

UNIVERSITÉ DES SCIENCES ET DE LA
TECHNOLOGIE HOUARI BOUMEDIENE



RCR2

Rapport du TP
Inférence logique et propagation
graphique

Rédaction:

MOULAI HASSINA SAFAA

Matricule : 201400007564

HOUACINE NAILA AZIZA

Matricule : 201400007594

M2 SII Groupe:3

Professeur

Mme. Hedjazi Badiaa

January 4, 2019

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	7
2.3.1	Explication	7
2.4	Étape 3	10
2.4.1	Génération automatique aléatoire des Graphes	10
2.5	Résultats	11
2.6	Observation pour chaque type	11
2.6.1	Observation pour les Polytrees	11
2.6.2	Observation pour les Multiconnected	11
2.6.3	Observation pour les Simply connected	11
3	Conclusion et comparaison	12

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
3.1	Expérimentation 1	14
3.2	Expérimentation 2	14
3.3	Expérimentation 3	15
3.4	Expérimentation 4	15

Chapter 1

Introduction

Nous avons précédemment (dans le module RC1) manipulé les notions d'inférence logique avec Max-Sat, mais plusieurs autres méthodes existent dont les méthodes graphiques.

Le présent TP est orienté théorie des possibilités, celui-ci pouvant être modélisé aussi bien graphiquement que par une représentation MAX Weighted SAT, nous avons pour but de tester les deux méthodes sur différents problèmes c'est à dire représentés sous forme de polytree, graph multiconnected et graph simply connected.

Pour cela deux outils largement utilisés dans ce domaine ont été mis à notre disposition, il s'agit de la **PNT** de **Matlab** et l'outil **cygwin** sous Windows.

Ainsi nous allons grâce à ces outils comparer les temps d'inférence logique avec les temps de propagation de la méthode graphique afin de comprendre les avantages et inconvénients de chacun, et de savoir sur quelle base choisir la méthode d'inférence du degré de possibilité la plus adéquate.

Chapter 2

Réalisation

2.1 L'outil cygwin

Cygwin est une collection de logiciels libres à l'origine développés par Cygnus Solutions permettant à différentes versions de Windows de Microsoft d'émuler un système Unix. Il vise principalement l'adaptation à Windows de logiciels qui fonctionnent sur des systèmes POSIX (tels que les systèmes GNU/Linux, BSD, et Unix). Cygwin simule un environnement Unix sous Windows, rendant possible l'exécution de ces logiciels après une simple compilation.[1]

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) créé. 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 nœuds du graphe à construire
2. Création de liens de façon aléatoire entre les nœuds.

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)

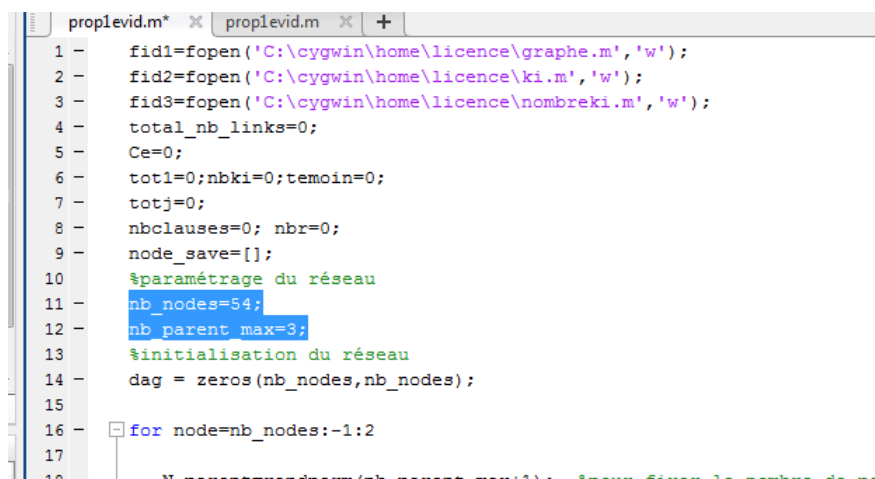
Concernant **Prodevid2** c'est le même fonctionnement à part qu'on a droit à deux évidences donc deux informations vont influencer notre réseau , en théorie on peut penser que la propagation des deux évidences prendra plus de temps que celle d'une seule , on testera ce cas dans ce qui suit .

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=randnrm(nb_parent_max+1); %pour fixer le nombre de na

```

Figure 2.1: Le changement des nb noeuds ainsi que nb parents max dans le code proplevid.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 $[1, \text{nbparentsmax}+1]$ donc concrètement le nombre de parents max est de 4 et non de 3 voir code:prod1evid.m.

Affichage

le résultat était le suivant :

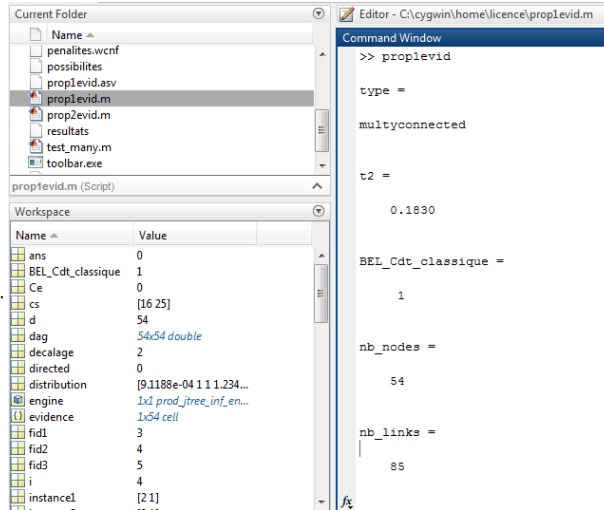


Figure 2.2: Une seule evidence

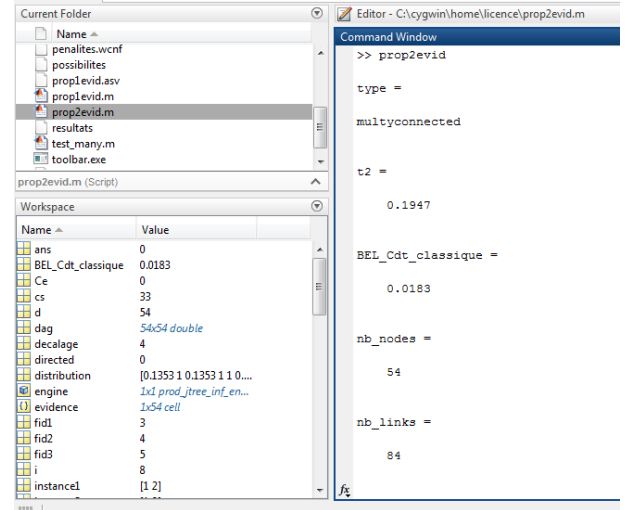


Figure 2.3: Deux evidences

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

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 conditionnelles est plus complexe avec deux informations (deux conditions) qui arrivent qu'avec une seule (une seule condition) .

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

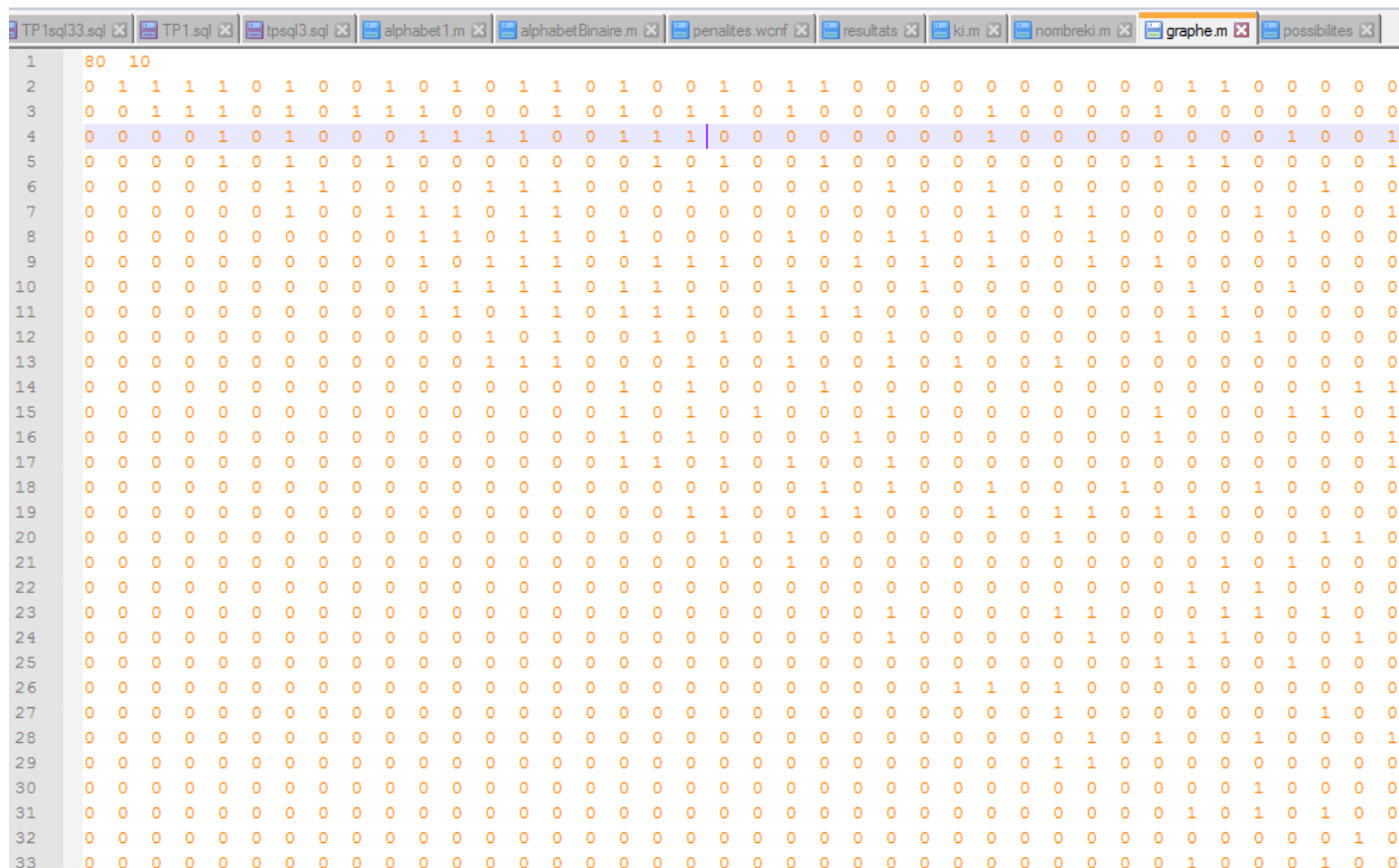


Figure 2.4: exemple d'un graphe en sortie à l'exécution de prodlevid.m avec noeuds=80 et parents =10

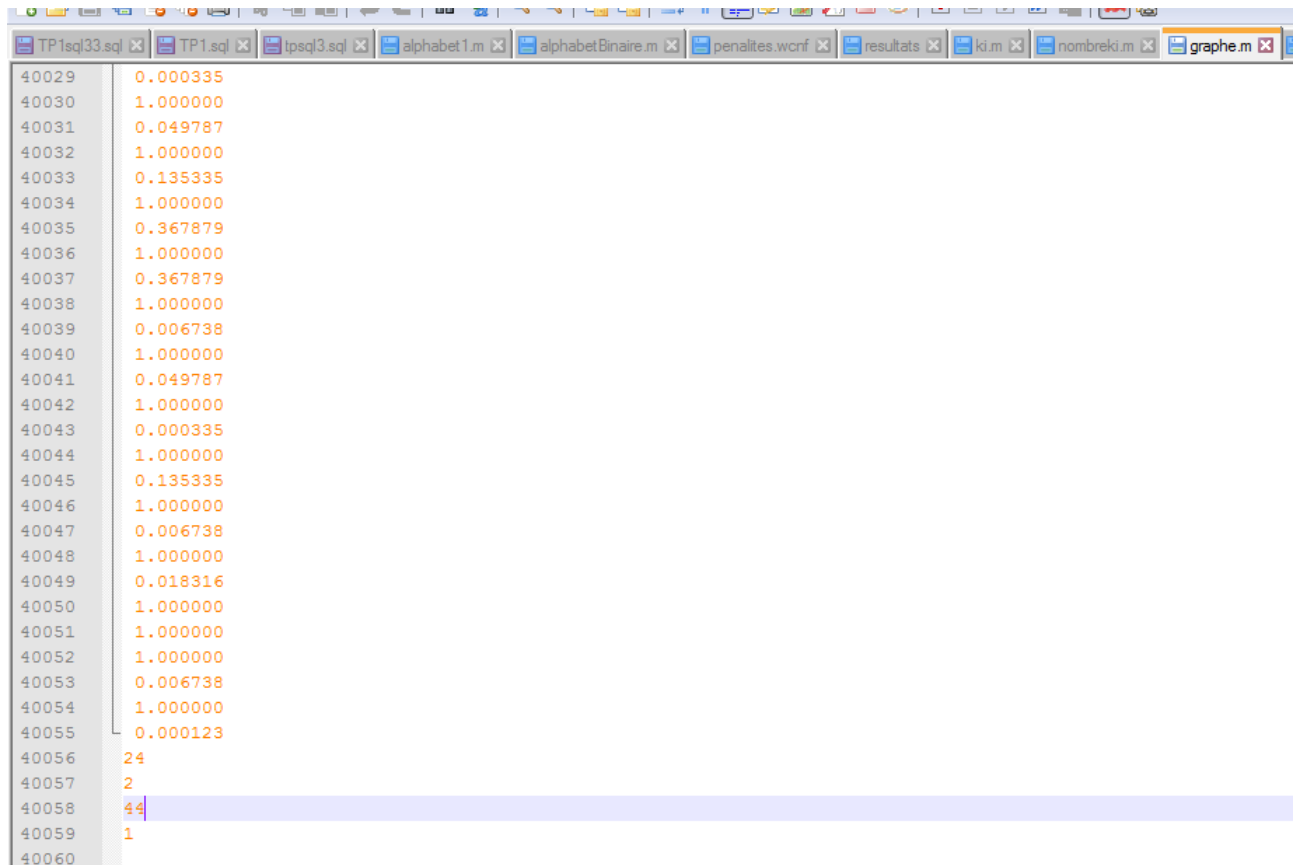


Figure 2.5: exemple d'un graphe en sortie à l'exécution de prod1evid.m avec noeuds=80 et parents =10

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 resultats

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 a un réseau multi-connected on a remarqué que la propagation 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 Étape 3

2.4.1 Génération automatique aléatoire des Graphes

Génération des Polytree

Un **polytree** est un graphe ayant les particularités suivantes: c'est un graphe orienté (les arcs reliant les noeuds sont orienté parent \rightarrow fils),connexe (pas de noeuds ou sous arbre libre), acyclique (ne contient pas de cycle/ boucle), et enfin simplement connecté.

Ainsi afin d'obtenir des polytree, nous avons limité le nombre maximum de parent à 1 seul parent.

Par contre, pour ce qui est du nombre de noeuds nous avons le choix, mais nous nous sommes limité aux testes montrés dans le tableau récapitulatif plus bas.

Génération des Multiconnected

On parle graphes **Multiconnected** lorsque nous avons un graphe orienté (les arcs reliant les noeuds sont orienté parent \rightarrow fils),connexe (pas de noeuds ou sous arbre libre), acyclique (ne contient pas de cycle/ boucle), et enfin à connections multiple de ses nœuds.

Pour obtenir cette catégorie de graphe nous pris des nombres de parents supérieur à 1 , et même de l'ordre de 7 à 10 parents par noeuds, toujours en faisant varier le nombre de noeuds, afin de diversifier les résultats et explorer plus de combinaison.

Génération des Simply connected

la notion de simple connexité raffine celle de connexité : là où un espace connexe est simplement "d'un seul tenant", un espace simplement connexe est de plus sans "trou" ni "poignée" mais nous parlant toujours de graph sans boucles.[2]

De manière analogue que pour les deux type de graphes précédents nous avons généré ce type de graphe selon le nombre de noeuds et de parent et obtenu les résultats regroupés dans la section "résultats".

2.5 Résultats

Suite à maintes expérimentations et testes avec les différentes configurations précédentes, nous avons résumé quelques résultats dans le tableau suivant:

NbrNœuds/NbrParents	Temps Propagation	Temps Inférence	Degrés Possibilité
25 Nœuds / 1 Parents	0.156598 sec	0.171601 sec	0.049787
50 Nœuds / 1 Parents	0.282039 sec	0.218400 sec	1
15 Nœuds / 3 Parents	0.052538 sec	0.156000 sec	0.049787
25 Nœuds / 4 Parents	0.081438 sec	0.156000 sec	1
25 Nœuds / 7 Parents	0.078056 sec	0.249600 sec	1
25 Nœuds / 10 Parents	0.55983 sec	0.315625 sec	0.018316
40 Nœuds / 10 Parents	14.1780 sec	0.50927 sec	0.36788
30 Nœuds / 7 Parents	0.094228 sec	0.218401 sec	1
30 Nœuds / 1 Parent	0.196302 sec	0.187200 sec	0.018316
80 Nœuds / 10 Parents	pc stopped	pc stopped	pc stopped

Quelques résultats sont mis dans l'annexe sous forme de capture d'écran.

2.6 Observation pour chaque type

2.6.1 Observation pour les Polytree

Comme le montre les résultats précédemment obtenus, les polytree sont déjà très rapide en temps de construction, puis nous remarquons que ce soit par la méthode graphique (**Propagation**) ou la méthode logique (**Inférence**) les degrés de possibilité sont donné très rapidement, même que pour certains cas la méthode graphique est très légèrement plus rapide

2.6.2 Observation pour les Multiconnected

D'après les résultats produits suite aux nombreuses expérimentations que nous avons effectué, nous sommes en mesure de souligner la rapidité de la méthode d'inférence logique (basé maw weighted sat) comparé à la méthode graphique (propagation), tel que pour de grandes instances / graphe multiconnectés (beaucoup de noeuds et beaucoup d'arcs) la méthode d'inférence logique reste raisonnablement rapide, légèrement plus lente que pour les polytree, contrairement à la méthode graphique de propagation qui voit le temps de propagation croître de l'ordre de 10 aine, jusqu'à ne plus être exécutable sur nos machine comme pour (80 noeuds et 10 parents).

2.6.3 Observation pour les Simply connected

En vu des résultats des expérimentations sur ce type de graphe nous avons constaté que le temps de propagation via la méthode graphique est plus rapide que le temps d'inférence de la méthode logique, bien que les deux reste inférieur à 1 second.

Chapter 3

Conclusion et comparaison

Suite aux expérimentation effectuées avec les deux techniques misent à notre disposition lors de ce Tp, nous pouvons constater que la méthode d'inférence logique est bien plus rapide et efficace que la méthode graphique de propagation.

Car bien que les méthodes graphiques soient plus parlantes et lisible pour l'humain, leurs complexité n'en est pas pour autant bonne, car en plus du temps de propagation très lent pour les grandes instances de problème (multiconnected), la méthode de propagation nécessite aussi la construction de l'arbre de jonction qui prends elle aussi un temps non négligeable.

Ainsi à la limite pour les polyarbe nous pouvant utiliser les deux méthodes car bien que la méthode graphique soit légèrement plus rapide que celle d'inférence logique, les deux méthodes donne le résultats très rapidement, d'ailleurs la méthode graphique ne nécessite pas dans ce cas là la construction de l'arbre de jonction.

Bibliographie

[1] : <https://fr.wikipedia.org/wiki/Cygwin>

[2] : https://fr.wikipedia.org/wiki/Connexit%C3%A9_simple

ANNEXE

```

/home/licence
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 0
avec un degré de possibilité correspondant est égale à :1.000000
la durée de l'inférence est: 171601 millisecondes
master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference
187200
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 30
nombre de parentsmax: 0
l'evidence: 28
variable d'intérêt: 23
possibilité conditionnelle de l'intérêt | evidences 1.000000
temps de propagation 0.196302 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 58
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 0
avec un degré de possibilité correspondant est égale à :1.000000
la durée de l'inférence est: 187200 millisecondes
master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference
218401
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 30
nombre de parentsmax: 6
l'evidence: 17
variable d'intérêt: 30
possibilité conditionnelle de l'intérêt | evidences 0.018316
temps de propagation 0.094228 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 509
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 4
avec un degré de possibilité correspondant est égale à :0.018316
la durée de l'inférence est: 218401 millisecondes
master@master-PC /home/licence

```

Figure 3.1: Expérimentation 1

```

/home/licence
la durée de l'inférence est: 1107602 millisecondes
master@master-PC /home/licence
$ ./passage.exe

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

master@master-PC /home/licence
$ ./inference
218400
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 50
nombre de parentsmax: 0
l'evidence: 35
variable d'intérêt: 22
possibilité conditionnelle de l'intérêt | evidences 1.000000
temps de propagation 0.282039 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 97
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 0
avec un degré de possibilité correspondant est égale à :1.000000
la durée de l'inférence est: 218400 millisecondes
master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference
171601
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 30
nombre de parentsmax: 3
l'evidence: -3
variable d'intérêt: 28
possibilité conditionnelle de l'intérêt | evidences 1.000000
temps de propagation 0.000000

*****Résultats de l'inférence logique*****
le nombre de clauses d
la variable d'intérêt
le coût de pénalité es
avec un degré de poss
la durée de l'inférenc
master@master-PC /home
$

```

Figure 3.2: Expérimentation 2

```

/home/licence
le nombre de clauses dans les bases est de: 509
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 4
avec un degré de possibilité correspondant est égale à :0.018316
la durée de l'inférence est: 218401 millisecondes
master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference
156000
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 15
nombre de parentsmax: 2
l'evidence: -6
variable d'intérêt: 7
possibilité conditionnelle de l'intérêt | evidences 0.049787
temps de propagation 0.052538 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 39
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 3
avec un degré de possibilité correspondant est égale à :0.049787
la durée de l'inférence est: 156000 millisecondes
master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference
171601
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 25
nombre de parentsmax: 0
l'evidence: -15
variable d'intérêt: -15
possibilité conditionnelle de l'intérêt | evidences 0.049787
temps de propagation 0.156598 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 51
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 3
avec un degré de possibilité correspondant est égale à :0.049787
la durée de l'inférence est: 171601 millisecondes
master@master-PC /home/licence

```

Figure 3.3: Expérimentation 3

```

/home/licence
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 25
nombre de parentsmax: 3
l'evidence: -10
variable d'intérêt: 4
possibilité conditionnelle de l'intérêt | evidences 1.000000
temps de propagation 0.081438 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 97
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 0
avec un degré de possibilité correspondant est égale à :1.000000
la durée de l'inférence est: 156000 millisecondes
master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference
249600
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 25
nombre de parentsmax: 6
l'evidence: -21
variable d'intérêt: -1
possibilité conditionnelle de l'intérêt | evidences 1.000000
temps de propagation 0.078056 secondes

*****Résultats de l'inférence logique*****
le nombre de clauses dans les bases est de: 611
la variable d'intérêt est inférée à partir de la base de pénalités
le coût de pénalité est de 0
avec un degré de possibilité correspondant est égale à :1.000000
la durée de l'inférence est: 249600 millisecondes
master@master-PC /home/licence
$ ./passage.exe

master@master-PC /home/licence
$ ./inference
218401
master@master-PC /home/licence
$ cat resultats
*****Résultats de la propagation graphique*****
nombre de variables: 25
nombre de parentsmax: 9
l'evidence: -15

```

Figure 3.4: Expérimentation 4