SORBONNE UNIVERSITÉ

Rapport de devoir de programmation

UE d'ALGAV

Delannay Léa(3672098)

1.Présentation

Choix du langage:

Je me suis d'abord tournée vers le langage ocaml car j'avais eu à réaliser un autre devoir de programmation dans celui-ci, j'étais donc habituée. J'ai fait les quatres questions de la partie échauffement sans trop de difficultés, mais par la suite je me suis rendu compte que je m'en sortirai beaucoup mieux en python pour réaliser les arbres et leur compression. J'ai donc tout refait en python.

2. Arbre de décision et compression

Structure de donnée:

```
class ArbreBinDec():

   def __init__(self, val, fg, fd,luka,pere):

       self.val = val

       self.fg = fg

       self.fd = fd

       self.luka = luka
```

Un arbre binaire de décision se crée grâce à une valeur (étiquette du nœud), un fils gauche et un fils droit (qui sont eux même des arbres) et un mot de Lukasiewicz.

Fonction dot :

J'ai rencontré un souci au moment de tester ma fonction sur un arbre préalablement compressé. Les graphes que j'obtenais n'étaient pas du tout corrects, alors qu'elle renvoyait les bons graphes pour des arbres non compressés.

J'ai donc créé une deuxième fonction (que j'ai appelée dot2) qui elle fonctionne seulement pour les DAG et ROBDD. Je n'ai pas réussi à trouver comment les fusionner pour qu'elles fonctionnent pour tous les arbres.

3. Arbre de décision et ROBDD

3.11

On sait donc que:

Soit h, la hauteur de l'arbre. Il y a 2^h feuilles et 2^h-1 nœuds internes.

Calculons le nombre de parenthèses du mot associé à l'arbre:

Nous savons que tous les nœuds sont entourés de parenthèses sauf la racine.

Donc il y a (nombre de feuilles + nombre de nœuds internes - 1) nœuds parenthésés.

C'est à dire :

2^h+2^h-1 -1 $=2^h+2^h-2$ $= 2.2^h-2$ noeuds parenthèses

Il y a une parenthèse ouvrante avant la valeur du nœud et une fermante après (soit 2 caractères).

Il y a donc (2.2^h-2)x2 = $\mathbf{4.2^h-4}$ parenthèses.

<u>Calculons la longueur des valeurs des feuilles :</u>

Au pire cas toutes les feuilles sont égales à False, leur mot prend donc 5 caractères.

Comme il y a 2^h feuilles, les feuilles prennent $\mathbf{5.2^h}$ caractères.

<u>Calculons la longueur des valeurs des noeuds internes:</u>

Les valeurs prennent la forme de "xv" avec $1 \leq v \leq 9$, donc elles prennent toujours 2 caractères.

Comme il y a 2^h-1 nœuds internes, les noeuds internes prennent (2^h-1)x2=$\mathbf{2.2^h-2}$ caractères

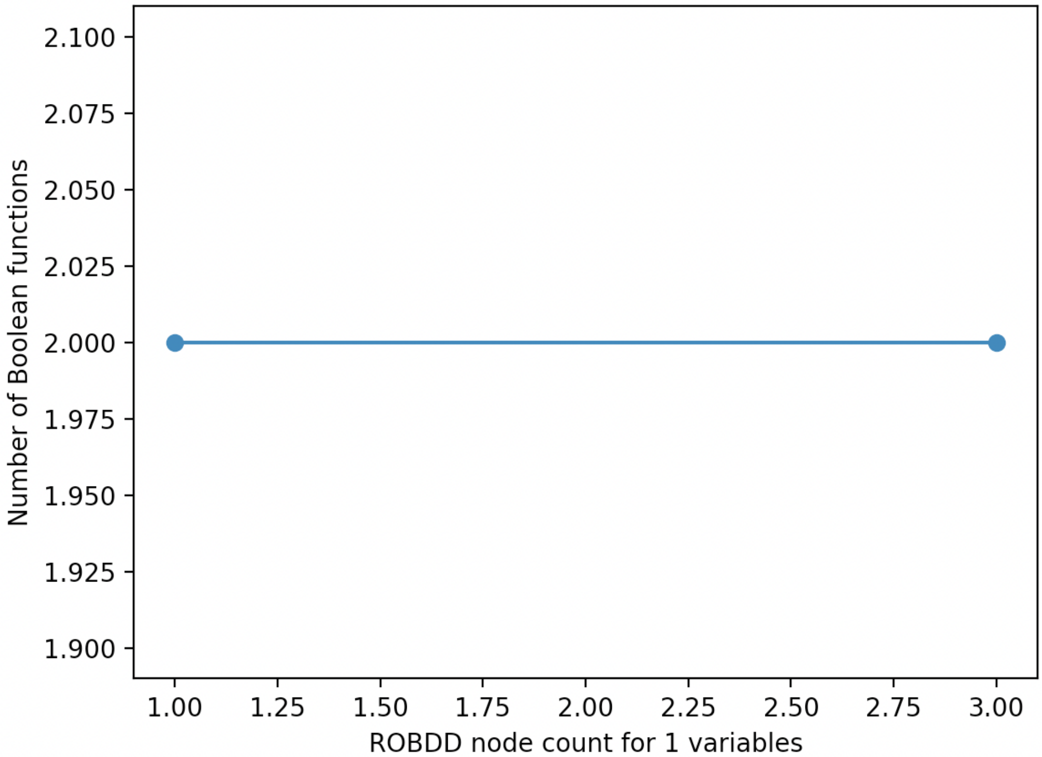
Nous pouvons donc dire que la longueur du mot de Lukasiewicz est égale à:

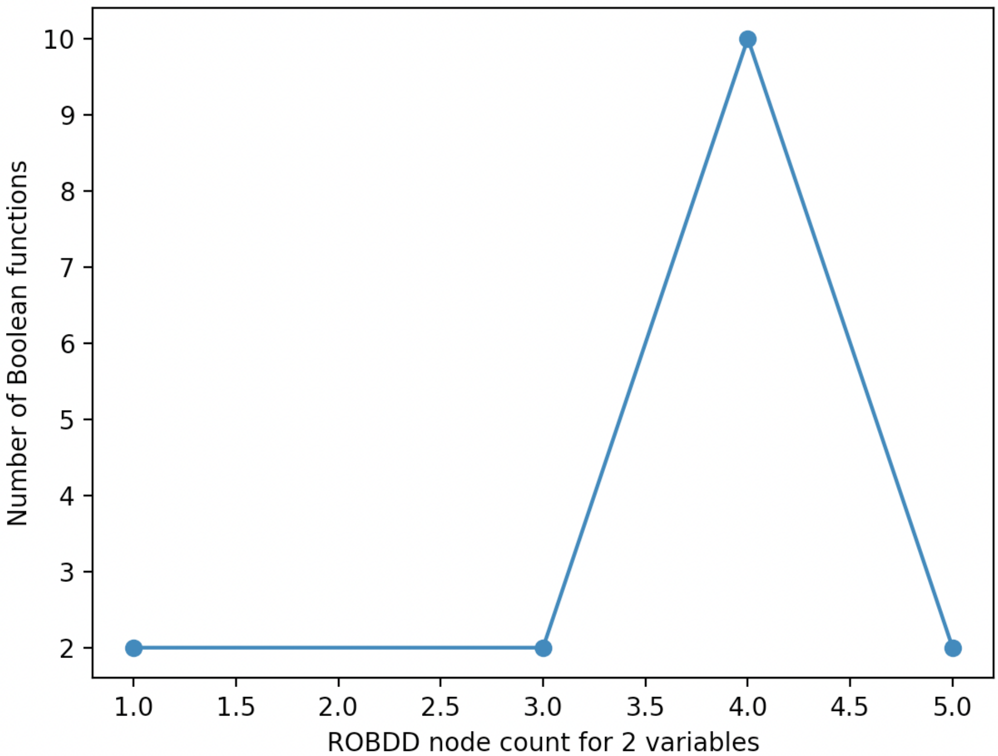
nombre de parenthèses + longueurs des valeurs des feuilles + longueurs des valeurs des noeuds internes = $4.2^h-4 + 5.2^h+2.2^h-2$ = $\mathbf{11.2^h-6}$

3.12

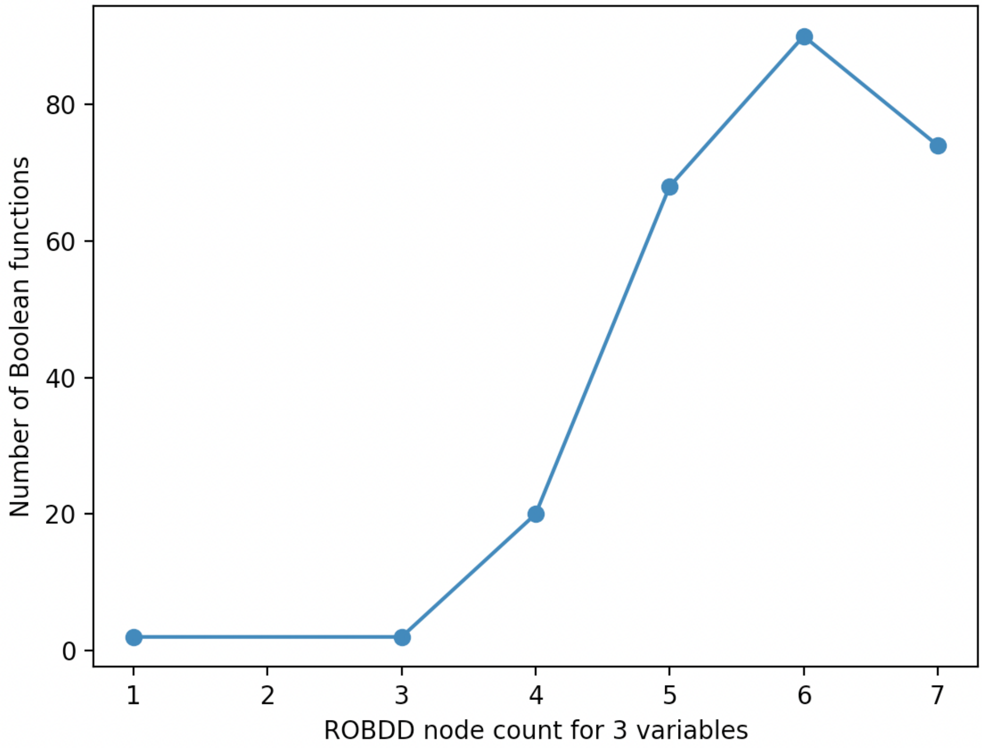
3.13

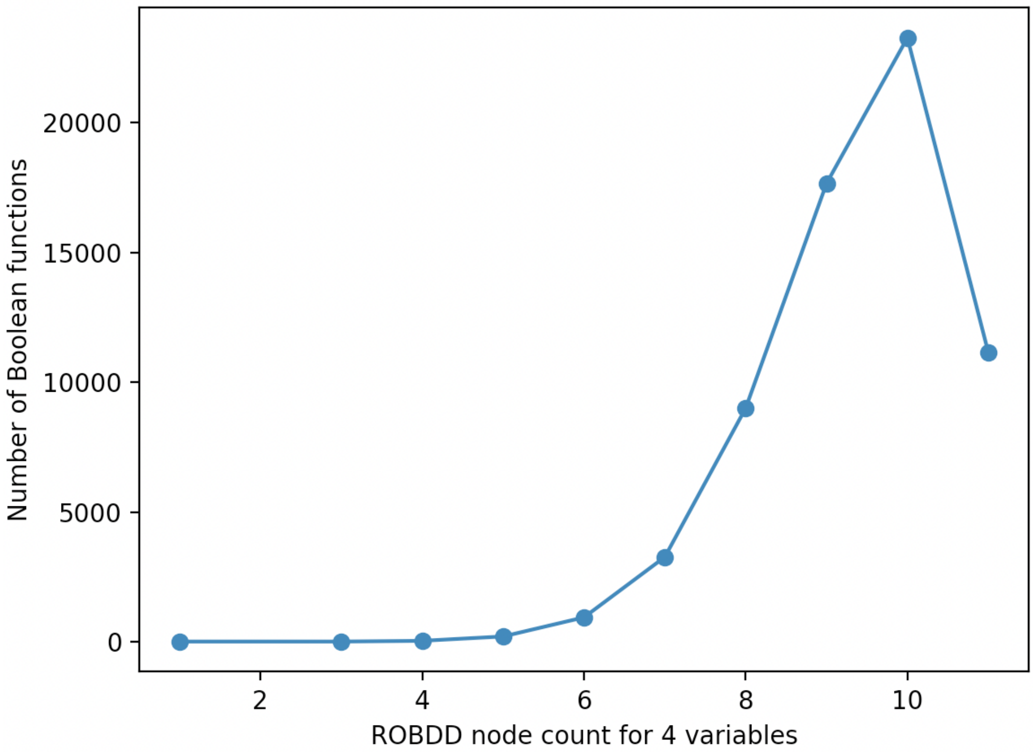
4. Etudes expérimentales



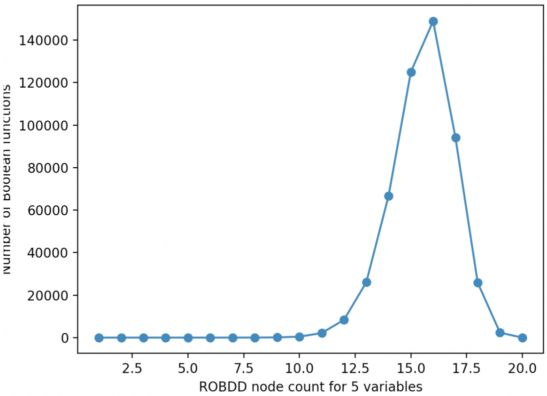


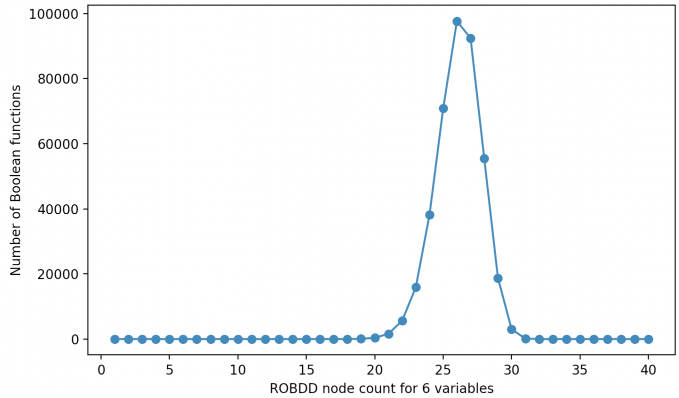
Pour 2 variables, normalement on devrait avoir 4 fonctions booléennes avec 2 nœuds et 8 avec 4 nœuds. J’ai un souci dans mon algorithme de compression_bdd qui ne gère pas le cas où la racine a les mêmes fils gauche et droit. Elle devrait donc disparaître pour ne laisser qu’un seul des fils, mais ici elle reste en plus de garder un des fils.

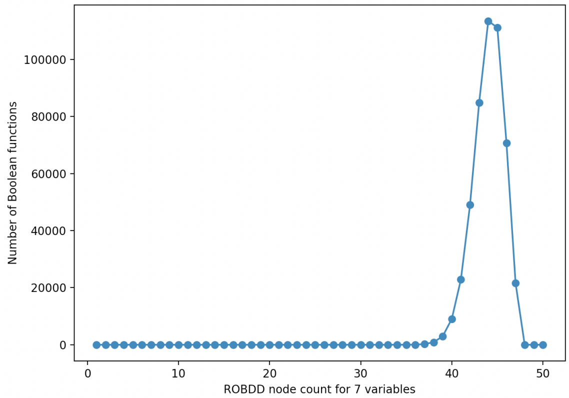


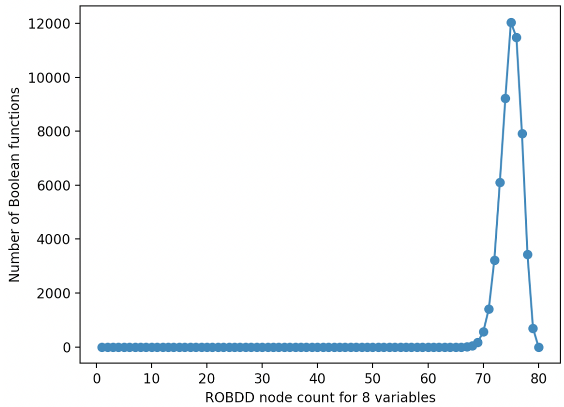


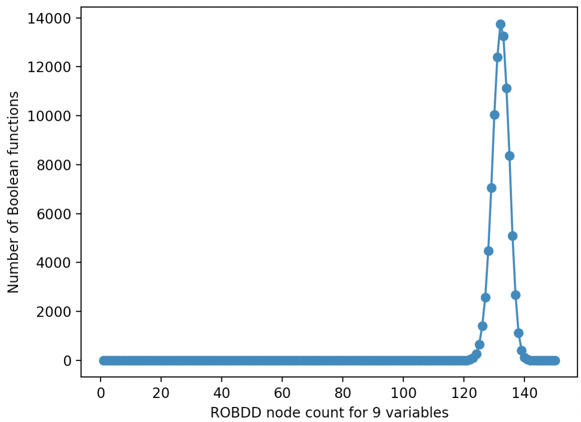
Je n’ai pas réussi à trouver la distribution exacte pour 5 variables, j’ai donc dû échantillonner comme pour la figure 10.

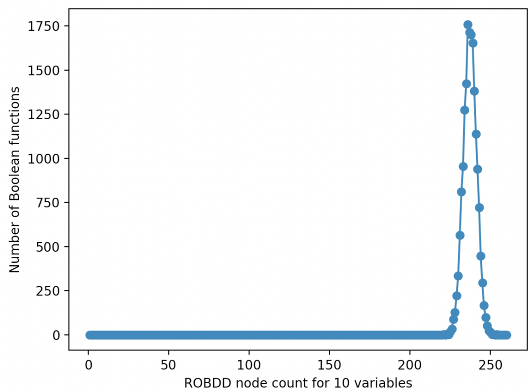












Mes courbes ne correspondent pas totalement à celles présentées en figure 10. Je pense que cela est dû encore à ma fonction compression_bdd qui parfois ne doit pas regrouper assez de nœuds.

J’ai décidé d’utiliser le même nombre d’échantillons que ceux proposés en figure 11. Voici donc le tableau avec mes résulats:

No. Variables (n)	No. Samples	No. Unique sizes	Compute Time (sec)	Seconds ROBDD
5	500,003	14	239.53368902	0,00047906
6	400,003	16	366.27800417	0,00091569
7	486,892	15	874.25572681	0,00179558
8	56343	16	207.95371508	0,00369085
9	94999	25	717.37357687	0,00755138
10	17975	31	288.21703601	0,01603433

Les nombres de nœuds uniques varient aussi. Par contre les temps d'exécutions sont nettement inférieurs.