le ./passage.exe qui convertit les graph.m en clauses (maxSat)
- Lancement du processus d'inférence on aura le temps de l'inférence en MICRO SECONDES.
- Affichage du résultat avec la commande : cat résultats

Affichage

```

/home/licence
master@master-PC ~
$ cd /home

master@master-PC /home
$ ls
ki.m  licence  master  nombreki.m

master@master-PC /home
$ cd master

master@master-PC ~
$ ls

master@master-PC ~
$ cd /home

master@master-PC /home
$ cd licence

master@master-PC /home/licence
$ ls
cygwin1.dll  ki.m          Pnt          prop2evid.m  wmaxsat
data         nombreki.m   possibilites resultats
graphe.m     passage.exe  prop1evid.asv test_many.m
inference.exe penalites.wcnf prop1evid.m  toolbar.exe

```

Figure 2.6: Les commandes nécessaires pour lancer l'inférence

```

master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference.exe
249600
master@master-PC /home/licence
$ ls resultats
resultats

master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 54
nombre de parentsmax: 3
l'evidence: -11
variable d'interet: 43
possibilité conditionnelle de l'interet | evidences 0.018316
temps de propagation 0.194687 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 190
la variable d'interet est inférée à partir de la base de pénalités
le coût de pénalité est de 4

```

Figure 2.7: Lancement de passage et inférence

Observation

on remarque dans notre cas on à affaire à un réseau multi-connected on a remarqué que la propagation était prenait presque autant de temps que l'inférence , on peut expliquer ça par le fait qu'on manipule des graphes où le nombres de noeuds et connexions n'est pas conséquent par contre dans le cas ou le nombre de noeuds était 80 et parents = 10 la propagation avait pris beaucoup plus de temps voir l'etape 3 comparé à l'inférence qui fait appel à un solver max sat .

2.4 Etape 3

2.4.1 Génération automatique aléatoire des Graphes

Génération des Polytree(youpaii)

Génération des Multiconnected

Génération des simplement connected

2.5 Résultats

NbrNœuds/NbrParents	Temps Propagation	Temps Inférence	Degrés Possibilité
25 Nœuds / 1 Parents	0.156598 sec	171601 milisec	0.049787
50 Nœuds / 1 Parents	0.282039 sec	218400 milisec	1
15 Nœuds / 3 Parents	0.052538 sec	156000 milisec	0.049787
25 Nœuds / 4 Parents	0.081438 sec	156000 milisec	1
25 Nœuds / 7 Parents	0.078056 sec	249600 milisec	1
25 Nœuds / 10 Parents	0 sec	0 milisec	0
30 Nœuds / 4 Parents	0 sec	0 milisec	1

2.6 Observation pour chaque type

2.7 Observation et comparaison entre les differents types