# Outils

Pour la réalisation de ce TP nous avons tenu à combiner la force de calcule qu’offre MATLAB ainsi que la stabilité et efficacité multiplateforme de JAVA pour l’automatisation des taches de tests et la création d’une interface graphique conviviale, en combinaison avec Cygwin l’émulateur Unix sous Windows pour utiliser les outils fournis pour la réalisation de ce tp.

# Procédure

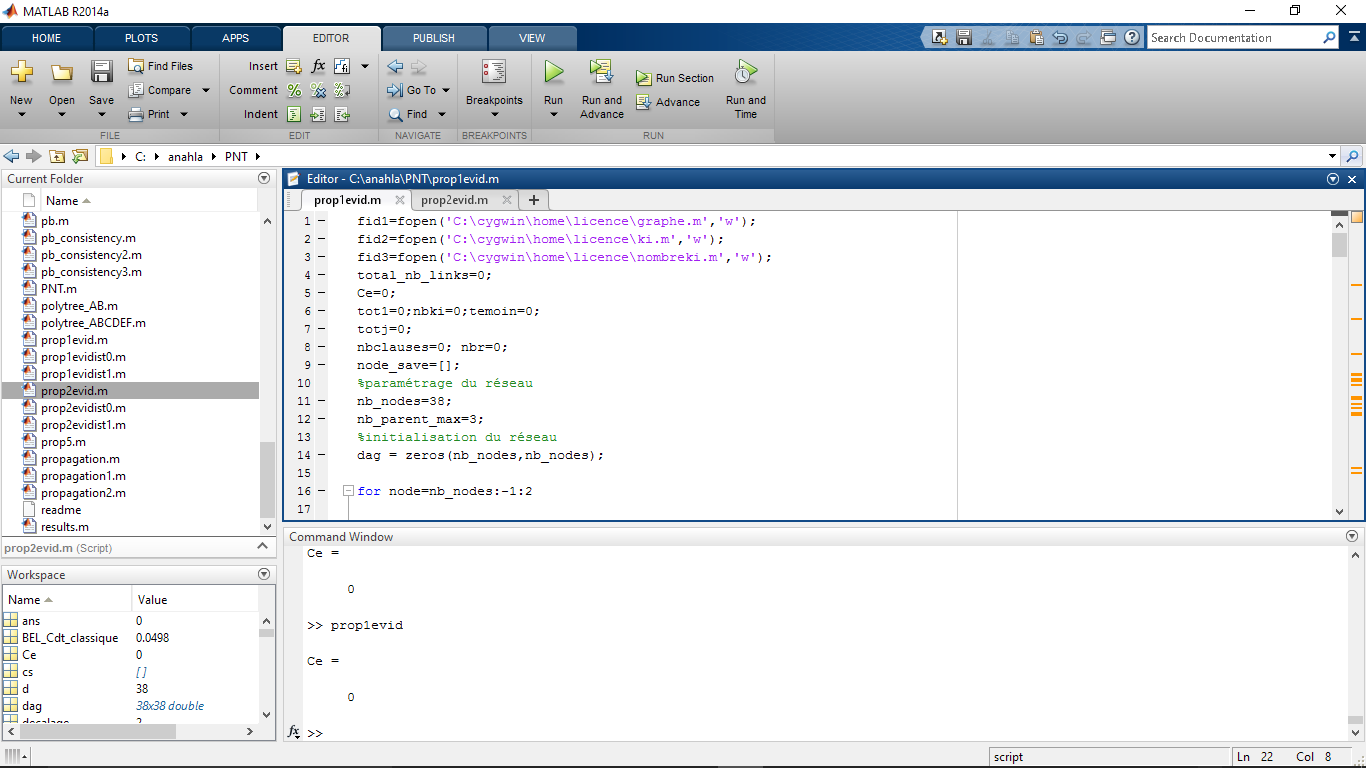
Nous avons commencé par réaliser toutes les étapes d’installation de MATLAB et de cygwin, puis nous avons mis en places les fichiers nécessaires à la mise en marche de l’environnement (pnt). Après que l’environnement soit devenu exploitable, nous avons enchainé une série de tests en lançant d’abord **prop1evid.m** sous MATLAB pour une seule évidence en changeant minutieusement le nombre de nœuds et le nombre de parents maximal en essayant de maximiser les deux variables, ensuite nous somme passé à **prop2evid.m** pour deux évidences (voir détails des tests section Tests). Enfin une fois les fichiers graph.m ki.m et nombreki.m générés nous passion sous Cygwin et lançons les programmes :

./passage.exe : qui créer à partir du graphe généré précédemment une base de connaissance possibiliste quantitative correspondante à ce dernier.

Et ./inférence.exe : celui-ci lance le processus d’inférence càd calcule le degré de possibilité ainsi que le temps de l’inférence.

# Tests

Une fois l’installation prête nous avons réalisé une série de tests sur matlab.



**Figure 1 :** Lancement de prop1evid.m

Nous avons commencé par tester prop1evid.m càd une seule évidence (les fichiers résultants .m sont disponibles) en faisant varier le nombre de nœuds et de parents dans la mesure du possible en essayant de maximiser les deux paramètres.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Noeuds** | 20 | 20 | 30 | 35 | 38 | 50 | 52 | 70 | 70 | 70 | 100 | 133 | 120 | 140 |
| **Parents** | 4 | 5 | 5 | 5 | 3 | 5 | 3 | 5 | 6 | 7 | 3 | 3 | 3 | 3 |
| **Temps (ms)** | 9070 | 8583 | 10302 | 10763 | 9681 | 8313 | 10426 | 10063 | 22189 | 25268 | 9005 | 12784 | 9536 | 11717 |

**Tableau 1 :** Différents tests sur une seule évidence.

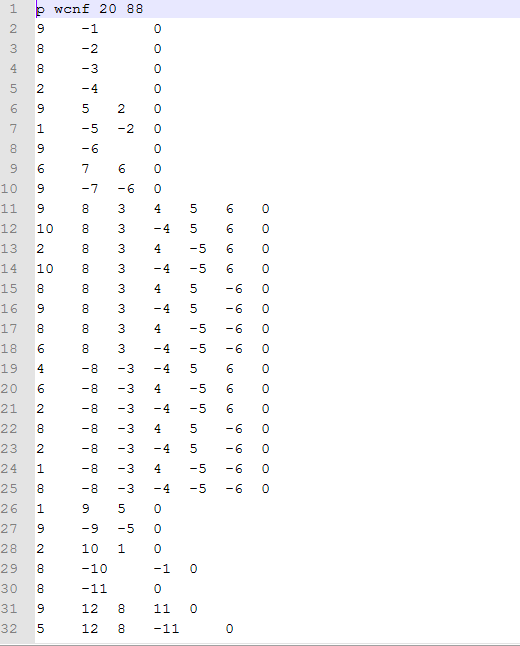
De la même manière nous avons effectué une série de tests sur prop2evid.m càd pour 2 évidences

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Noeuds** | 10 | 20 | 35 | 38 | 50 | 70 | 70 | 80 | 90 | 100 | 100 | 133 | 120 | 140 |
| **Parents** | 3 | 4 | 5 | 3 | 5 | 6 | 7 | 5 | 4 | 3 | 4 | 3 | 3 | 3 |
| **Temps (ms)** | 9194 | 9900 | 9845 | 8530 | bug | 24053 | bug | 45913 | 10732 | bug | 11666 | 12329 | bug | 13096 |

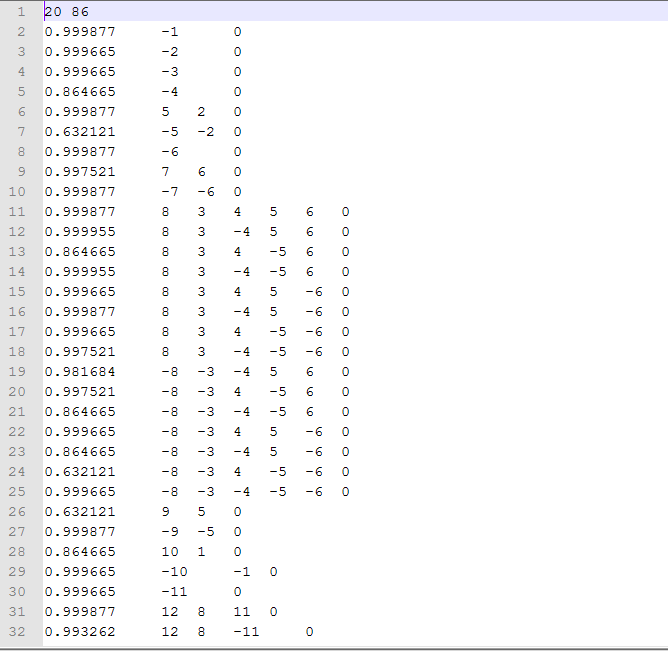
**Tableau 2 :** Différents tests sur deux évidences.

Une fois que a nos 3 fichiers graphe ki et nombreki nous passion par Cygwin et lançon la commande « ./passage.exe » une fois lancé nous avons obtenus 4 nouveaux fichiers : resultats, penalites.wcnf, possiblites et data. Le penalite.wcnf est un simple fichier wcnf (weighted cnf) càd il contient des variables des clauses ainsi que des poids (cout des pénalités) il représente la base de connaissance (de pénalités), ce fichier sera utilisé par un solveur wmaxsat pour essayer de trouver une solution au sat qu’il représente. Ensuite le fichier possibilite est une copie du fichier penalite.wcnf dans lequel à chaque clause est associé un degré de possibilité compris entre 0 et 1. Enfin le fichier resultats résume le nombre de variables et de parents, l’évidence le temps de la propagation.

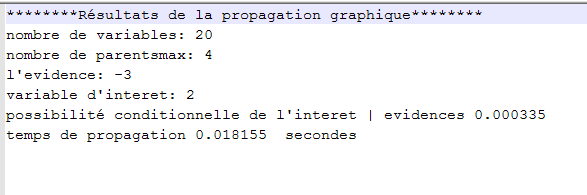
Enfin après le lancement de « ./inférence.exe » le solveur wmaxsat est lancé sur la base des pénalités penalite.wcnf la variable d’intérêt est inférée à partir de celle-ci et un fichier wmaxsat est créée qui contiendra l’optimum trouvé dans quel nœuds il a été trouvé et en combien de temps, et enfin le programme rajoute aussi le nombre de clause de la base la variable d’intérêt le cout de pénalité et son degré de possibilité et la durée de l’inférence au fichier resultats.



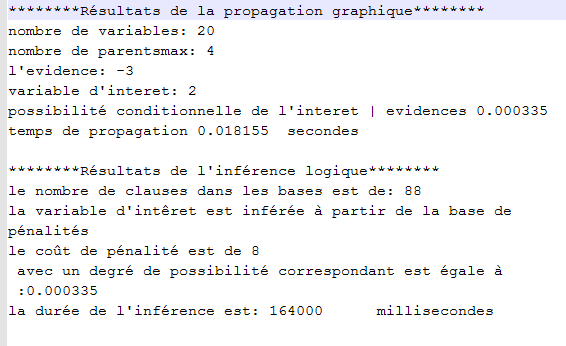
**Figure 2 :** exemple base de pénalités générée.



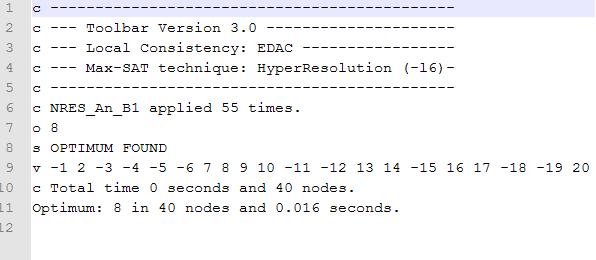
**Figure 3 :** exemple du fichier possibilite.



**Figure 4 :** exemple du fichier resultat après le passage de la 1ère commande.



**Figure 5 :** Exemple du fichier resultats après l’execution de la 2ème commande.



**Figure 6 :** exemple du fichier wmaxsat.

# Erreurs rencontrées

Dans les deux cas à partir d’une certaine valeur le processus bug et nous obtenons des erreurs liées à la mémoire essentiellement.

Dans le cas de « **prop1evid** » nous n’avons pas pu tester les combinaisons suivantes :

Nœuds 90 , Parents 7

Nœuds 100 , Parents 5

Nœuds 70 , Parents 8

Nœuds 150 , Parents 3

dans le cas de «**prop2evid** » nous n’avons pas pu tester les combinaisons suivantes :

Nœuds 150, Parents 3

Nœuds 100, Parents 5

Nœuds 120, Parents 4

Nœuds 88, Parents 6

L’erreur using ones:

Error using ones

Out of memory. Type HELP MEMORY for your options.

Error in myones (line 15)

T = ones(sizes(:)');

Error in dpot (line 12)

pot.T = myones(sizes);

Error in mk\_initial\_pot\_d (line 11)

pot = dpot(dom, ns(dom));

Error in prod\_jtree\_inf\_engine/enter\_soft\_evidence (line 21)

clpot{i} = mk\_initial\_pot\_d(engine.clusters{i}, ns, onodes);

Error in prod\_jtree\_inf\_engine/global\_propagation (line 19)

[clpot] = enter\_soft\_evidence(engine, CPDpot, onodes);

Error in prop1evid (line 178)

[engine] = global\_propagation(engine, evidence);

L’erreur using repmat:

Error using repmat

Out of memory. Type HELP MEMORY for your options.

Error in myrepmat (line 8)

T = repmat(T, sizes(:)');

Error in extend\_domain\_table (line 23)

B = myrepmat(B, sz(:)');

Error in dpot/multiply\_by\_pot (line 7)

Ts = extend\_domain\_table(Tsmall.T, Tsmall.domain, Tsmall.sizes, Tbig.domain, Tbig.sizes);

Error in prod\_jtree\_inf\_engine/enter\_soft\_evidence (line 28)

clpot{c} = multiply\_by\_pot(clpot{c}, potential{i});

Error in prod\_jtree\_inf\_engine/global\_propagation (line 19)

[clpot] = enter\_soft\_evidence(engine, CPDpot, onodes);

Error in prop1evid (line 178)

[engine] = global\_propagation(engine, evidence);

L’erreur using .\* :

Error using .\*

Out of memory. Type HELP MEMORY for your options.

Error in dpot/multiply\_by\_pot (line 8)

Tbig.T = Tbig.T .\* Ts;

Error in prod\_jtree\_inf\_engine/enter\_soft\_evidence (line 28)

clpot{c} = multiply\_by\_pot(clpot{c}, potential{i});

Error in prod\_jtree\_inf\_engine/global\_propagation (line 19)

[clpot] = enter\_soft\_evidence(engine, CPDpot, onodes);

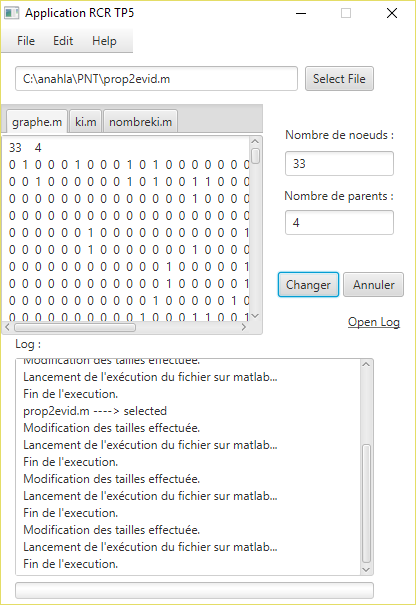
Error in prop2evid (line 188)

[engine] = global\_propagation(engine, evidence);

# Application

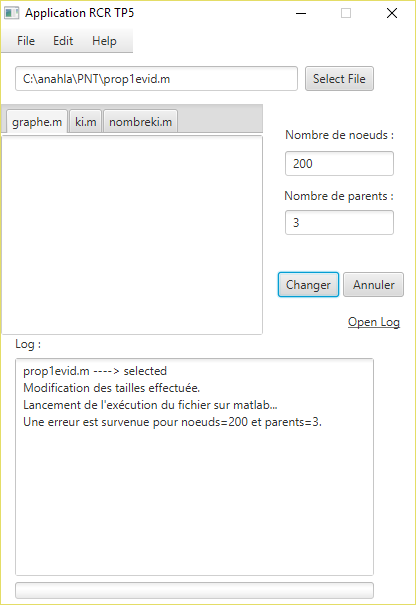
Pour faciliter la tâche de tests nous avons créé une interface permettant de faire tout ce qui était demandé de faire de façon beaucoup plus facile sans avoir à passer par plusieurs programmes intermédiaires

L’application se présente comme ceci une fois qu’un des deux fichiers **prop1evid.m** ou **prop2evid.m** est sélectionné, un panneau s’affiche sur la droite de l’application ce dernier permet de modifier facilement le nombre de nœuds et de parents et de le lancer sans avoir à lancer Matlab, 3 onglet représentants nos 3 fichiers sont alors présenté sur l’interface de l’application à chaque instant tous les détails de toutes action faite est reporté sur le log (exemple durée d’exécution) **. (Figure 7)**



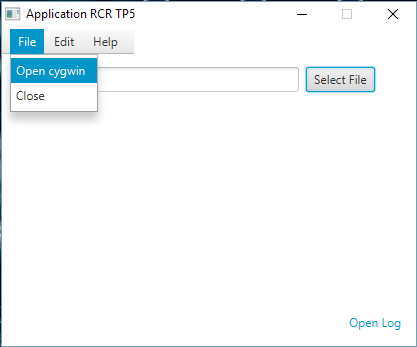
**Figure 7 :** tests de 2 évidences avec n=33 et p=4 à partir l’application

Lorsque l’une des erreurs vu précédemment se produit, elle est affiché dans le log ainsi que la durée d’exécution avec l’arrivé du bug et les 3 onglets restent vident. **(Figure 8)**



**Figure 8** : erreur répertorié dans le log.

Enfin il est aussi possible de lancer cygwin à partir de l’application pour passer directement à l’étape 2. **(Figure 9)** et **(Figure 10)**



**Figure 9 :** Sous menu pour l’ouverture de Cygwin à partir de l’application