MATTON Nicolas
BELANGER Alexis

Rapport Projet CPA

Pour l'ensemble des tme, un makefile est disponible dans le dossier /src.
Les sources sont disponible à cette adresse : https://github.com/Arthemil/CPA (dès le 07/04/19 23:59)

TME N°3

Exercice 2

Le fichier *"exercice2.c"* calcul le nombre d'arêtes et le nombre de noeuds et le temps d'exécution pour chaque fichiers.

	Nombre d'arêtes	Nombre de noeuds	Temps cpu
Email	**16 064**	**986**	**<1s**
Amazon	**925 872**	**334 863**	**<1s**
LJ	**34 681 189**	**3 997 962**	**5s**
Orkut	**117 185 083**	**3 072 441**	**19s**
Friendster	**1 806 067 135**	**65 608 366**	**7m20**

Exercice 3

la commande : awk '!seen[$1,$2]++ && !seen[$2,$1]' "Fichier" > "Fichier"_clean permet de supprimer les doublons et les arêtes de type "u u".

Exercice 4

Le fichier "*exercice4.c*" calcul pour les 4 premiers fichiers les degrés respectifs de chaque noeuds, et les enregistrent dans un fichier dans le répertoire "ressource/Degrees/".

Pour Friendster nous ne disposons pas assez de quota sur nos sessions pour stocker un tel fichier.

Exercice 5

Le fichier "*exercice5.c*" calcul pour les 4 premiers fichiers la somme Q. Pour cela nous stockons pour chaque fichier les degrés de chaque sommet dans un tableau indicé par le numéro des noeuds.
Le temps pour friendster est dû à l'obligation de recalculer les degrés pour chaque noeuds, ce qui **double** le temps.

	Email	**Amazon**	**Lj**	**Orkut**	**Friendster**
Quantité Q	**88 109 182**	**103 415 531**	**789 000 450 609**	**22 292 678 512 329**	**379 856 554 324 947**
Temps	**<1s**	**<1s**	**<6s**	**~20s**	**~15min**

Exercice 6

Le fichier "exercice6.c" permet de créer pour tous les fichiers la distribution des degrés dans le répertoire /ressource/Degrees/Distribution.

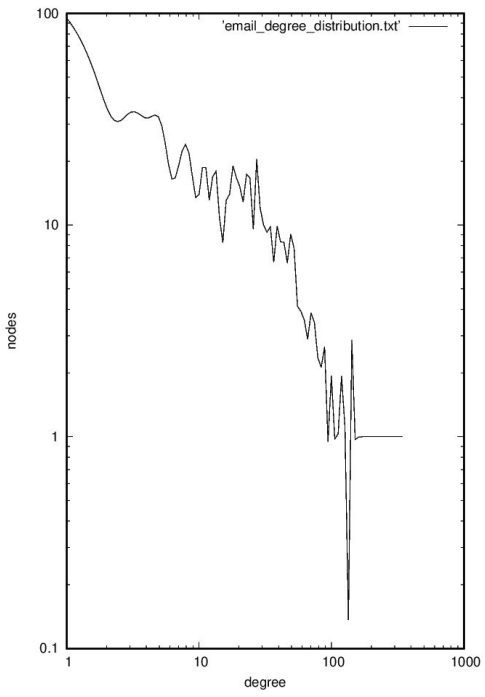


Figure 1 : Email

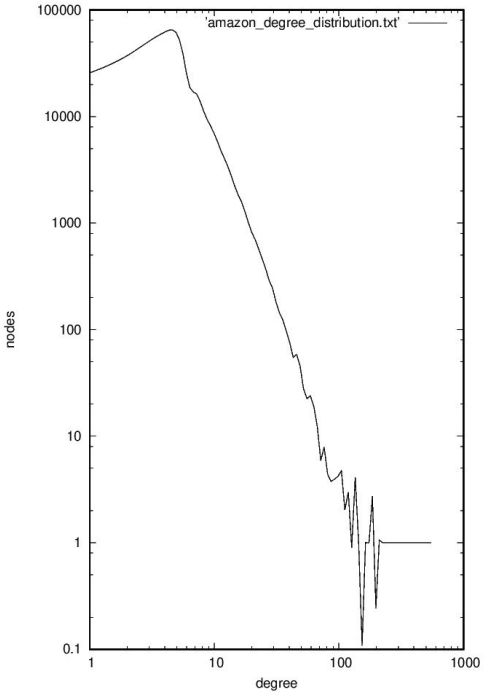


Figure 2 : Amazon

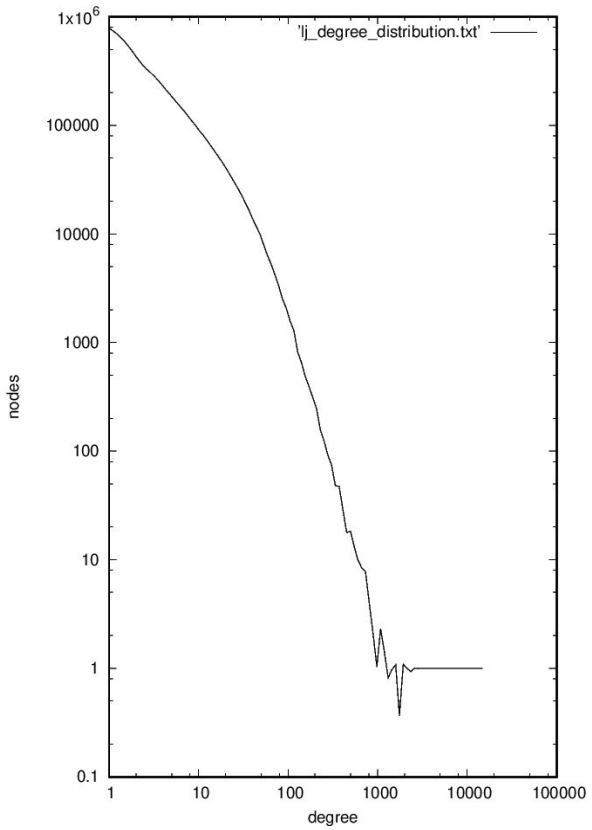


Figure 2 : LJ

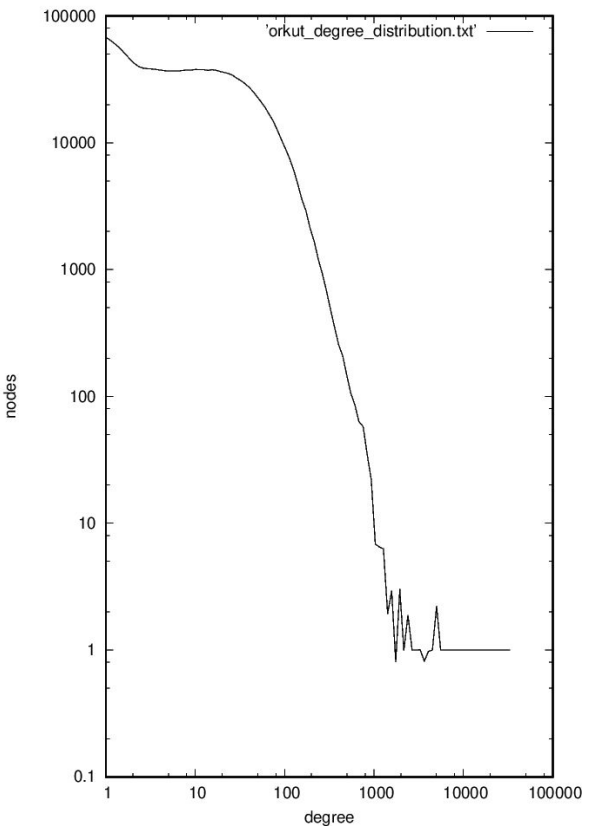


Figure 3 : Orkut

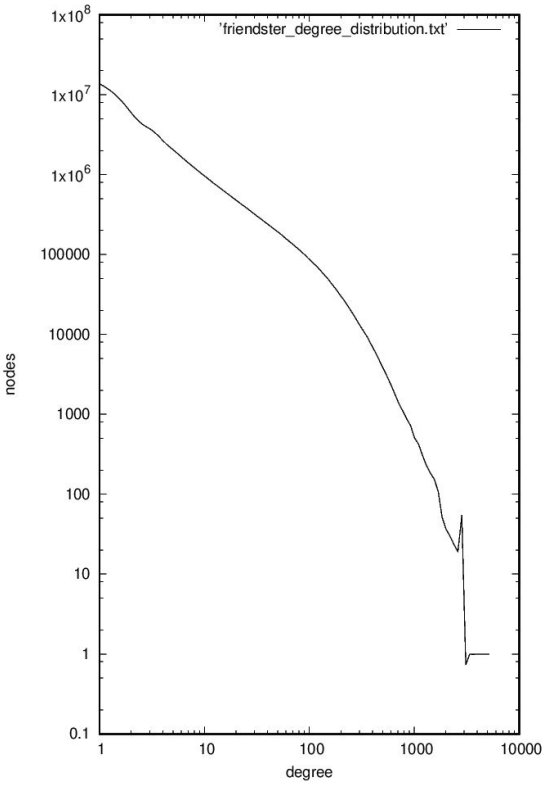


Figure 4 : Friendster

Pour tous les graphes on peut voir que la majorité des noeuds ont un degré faible, et que plus les degrés sont élevés, moins il y a de noeuds ayant ce degré.

Exercice 7

Le graphe ci-dessous représente le temps nécessaire au calcul des différentes structures de stockage des adjacences. Les résultats sont obtenus par les fichiers exercice7_1.c pour la liste d'arêtes, exercice7_2.c pour la matrice et exercice7_3.c pour le tableau d'adjacence.

	Email	**Amazon**	**Lj**	**Orkut**	**Friendster**
liste d'arêtes	**<1s**	**<1s**	**6s**	**20s**	**pas assez de mémoire**
matrice	**<1s**	**3s**	**pas assez de mémoire**	**pas assez de mémoire**	**pas assez de mémoire**
tableau d'adjacence	**<1s**	**<1s**	**50s**	**67s**	**pas assez de mémoire**

Stocker le graphe sous forme de matrice est bien trop coûteux en mémoire, et une liste d'arêtes ne permettrait pas d'accéder aux voisins d'un sommet en temps raisonnable, un tableau d'adjacence est donc le meilleur moyen de stocker le graphe.

Exercice 8

Le fichier exercice8.c permet de calculer le nombre de noeuds dans le plus grand composant du graphe, ainsi que le pourcentage par rapport au nombre total de noeuds du graphe.

	Email	**Amazon**	**Lj**	**Orkut**	**Friendster**
Nb noeuds connectés	**986**	**334 863**	**3 997 962**	**3 072 441**	**pas assez de mémoire**
Fraction	**100%**	**100%**	**100%**	**100%**	**pas assez de mémoire**
Borne inférieure diamètre	**7**	**47**	**21**	**9**	**pas assez de mémoire**

Les graphes sont donc fortement connexes.

Exercice 9

Le fichier *"exercice9.c"* permet de calculer l'ensemble des données pour chaque graph.
Pour Email, nous calculons le coefficient de groupement moyen sur le fichier ne contenant pas de doublon.

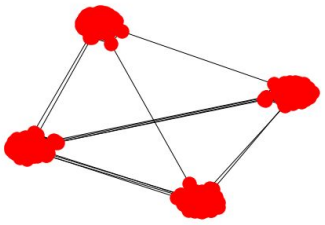
	Email	Amazon	Lj	Orkut	Friendster
noeuds	**986**	**334 863**	**3 997 962**	**3 072 441**	**65 608 366**
Triangles	**105 461**	**667 129**	**177 820 130**	**627 584 181**	**pas assez de mémoire**
Triplets	**1 183 216**	**9 752 186**	**4 255 768 195**	**45 625 466 571**	**pas assez de mémoire**
Transitivité	**0.2674**	**0.2052**	**0.1254**	**0.0413**	**pas assez de mémoire**
Average Clustering Coefficient	**0.4070**	**0.3967**	**0.2843**	**0.1667**	**pas assez de mémoire**
Temps	**<1s**	**1s**	**50s**	**5m40**	**pas assez de mémoire**

Pour friendster nous ne disposons pas d'assez de mémoire sur les machines de la PPTI pour le calculer.

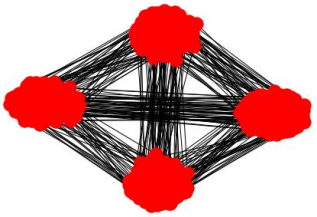
TME N°4

Exercice 1

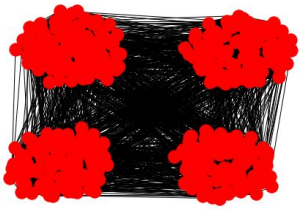
le fichier *"generaterGraph.py"* permet de generer un graphe de 400 noeuds séparé en 4 cluster de 100 noeuds. La probabilité *p* d'avoir un lien entre deux noeuds d'un même cluster est fixée à 0.5



q=1/8000



q=1/400



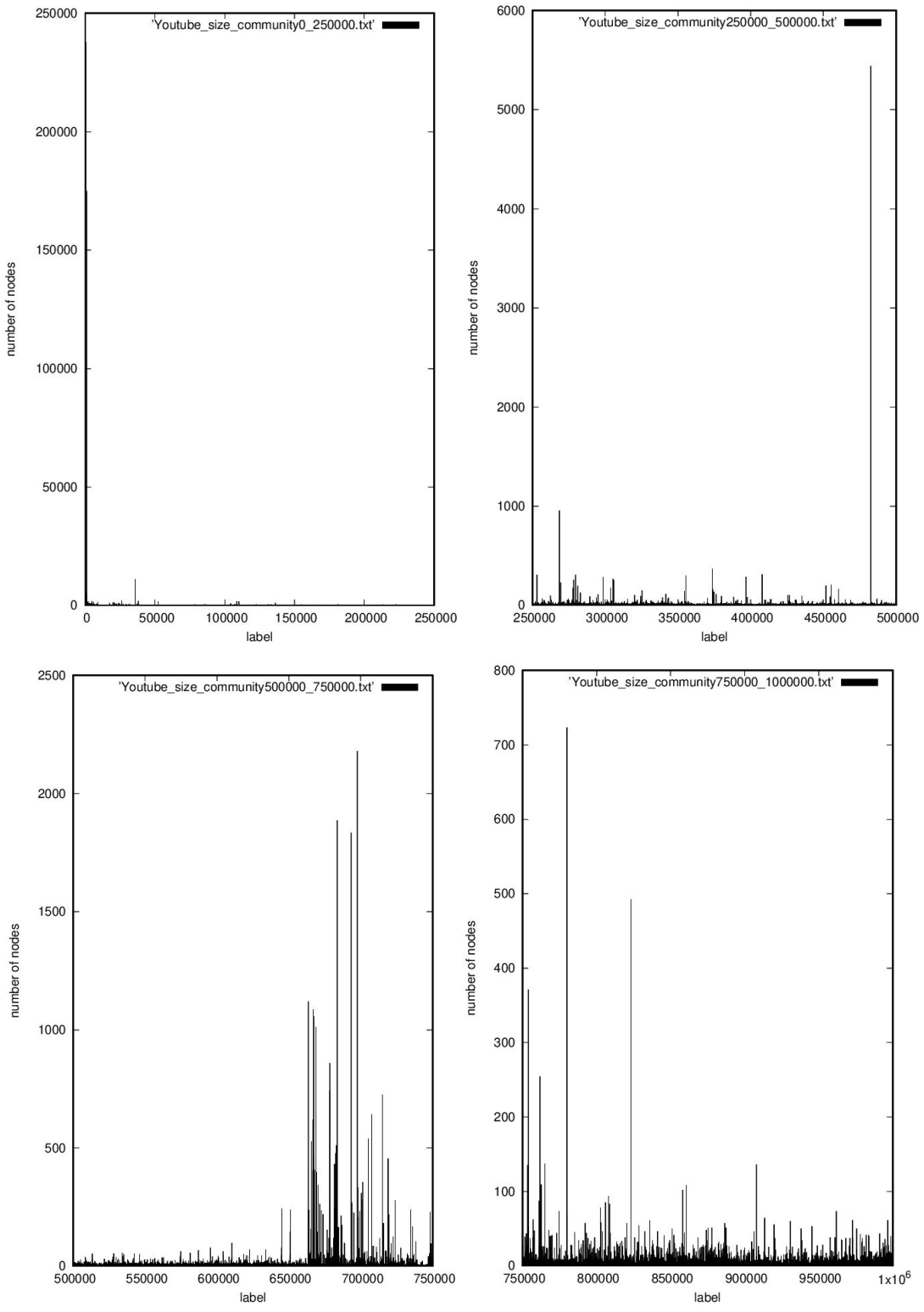
q=1/100

Dans chaque cluster chaque noeuds a une probabilité *p* d'être connecté à un autre noeud.
Entre deux noeuds appartenant à deux clusters différents, la probabilité de connexion est *q*.
Nous pouvons observer que, plus *q* s'approche de *p*, plus le nombre d'arêtes entre les clusters augmente.

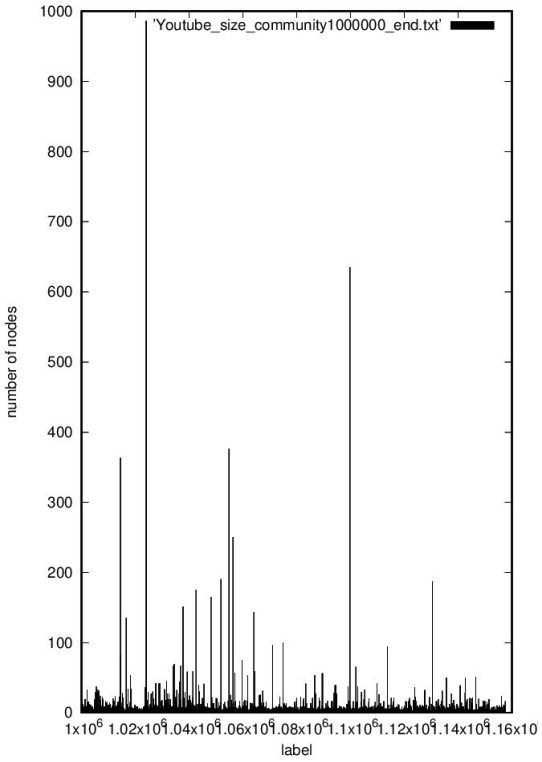
Exercice 2

Nous utilisons un tableau d'adjacence.

Le fichier *"exercice2.c"* effectue un label propagation. la répartition des labels est divisée en 5 graphes avec 250 000 labels sur chaque graphe pour pouvoir utiliser des échelles différentes en *Y* et mieux observer la répartition.

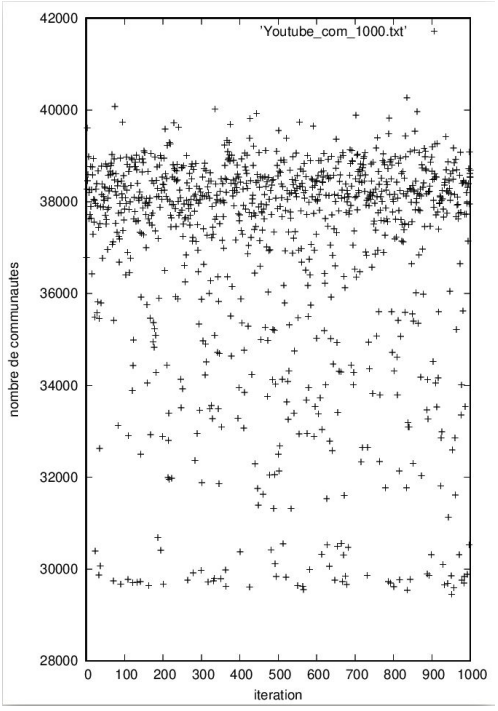


Nombre de noeuds par Label



Nombre de noeuds par Label

De ces graphes, on peut observer qu'il n'y a que quelques pics représentants les communautés majoritaires (La plus grande étant environ 240 000 sur le premier graphe). autours de ces grandes communautés, il y a peu de communautés importantes.



Nombre de communautés pour 1000itérations

Après 1000 itération on voit nettement une convergence vers 30 000.

Exercice 3

Après exécution de l'algorithme de Louvain nous obtenons ces résultats.
level 0: 1157828 nodes
level 1: 194554 nodes
level 2: 57180 nodes
level 3: 35711 nodes
level 4: 31165 nodes
level 5: 30487 nodes
level 6: 30372 nodes
level 7: 30367 nodes

Le dernier niveau nous donne donc le nombre de communautés , soit 30367 communauté, ce qui est en concordance avec la convergence de label propagation.

L'algorithme de Louvain est le plus efficace avec une complexité en O(*n log(n)*) par rapport au label propagation qui a une complexité en O(*n* nb_iterations*) où nb_iterations et le nombre de fois que les labels sont propagé après avoir été mélangé de manière aléatoire.

TME N°5

Exercice 1

Le fichier *"exercice1.c"* permet d'effectuer la recherche des cinq pages ayant les meilleurs valeurs de PageRank, ainsi que les cinq moins bonnes.

Cinq meilleurs:

1 -> United States PageRank: 0.003641
2 -> United Kingdom PageRank: 0.001593
3 -> Germany PageRank: 0.001365
4 -> 2007 PageRank: 0.001362
5 -> 2006 PageRank: 0.001358

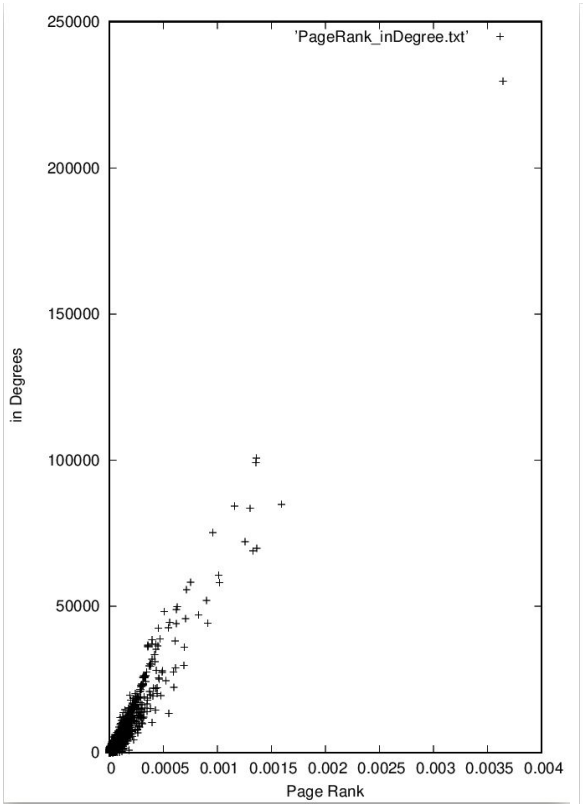
Cinq des moins bons :

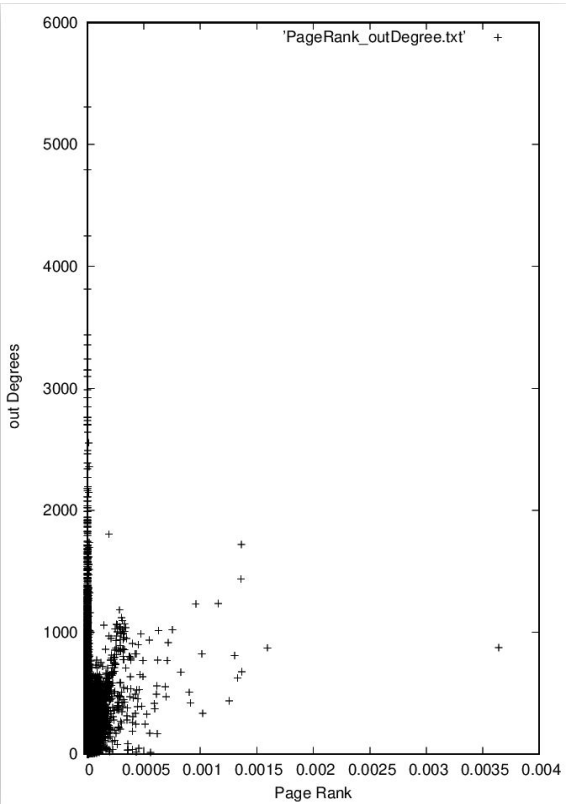
1 -> Military of Malawi
2 -> Military of Mozambique
3 -> Netwar
4 -> Philosophical view
5 -> Transport in São Tomé and Príncipe

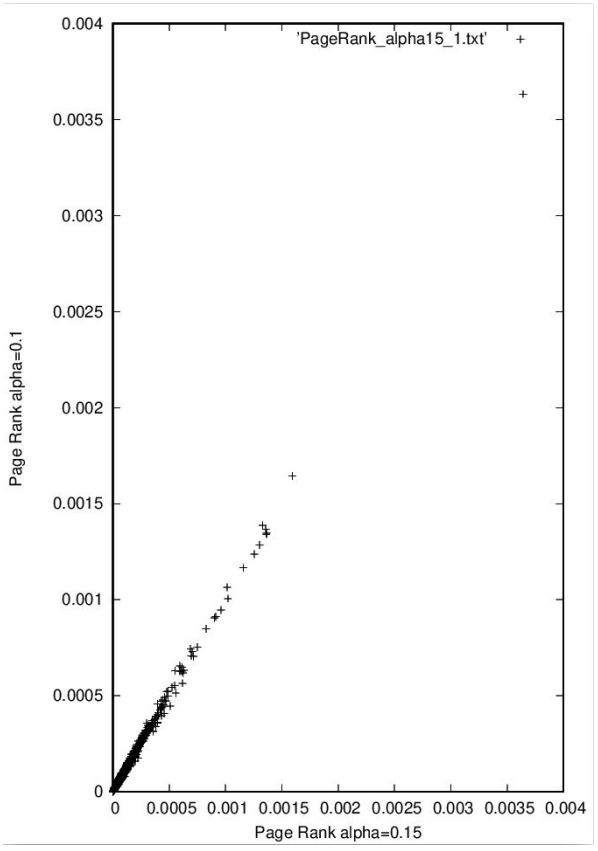
Pour les plus faible, plusieurs pages possèdent le même pageRank ce qui rend la recherche non déterministe.

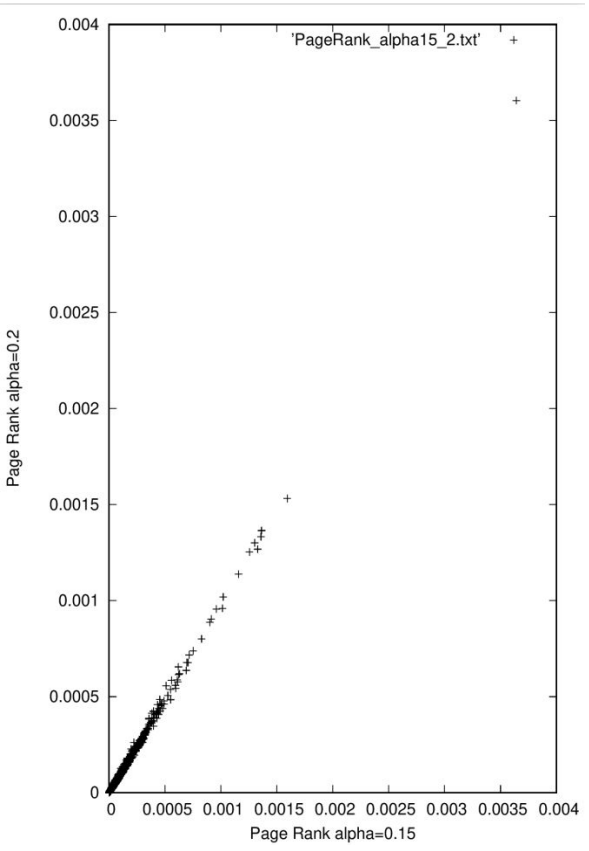
Une vingtaine d'itérations semblent suffisantes pour obtenir une convergence des valeurs du PageRank.

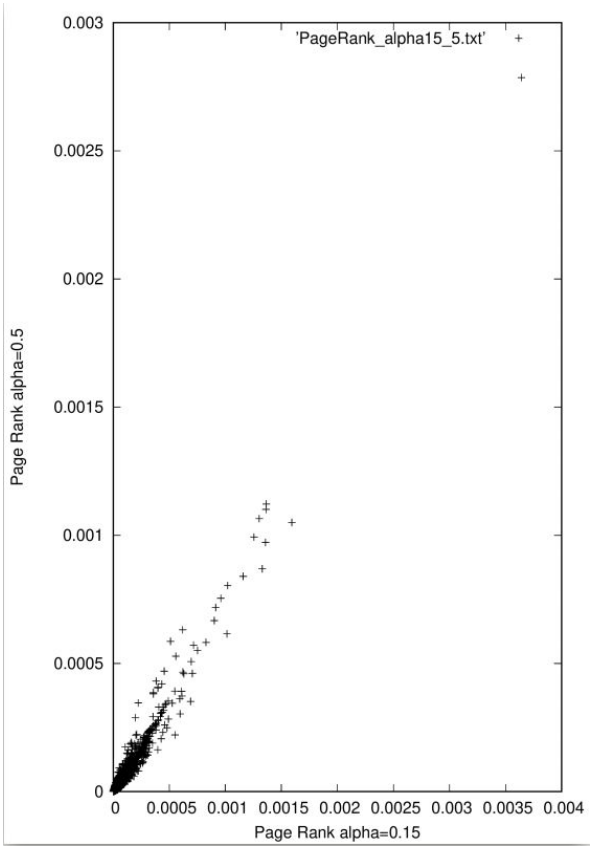
Exercice 2

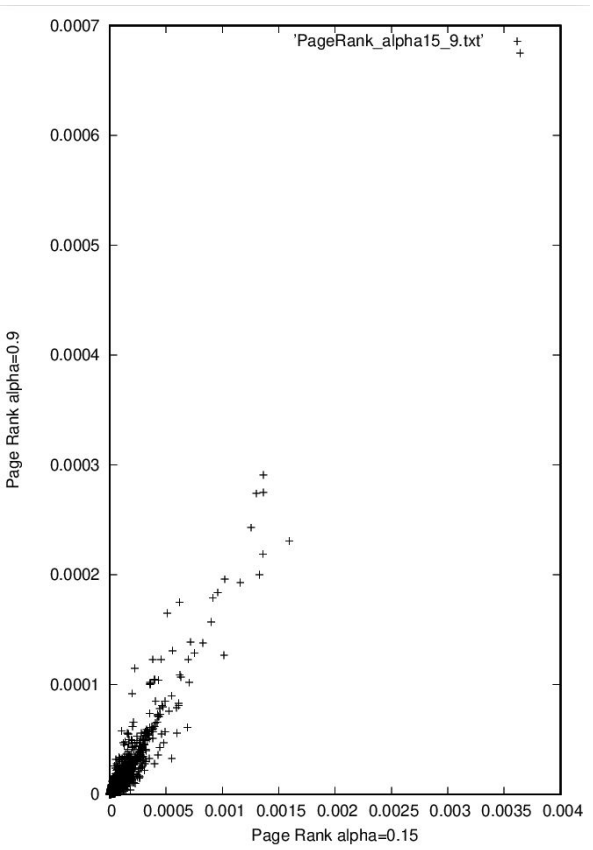

'PageRank_inDegree.txt'
in Degrees
Page Rank
250000
200000
150000
100000
50000
0
0 0.0005 0.001 0.0015 0.002 0.0025 0.003 0.0035 0.004


'PageRank_outDegree.txt'
out Degrees
Page Rank
6000
5000
4000
3000
2000
1000
0
0 0.0005 0.001 0.0015 0.002 0.0025 0.003 0.0035 0.004


'PageRank_alpha15_1.txt'
Page Rank alpha=0.1
Page Rank alpha=0.15
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0 0.0005 0.001 0.0015 0.002 0.0025 0.003 0.0035 0.004


'PageRank_alpha15_2.txt'
Page Rank alpha=0.2
Page Rank alpha=0.15
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0 0.0005 0.001 0.0015 0.002 0.0025 0.003 0.0035 0.004





Il ne nous apparaît pas utile d'utiliser une échelle logarithmique.

Nous pouvons conclure plusieurs choses de ces différents affichages :

- Plus un noeud a un in-degree élevé, plus haut sera son PageRank.
- Le out-degree d'un noeud n'a pas de conséquence sur son PageRank.
- Plus la valeur α est élevée, moins le PageRank sera "précis".

TME N°6

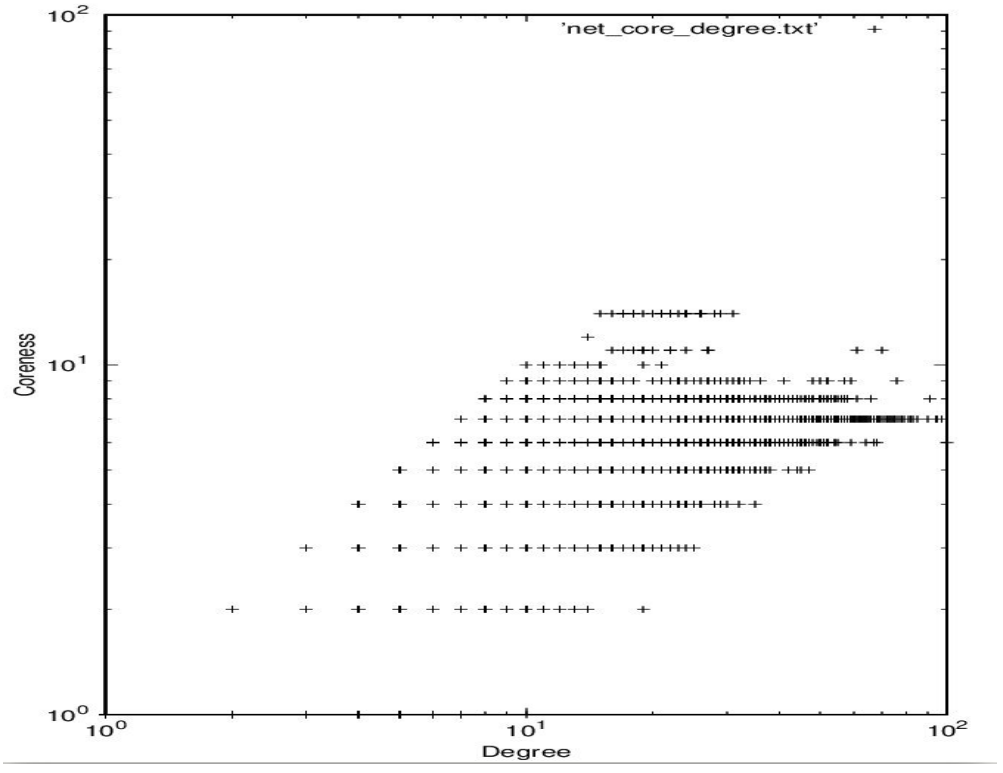
Exercice 1

Le fichier *"exercice1.c"* permet de calculer le plus grand K-core des différents graphes.

FICHIER	Email	Amazon	LJ	Orkut	Friendster
Core value	**34**	**6**	**360**	**253**	**pas assez de mémoire**
densité moyenne de degré du core le plus dense	**27.56**	**3.8571**	**190.9844**	**227.8727**	**pas assez de mémoire**
densité d'arêtes du core le plus dense	**0.2428**	**0.5934**	**0.9921**	**0.0176**	**pas assez de mémoire**
taille du core	**228**	**14**	**386**	**25829**	**pas assez de mémoire**

La construction du tas min se faisant avec un ConsIter et non un ajout successif peut donner des valeurs différentes pour la taille du sous graphe le plus dense et donc sa densité d'arêtes. Comme par exemple si deux noeuds ont le même degré mais pas les même voisins.

Exercice 2



Nous pouvons observer une forte concentration des citations d'auteurs koréens entre eux.

Exercice 3

t = 10				
FICHIER	**EMAIL**	**AMAZON**	**LJ**	**ORKUT**
densité moyenne de degrés	**27.2891**	**4.0100**	**188.4374**	**225.0381**
densité d'arêtes	**0.2200**	**0.0008**	**0.1525**	**0.0136**
taille	**249**	**9069**	**2471**	**32972**

t = 100				
FICHIER	**EMAIL**	**AMAZON**	**LJ**	**ORKUT**
densité moyenne de degrés	**27.5657**	**4.8020**	**193.2015**	**227.8029**
densité d'arêtes	**0.2428**	**0.1010**	**0.3803**	**0.1676**
taille	**228**	**96**	**1017**	**2714**

t = 1000				
FICHIER	**EMAIL**	**AMAZON**	**LJ**	**ORKUT**
densité moyenne de degrés	**27.5669**	**4.8041**	**193.5136**	**227.8727**
densité d'arêtes	**0.2472**	**0.1000**	**0.8816**	**0.0174**
taille	**224**	**97**	**440**	**26054**

Pour t = 10

Nous pouvons observer que les valeurs diffèrent de l'exercice 1 légerement pour email mais beaucoup pour amazon lj et orkut.

Pour t = 100

Nous pouvons observer que les valeurs sont les mêmes pour email, proches pour orkut mais toujours éloignée pour amazon et lj

Pour t = 1000

Nous pouvons observer que les valeurs pour Email commencent a différer, mais pour lj se rapprochent, et reste quasi identique pour amazon

Nous pouvons en déduire que la précision dépend de la taille du graphe, et que le nombre de round pour *MkScore* en dépends.

Pour prouver brièvement que le score de densité le plus élevé est une borne supérieure, nous pouvons dire que le score le plus élevé conduit à ce que le noeud lui correspondant soit le premier parcouru lors du parcours des noeuds en ordre décroissant. Puisque par la suite il y aura de plus en plus de noeuds, cette valeurs sera toujours supérieure.

Score de densité le plus élevé				
FICHIER	**EMAIL**	**AMAZON**	**LJ**	**ORKUT**
densité moyenne de degrés(t = 1000)	**27.5669**	**4.8041**	**193.5136**	**227.8727**
max	**27.5830**	**4.8070**	**193.5910**	**228.0322**