PROJET
UE Méthodes de ranking et recommandations
Sujet 5 : Simulation d'un Google Bombing

Maxime Gonthier - Laureline Martin

25 avril 2019

Table des matières

1 Introduction **2**

2 Manuel utilisateur **2**

3 Tests initiaux et hypothèses **4**
3.1 Explications du code 4
3.2 Résultats des tests initiaux 6
3.2.1 Stanford.txt sans modification 6
3.2.2 Résultats 6
3.3 Conclusion des tests initiaux 7
3.4 Hypothèses 7

4 Tests avec un nombre de graphes générés aléatoirement **7**
4.1 Explication du code 8
4.2 Résultats des tests 9
4.3 Conclusion sur l'impact d'un nombre aléatoires de structures 10

5 III. Expériences avec un graphe de degré aléatoire **10**
5.1 Résultats des tests 10
5.2 Analyses 12

6 IV. Expériences sur l'efficacité **12**
6.1 Résultats des tests 12
6.2 Analyses 13

7 Conclusions **15**

8 Annexe **15**

1 Introduction

Le but de ce projet est de simuler, d'évaluer et de proposer la méthode la plus efficae pour un Google Bombing. Il s'agit d'une technique d'attaque sur le graphe du web permettant aux attaquants d'augmenter la pertinence d'une page cible.
Pour cela, plusieurs attaquants créent des pages et y insèrent des liens vers une page cible. Plusieurs structures de pages attaquantes sont possibles :

- seul
- complet
- anneau
- arbre

Nous testerons ces différentes structures sur des pages cibles dont la pertinence, attribuée par l'algorithme Pagerank, est différente. Ainsi, nous déterminerons quelle est la structure la plus avantageuse pour obtenir la pertinence la plus forte possible.
Dans un premier temps, nous allons effectué des tests simples en utilisant les quatre types de structures énoncé ci-dessus ainsi que trois cibles de pertinences différentes (pertinence forte, moyenne et faible). Nous en déduirons des hypothèses sur l'efficacité de chaque structure et de l'impact sur la pertinence du sommet cible.
Dans un second temps, nous étudierons l'impact sur la pertinence du sommet cible de graphes générés en nombre aléatoire pour étudier quelle structure est la plus efficace.
Dans un troisième temps, nous testerons la même choses avec cette fois-ci un graphe dont les arcs le reliant à lui-même sont générés aléatoirement. Cela signifie que le degré du graphe est généré aléatoirement. Dans un dernier temps, nous ferons varier le nombre de sommets du graphe attaquant pour en déduire son efficacité.

2 Manuel utilisateur

Notre application se lance grâce à la commande `make all` suivi d'un fichier " .txt" à modifier.
Pour effacer les fichiers .o et l'executable : `make clean`.
Pour compiler : `make compil`. On peut ensuite lancer le programme avec `./ranking` suivi du fichier à modifier.
Une fois l'application lancée :

```
user@debian:~/Bureau/Ranking$ make clean
user@debian:~/Bureau/Ranking$ make compil
user@debian:~/Bureau/Ranking$ make all Stanford/Stanford.txt
./ranking Stanford/Stanford.txt
Random cible si besoin : 241859

pages : 281903
lien : 2312497
Entrez la structure que vous voulez insérer :
 1 pour un sommet seul
 2 pour un anneau
 3 pour un graphe complet
 4 pour un arbre
 5 pour un nombre d'anneaux aléatoire
 6 pour un nombre de graphes complet aléatoire
 7 pour un nombre d'arbres aléatoire
 8 pour un arbre a attaquant unique
 9 pour un graphe de 100 sommets avec un degré aléatoire
 10 pour un graphe avec une certaine probabilité de connecter les sommets entre eux
2
Entrer le nombre de structure que vous voulez ajouter :
1
Entrez le nombre de sommets par structure a ajouter :
150
Entrez la pertinence de la cible :
 1 pour une pertinence forte
 2 pour une pertinence moyenne
 3 pour une pertinence faible
 4 Pour entrer manuellement le sommet cible
1
Entrez le numéro du sommet cible :
1
282053 pages
2312797 liens
File read!
132 itérations
Done!
La pertinence du sommet cible est : 0.000207197
Le temps d'execution est de : 34.565403 secondes
```

Dans un premier temps, l'utilisateur doit choisir la structure qu'il veut ajouter au graphe du web qu'il aura préalablement donné. Il y a des structures simples (arbre, anneau, graphe complet sommet seuls), des structures moins conventionelles (arbre à attaquant unique, arbre où seul le sommet racine pointe vers la cible). L'utilisateur peut aussi ajouter un nombre de structures aléatoires (avec une limitation de 100 sommets). Il peut également insérer un graphe de degré aléatoire ou un graphe dont le degré est une probabilité.
Dans un deuxième temps, l'utilisateur doit entrer le nombre de structures qu'il souhaite. Attention, cela ne s'applique pas pour les structures numéro 5, 6 et 7.
Dans un troisième temps, l'utilisateur doit entrer le nombre de sommets de la/les structure(s) à ajouter.
Dans un quatrième temps, l'utilisateur doit entrer le sommet cible. Le numéro du sommet cible est demandé une seconde fois pour savoir quel sommet

sera donné en résultat.

Le programme affiche le nombre de pages, le nombre de liens, le temps d'éxecution et la pertinence du sommet cible.

3 Tests initiaux et hypothèses

L'algorithme power calculant les pertinences est contenu dans le fichier `ranking.c`. Il n'est pas détaillé dans ce rapport mais le fichier est commenté.

On considère ici le graphe du web Stanford.txt. Ce graphe est modifié par l'ajout de sommets et d'arcs afin d'augmenter la valeur d'un sommet ciblé. On considère que :

- Les sommets représentent les pages du web.
- Les arcs représentent les liens dirigeant vers d'autres pages.
- Les valeurs des sommets représentent les pertinences calculés par l'algorithme pagerank.
- Le sommet cible représente la page dont on souhaite augmenter la pertinence.
- 100 sommets sont ajoutés pour chaque test.

3.1 Explications du code

Les fonctions `ajoutanneau`, `ajoutsommetseul`, `ajoutcomplet` et `ajoutarbre` crée des structures à la fin du fichier. Ces fonctions ont pour arguments :

- `nbajout` : le nombre de sommets par structures,
- `nom` : le nom du fichier contenant le graphe du web,
- `nbpages` et `nbliens` : le nombre de pages et de liens du graphe du web,
- `perticible` : la cible à attaquer,
- `nbstructure` : le nombre de structures à ajouter.

La fonction `ajoutanneau` :

```
FILE *g = fopen(nom,"a");
for (int y = 0; y < nbstructure; y++) {
        for (i = 1; i < nbajout; i++) {
                fprintf(g, "%d %d %d 0.500000 %d 0.500000\n",
                        nbpages+i+(y*nbajout), degre,
                        nbpages+i+1+(y*nbajout), cible);
        }
        //ecriture du dernier sommet qui se relie au
        //premier sommet de l'anneau
        fprintf(g, "%d %d %d 0.500000 %d 0.500000\n",
```

```
                nbpages+i+(y*nbajout), degre,
                nbpages+1+(y*nbajout), cible);
}
```

Ici y et i vont incrémenter respectivement le nombre de structures et le sommet suivant à écrire. i commence à 1 pour écrire le numéro de la page suivant le dernier numéro de page du graphe. Les nouveaux nombres de pages et de liens du graphe sont écrits de cette mainière :

```
fprintf(h, "%d %d", nbpages+(nbajout*nbstructure),
 nbliens+(nbajout*nbstructure)*2);
```

La fonction `ajoutcomplet` :

```
for (int y = 0; y < nbstructure; y++) {
        for (i = 1; i < nbajout + 1; i++) {
                fprintf(g, "%d %d", nbpages+i+(y*nbajout), degre);
                for(j = 1; j < nbajout + 1; j++){
                        if (nbpages+j == nbpages+i) {}
                        else {
                                fprintf(g, " %d %f",
                                        nbpages+j+(y*nbajout), x);}
                }
                fprintf(g, " %d %f\n", cible, x);
}
```

Le premier `fprintf` écrit le numéro de la page courante et son degré. Le second `fprintf` écrit les liens entre cette page et toutes les autres pages du graphe complet. Le `if` sert à vérifier que l'on écrit pas un lien de la page vers elle-même.
La fonction `ajoutarbre` :

```
for (int y = 0; y < nbstructure; y++) {
        // Initialisation du sommet racine
        racine = nbpages + 1+(y*nbajout);
        fprintf(g, "%d %d %d 1.000000\n", racine, 1, cible);
        // Sommets 2 a 2 (arbre binaire) et on pointe
        //vers le sommet pere
        // lui meme calcule par sa distance a la racine
        for (i = 1; i < nbajout - 1; i+=2) {
                fprintf(g, "%d %d %d 0.500000 %d 0.500000\n",
                        racine+i, degre, racine + compteur, cible);
                fprintf(g, "%d %d %d 0.500000 %d 0.500000\n",
                        racine+i+1, degre, racine + compteur, cible);
                compteur++;
        }
        // Le dernier sommet dans le cas d'un nombre
```

```
        //PAIR de sommets
        // a entrer (pair car le sommet racine est deja
        //ecris hors de la boucle)
        if (nbajout%2 == 0){
                fprintf(g, "%d %d %d 0.500000 %d 0.500000\n",
                        racine+i, degre, racine + compteur, cible);
        }
        //on reinitialise le compteur car on va commencer
        //un nouvel arbre
        compteur = 0;
}
```

`racine` représente le sommet racine et permet d'écrire les pages suivantes. Les pages sont écrites deux à deux et calculées avec leur distance par rapport à `racine`. Elles sont écrites deux à deux car on écrit les deux fils d'un sommet père à chaque itération. `compteur` est utilisé dans le cas de la création de plusieurs arbres.

3.2 Résultats des tests initiaux

3.2.1 Stanford.txt sans modification

281903 pages
2312497 liens
132 itérations
27.627466 secondes

Voici les pertinences de base des trois sommets étudiés :
Pertinence forte : Page 280545 - 9.96199e-05
Pertinence moyenne : Page 281466 - 7.53954e-06
Pertinence faible : Page 281574 - 6.05222e-07

3.2.2 Résultats

Pertinences des sommet cible	Forte	Moyenne	Faible
Sans modification	9,96E-05	7,54E-06	6,05E-07
Difference sans modifcation	0,00E+00	0,00E+00	0,00E+00
Attaque a 100 sommets seuls	1,82E-04	5,89E-05	5,19E-05
Differences avec 100 sommets seuls	8,25E-05	5,13E-05	5,13E-05
Attaque avec un anneau de 100 sommets	1,71E-04	5,22E-05	4,52E-05
Differences avec un anneau a 100 sommets	7,17E-05	4,46E-05	4,46E-05
Attaque avec un graphe complet de 100 sommets	1,05E-04	1,08E-05	3,84E-06
Differences avec un graphe complet a 100 sommets	5,17E-06	3,23E-06	3,24E-06
Attaque avec un arbre de 100 sommets	1,72E-04	5,25E-05	4,55E-05
Differences avec un arbre a 100 sommets	7,22E-05	4,49E-05	4,49E-05
Attaque avec un arbre a attaquant unique de 100 sommets	1,38E-04	3,17E-05	2,47E-05
Differences avec un arbre a attaquant seul a 100 sommets	3,88E-05	2,41E-05	2,41E-05

3.3 Conclusion des tests initiaux

Cinq structures différentes sont ajoutées : des sommets seuls, un graphe complet, un anneau, un arbre et un arbre à attaquant unique(seul le sommet racine est relié à la cible). L'objectif de cette structure est de minimiser le nombre de sommets reliés à la cible pour minimiser les chances de se faire repérer lors de l'attaque.
On observe pour chaque cible que la structure la plus efficace est le sommet seul, suivi de l'arbe, de l'anneau, de l'arbre à attaquant unique et enfin du graphe complet.
La raison de cette ordre serait que pour des sommets seuls, la probabilité de pointer sur la cible est de $1/1$ alors qu'elle est de $1/n$ pour un graphe complet avec n le nombre de sommets. Ainsi la probabilité est dilué et donc la cible en sera moins modifié.
On observe que pour chaque structure la différence de modification de la pertinence est presque identique entre les cibles faibles et moyennes. Ainsi on peut en déduire que modifier la pertinence d'une cible à 10^{-06} ou 10^{-05} est de même difficulté.
Pour le sommet seul, on observe que la modification de pertinence (la différence) pour une cible forte, moyenne ou faible sont respectivements $8, 25.10^{-05}$, $5.13.10^{-05}$ et $5, 13.10^{-05}$. On peut donc en déduire que modifier la pertinence d'une cible est plus significative quand la cible est déjà à pertinence forte.

3.4 Hypothèses

On suppose que les structures les plus efficaces sont les sommets seuls et les arbres. On suppose que changer la pertinence d'une cible devient plus difficile à partir d'une cible de pertinence $X.10^{-05}$ et que c'est assez identique pour les cibles de pertinences $X.10^{-06}$ et $X.10^{-07}$.

4 Tests avec un nombre de graphes générés aléatoirement

Nous allons désormais insérer des graphes générés aléatoirement. Ce qui est aléatoire n'est pas la structure des graphes mais le nombre de garphes générés.
L'objectif de cette démarche est de déterminer quel structure est la plus efficace globalement.
C'est a dire quelle structure influe le plus sur la pertinence quelle que soit la situation.
De plus les resultats nous aiderons aussi a determiner l'impact qu'a une certaine structure sur la cible de manière plus générale que dans les cas prédéfinis de la partie précedente. Le nombre de sommet des graphes ajoutées est fixé a 100. La cible est également fixé ainsi que la structure des graphes

ajoutées. On va par exemple insérez 5 graphes complet de nombre de sommet respectivement : 25, 10, 5, 6 et 4. Tous reliées au sommet cible. Dans un premier temps nous allons expliquer le code derrière cette démarche puis nous analyserons les résultats.

4.1 Explication du code

Tous est dans le fichier *ajoutsommetsattanquants.c*. Les fonctions utilisées sont *ajoutanneaualeatoire*, *ajoutcompletaleatoire* et *ajoutarbrealeatoire*. Ces fonctions reprennent en partie le code des trois fonctions presque eponyme décrite précedemment. Regardons ce qui a changé.

```
while(nbsommetrestant > 0) {
        nbajout = rand()%(100-nouveausommets);
        if (nbajout <= 3) { nbajout = 3; }
        nbsommetrestant -= nbajout;
```

nbajout représente le nombre de sommet que l'on va ajouter dans la première structure créer. Il choisis donc un nombre aléatoire entre 0 et 100 car le nombre de sommet total est limité a 100. Si le nombre choisis est inférieur a 3 on le fixe a 3 car créer des anneau ou des graphes complets de tailles inférieure a 3 reviens juste a créer des sommets seuls. *nbajout* est enlevé au nombre de sommet restant a ajouter. Puis on lance la construction de la structure comme vu précedemment avec *nbajout* représentant le nombre de sommets. A la fin de cette itération *nbajout* reprends un nombre aléatoire qui cette fois prend une valeur entre 0 et 100 moins le nombre de sommets ajouté precedement représenté par *nouveausommets*.

4.2 Résultats des tests

Type de graphe	Sans modification	Anneau	Graphe complet	Arbre
Pertinences	9,96E-05	1,7279E-04	1,1072E-04	1,7433E-04
	9,96E-05	1,7207E-04	1,1198E-04	1,7345E-04
	9,96E-05	1,7279E-04	1,2006E-04	1,7417E-04
	9,96E-05	1,7135E-04	1,1040E-04	1,7326E-04
	9,96E-05	1,7135E-04	1,1179E-04	1,7372E-04
	9,96E-05	1,7135E-04	1,1042E-04	1,7387E-04
	9,96E-05	1,7279E-04	1,1173E-04	1,7378E-04
	9,96E-05	1,7207E-04	1,1234E-04	1,7456E-04
	9,96E-05	1,7207E-04	1,1073E-04	1,7265E-04
	9,96E-05	1,7279E-04	1,1796E-04	1,7477E-04
	9,96E-05	1,7207E-04	1,1729E-04	1,7273E-04
	9,96E-05	1,7207E-04	1,1960E-04	1,7287E-04
	9,96E-05	1,7135E-04	1,2463E-04	1,7450E-04
	9,96E-05	1,7207E-04	1,1456E-04	1,7338E-04
	9,96E-05	1,7207E-04	1,1443E-04	1,7474E-04
	9,96E-05	1,7207E-04	1,1335E-04	1,7332E-04
	9,96E-05	1,7135E-04	1,1246E-04	1,7427E-04
	9,96E-05	1,7207E-04	1,1346E-04	1,7272E-04
	9,96E-05	1,7279E-04	1,2473E-04	1,7454E-04
	9,96E-05	1,7207E-04	1,1858E-04	1,7256E-04
	9,96E-05	1,7279E-04	1,1057E-04	1,7301E-04
	9,96E-05	1,7135E-04	1,1457E-04	1,7436E-04
	9,96E-05	1,7135E-04	1,1483E-04	1,7329E-04
	9,96E-05	1,7279E-04	1,1558E-04	1,7326E-04
	9,96E-05	1,7207E-04	1,1781E-04	1,7281E-04
	9,96E-05	1,7279E-04	1,2091E-04	1,7390E-04
	9,96E-05	1,7279E-04	1,2048E-04	1,7202E-04
	9,96E-05	1,7279E-04	1,1457E-04	1,7362E-04
	9,96E-05	1,7279E-04	1,1460E-04	1,7410E-04
	9,96E-05	1,7135E-04	1,1446E-04	1,7333E-04

La cible est de pertinence forte.

Nb tirages	30	30	30	30
Espérance	9,96E-05	1,72E-04	1,15E-04	1,74E-04
Ecart type	0	5,75759E-07	4,06743E-06	7,347E-07
Intervalle de confiance : 95% de la moyenne				
t-normale	1,645	1,645	1,645	1,645
Erreur	0	1,7292E-07	1,22159E-06	2,2065E-07

En regardant les espérances on observe que c'est l'anneau est l'abre qui impactent le plus la pertinence de la cible. On observe aussi que les erreurs sont inférerieurs à 0,000015 on peut donc faire confiance aux résultats.
Regardons les trois courbes en annexe.
La courbe rouge représente la loi normale, la noire est la fonction de densité. Comparons ces résultats avec les résultats des tests initiaux. Les résultats sont comparable car dans les deux cas 100 sommets sont ajoutés. Dans les tests initiaux les pertinence apres attaque pour l'anneau, le graphe complet et l'arbre étaie,t respectivement 1.71e-4, 1.05e-4 et 1.72e-4. On remarque donc qu'il n'y a que 0.6% de différence pour l'anneau et 1.1% pour l'arbre. En revanche pour le graphe complet il y a 9.5% de différence ce qui est bien plus intéressant.

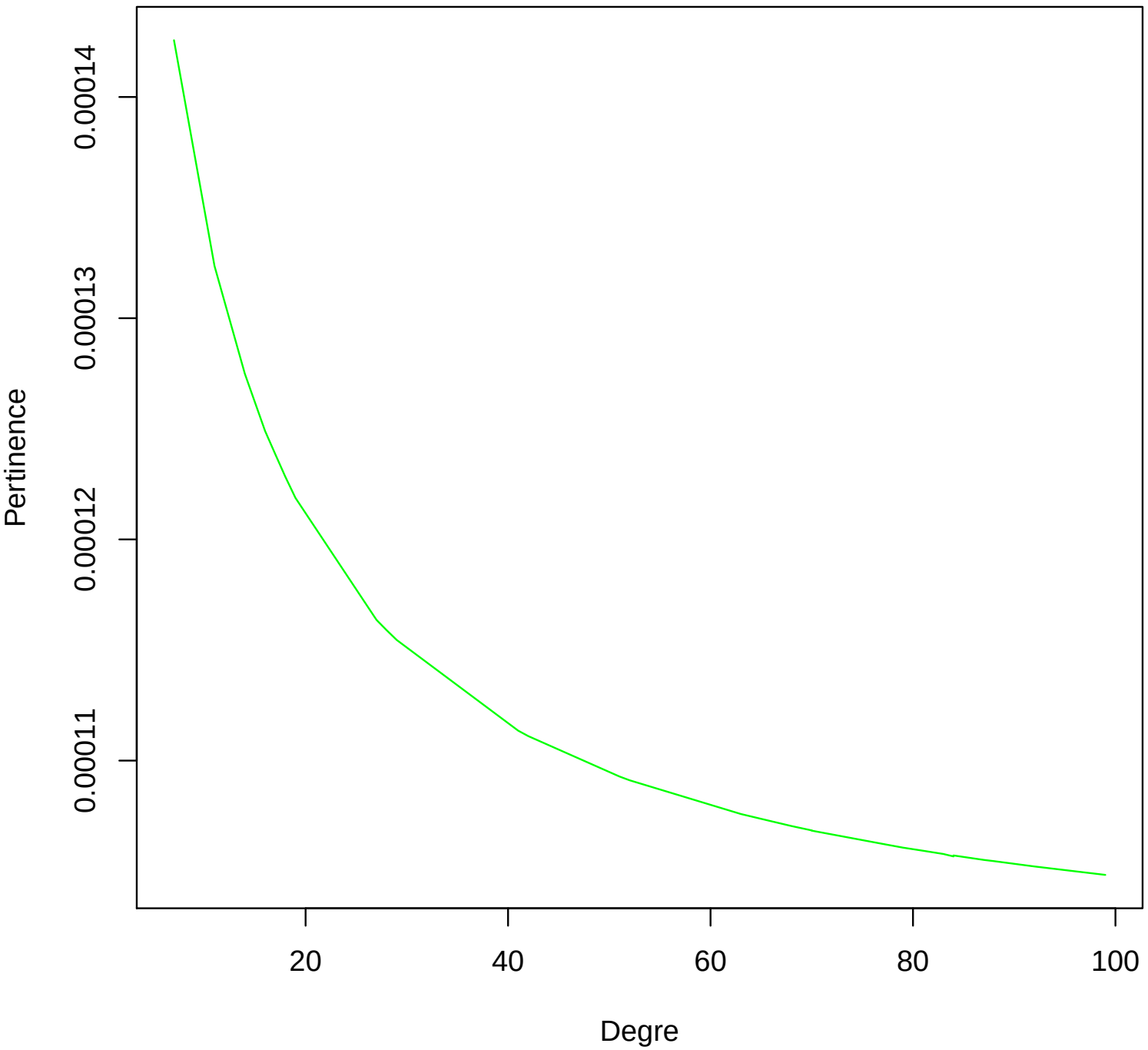
4.3 Conclusion sur l'impact d'un nombre aléatoires de structures

5 III. Expériences avec un graphe de degré aléatoire

Ici nous allons créer un graphe de 100 sommets de degré aléatoire. C'est à dire que chaque sommet sera relié au même nombre de sommets au sein du graphe.

5.1 Résultats des tests

	Pertinence initiale	Attaque avec un graphe de 100 sommets de degré aléatoire	Degré
	9,96E-05	1,0567E-04	84
	9,96E-05	1,0593E-04	81
	9,96E-05	1,0607E-04	79
	9,96E-05	1,0911E-04	52
	9,96E-05	1,1636E-04	27
	9,96E-05	1,2187E-04	19
	9,96E-05	1,0704E-04	68
	9,96E-05	1,3236E-04	11
	9,96E-05	1,0522E-04	92
	9,96E-05	1,0737E-04	65
	9,96E-05	1,0759E-04	63
	9,96E-05	1,0483E-04	99
	9,96E-05	1,2490E-04	16
	9,96E-05	1,1636E-04	27
	9,96E-05	1,1111E-04	42
	9,96E-05	1,1590E-04	28
	9,96E-05	1,2749E-04	14
	9,96E-05	1,1136E-04	41
	9,96E-05	1,2283E-04	18
	9,96E-05	1,0928E-04	51
	9,96E-05	1,0546E-04	88
	9,96E-05	1,0685E-04	70
	9,96E-05	1,1545E-04	29
	9,96E-05	1,0624E-04	77
	9,96E-05	1,0578E-04	83
	9,96E-05	1,0571E-04	84
	9,96E-05	1,0551E-04	87
	9,96E-05	1,4257E-04	7
	9,96E-05	1,2386E-04	17
	9,96E-05	1,0683E-04	70
Nb tirages	30	30	30
Espérance	9,96E-05	1,13E-04	
Ecart type	0	9,61535E-06	
Intervalle de confiance : 95% de la moyenne			
t-normale	1,645	1,645	
Erreur	0	2,88782E-06	


Pertinence
0.00014
0.00013
0.00012
0.00011
20
40
60
80
100
Degre

5.2 Analyses

On observe que plus le degré est fort, moins la pertinence de la cible est modifié. Ici l'espérance de la pertinence de la cible apres attaque est de 1.13e-04, ce qui est meilleur que la pertience de la cible lors des tests initiaux avec un graphe complet a 100 sommets (1.05e-04). Ainsi cette structure est plus eficace que le graphe complet. De plus cette structure a l'avantage d'etre moins detectable que l'anneau ou le graphe complet car le degré varie. En effet détecter un anneau ou un graphe complet est beaucoup plus aisé que de détecter un graphe de degré X.

6 IV. Expériences sur l'efficacité

6.1 Résultats des tests

Type de graphe attaquant	Pertinence de la cible	Nombre de sommets	Pertinence après attaque	Différence avec la pertinence initiale	% de modification	Efficacité
Anneau	9,96E-05	100	1,71E-04	7,17E-05	72,0%	7,17E-07
		250	2,79E-04	1,79E-04	179,9%	7,17E-07
		500	4,58E-04	3,58E-04	359,5%	7,16E-07
		1000	8,14E-04	7,15E-04	717,5%	7,15E-07
	7,54E-06	100	5,22E-05	4,46E-05	591,8%	4,46E-07
		250	1,19E-04	1,12E-04	1479,3%	4,46E-07
		500	2,31E-04	2,23E-04	2957,4%	4,46E-07
		1000	4,53E-04	4,46E-04	5910,4%	4,46E-07
	6,05E-07	100	4,52E-05	4,46E-05	7373,0%	4,46E-07
		250	1,12E-04	1,12E-04	18428,2%	4,46E-07
		500	2,24E-04	2,23E-04	36842,8%	4,46E-07
		1000	4,46E-04	4,46E-04	73630,6%	4,46E-07
Complet	9,96E-05	100	1,05E-04	5,17E-06	5,2%	5,17E-08
		250	1,05E-04	5,28E-06	5,3%	2,11E-08
		500	1,05E-04	5,23E-06	5,3%	1,05E-08
		1000	1,05E-04	5,05E-06	5,1%	5,05E-09
	7,54E-06	100	1,08E-05	3,23E-06	42,9%	3,23E-08
		250	1,09E-05	3,34E-06	44,2%	1,33E-08
		500	1,09E-05	3,36E-06	44,6%	6,72E-09
		1000	1,09E-05	3,36E-06	44,5%	3,36E-09
	6,05E-07	100	3,84E-06	3,24E-06	534,8%	3,24E-08
		250	3,95E-06	3,34E-06	552,2%	1,34E-08
		500	3,98E-06	3,38E-06	557,7%	6,75E-09
		1000	3,99E-06	3,39E-06	559,5%	3,39E-09
Arbre	9,96E-05	100	1,72E-04	7,22E-05	72,5%	7,22E-07
		250	2,79E-04	1,80E-04	180,4%	7,19E-07
		500	4,58E-04	3,59E-04	360,0%	7,17E-07
		1000	8,15E-04	7,15E-04	718,0%	7,15E-07
	7,54E-06	100	5,25E-05	4,49E-05	595,7%	4,49E-07
		250	1,19E-04	1,12E-04	1483,6%	4,47E-07
		500	2,31E-04	2,23E-04	2962,0%	4,47E-07
		1000	4,54E-04	4,46E-04	5915,2%	4,46E-07
	6,05E-07	100	4,55E-05	4,49E-05	7421,3%	4,49E-07
		250	1,12E-04	1,12E-04	18481,8%	4,47E-07
		500	2,24E-04	2,23E-04	36899,3%	4,47E-07
		1000	4,47E-04	4,46E-04	73689,8%	4,46E-07
Arbre à attaquant unique	9,96E-05	100	1,38E-04	3,88E-05	38,9%	3,88E-05
		250	1,79E-04	7,96E-05	79,9%	7,96E-05
		500	2,35E-04	1,36E-04	136,3%	1,36E-04
		1000	3,31E-04	2,31E-04	231,8%	2,31E-04
	7,54E-06	100	3,17E-05	2,41E-05	319,9%	2,41E-05
		250	5,71E-05	4,95E-05	657,2%	4,95E-05
		500	9,21E-05	8,45E-05	1121,2%	8,45E-05
		1000	1,51E-04	1,44E-04	1908,1%	1,44E-04
	6,05E-07	100	2,47E-05	2,41E-05	3985,3%	2,41E-05
		250	5,02E-05	4,96E-05	8187,8%	4,96E-05
		500	8,52E-05	8,45E-05	13969,5%	8,45E-05
		1000	1,44E-04	1,44E-04	23772,2%	1,44E-04

Ici nous avons testé chaque structure sur trois cibles différentes, tout en faisant varier le nombre de sommets par structure. Le pourcentage de modi-

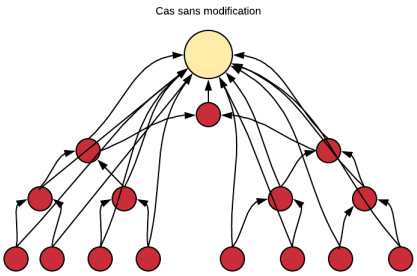
fication est calculé ainsi que l'efficacité calculé en divisant la différence par rapport a la pertinence initiale par le nombre de sommets ajoutés.

6.2 Analyses

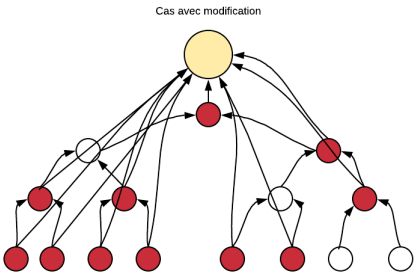
On observe que, toute pertinences confondus, la structure qui modifie le plus la cible est l'arbre, suivis de l'anneau, de l'arbre à attaquant unique et enfin du graphe complet. Cela confirme nos hypothèses des tests initiaux. La raison de cet efficacité est que la pertinence ne se disperse que tres peu dans ces structures, ce qui permet d'attaquer la cible efficacement. Par exemple dans un anneau chaque sommet recoit la pertinence d'un de ses voisins puis pointe sur la cible. Ainsi la pertinence est équitablement répartie et peut dispersé au sein de la structure. Pour l'arbre on observe un phénomène similaire car chaque sommet est relié à son sommet père et à la cible. On peut expliquer la légère différence entre arbre et anneau par le fait que plus on monte dans l'arbre, plus la pertinence est forte. Ainsi les attaques des sommets les plus haut dans l'arbre sont plus efficaces sur la cible.

On observe que, pour toute structures confondus, la différence avec la pertinence initiale est quasiment identique entre une cible moyenne (7.54e-06) et faible (6.05e-07). C'est à dire que on modifie autant la cible qu'elle soit de pertinence moyenne ou faible. Ainsi on peut en conclure que les cibles en dessous de Xe-06 sont toutes trés similaires à attaquer.

Pour l'anneau, le graphe complet et l'arbre on observe que la différence avec la pertinence initiale augmente de manière linéaire avec l'augmentation du nombre de sommets. par exemple pour l'arbre avec une cible forte on passe de 7.22e-5, 1.80e-4, 3.59e-4, 7.15e-4 pour 100, 250, 500 et 1000 sommets. Ce qui est presque parfaitement linéaire. Ce phénomène arrive quelquesoit la pertinence de la cible. Ainsi on peut en conclure que pour une attaque la plus efficace possible il faut augmenter le nombre de sommets au maximum. Cependant plus il y a de sommets, plus il est facile pour la cible de reconnaitre une attaque. Une stratégie raisonable serait d'utiliser un arbre de 500 sommets, ce qui modifie la pertinence de la cible de 360%, tout en ne représentant que 0.18% du graphe du web. De plus une structure en arbre peut être difficile à reconnaitre si l'attaquant décide de modifier légèrement sa structure. Par exemple enlever certains liens pointant vers la cible peuvent



donner cela :

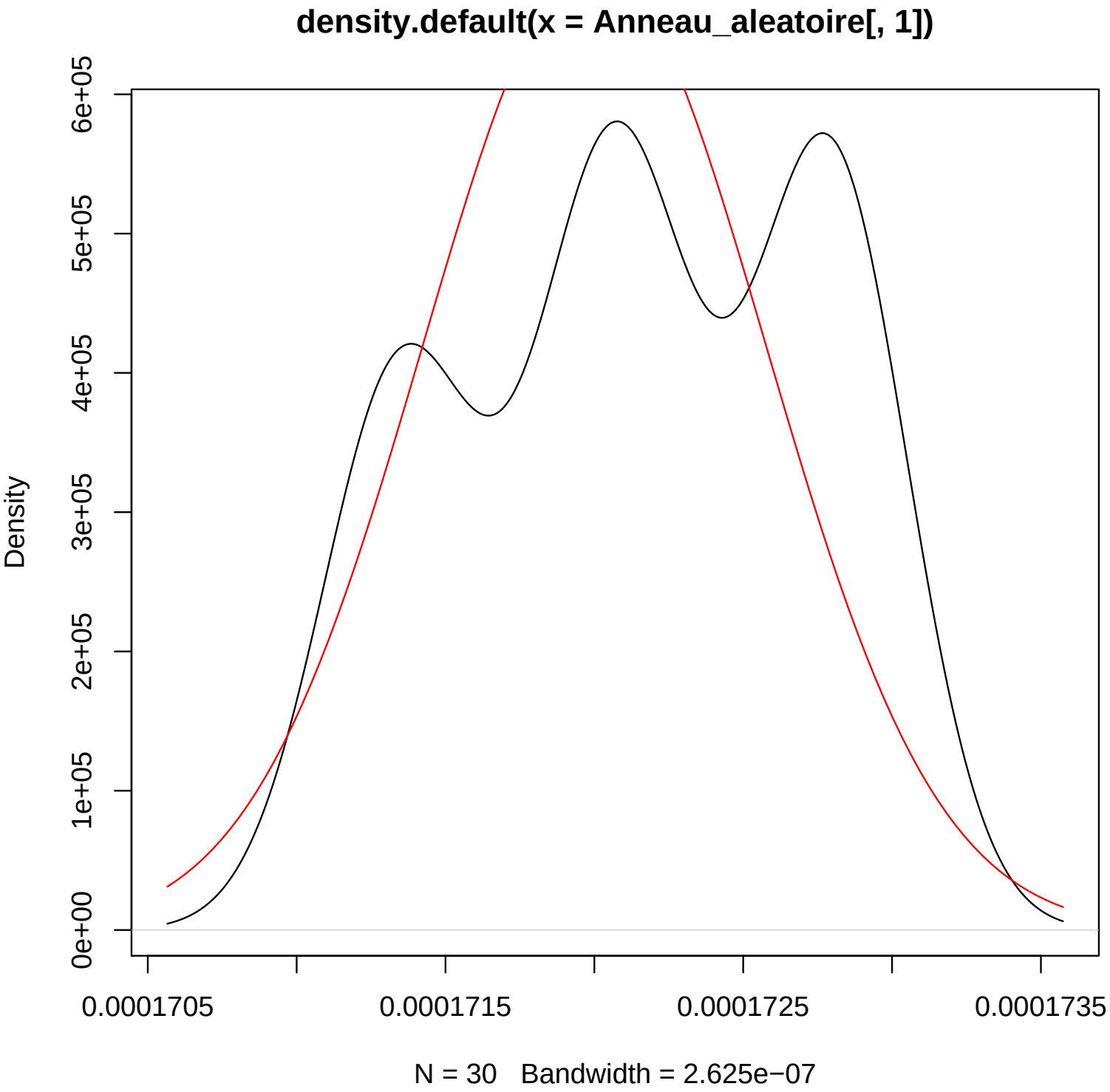


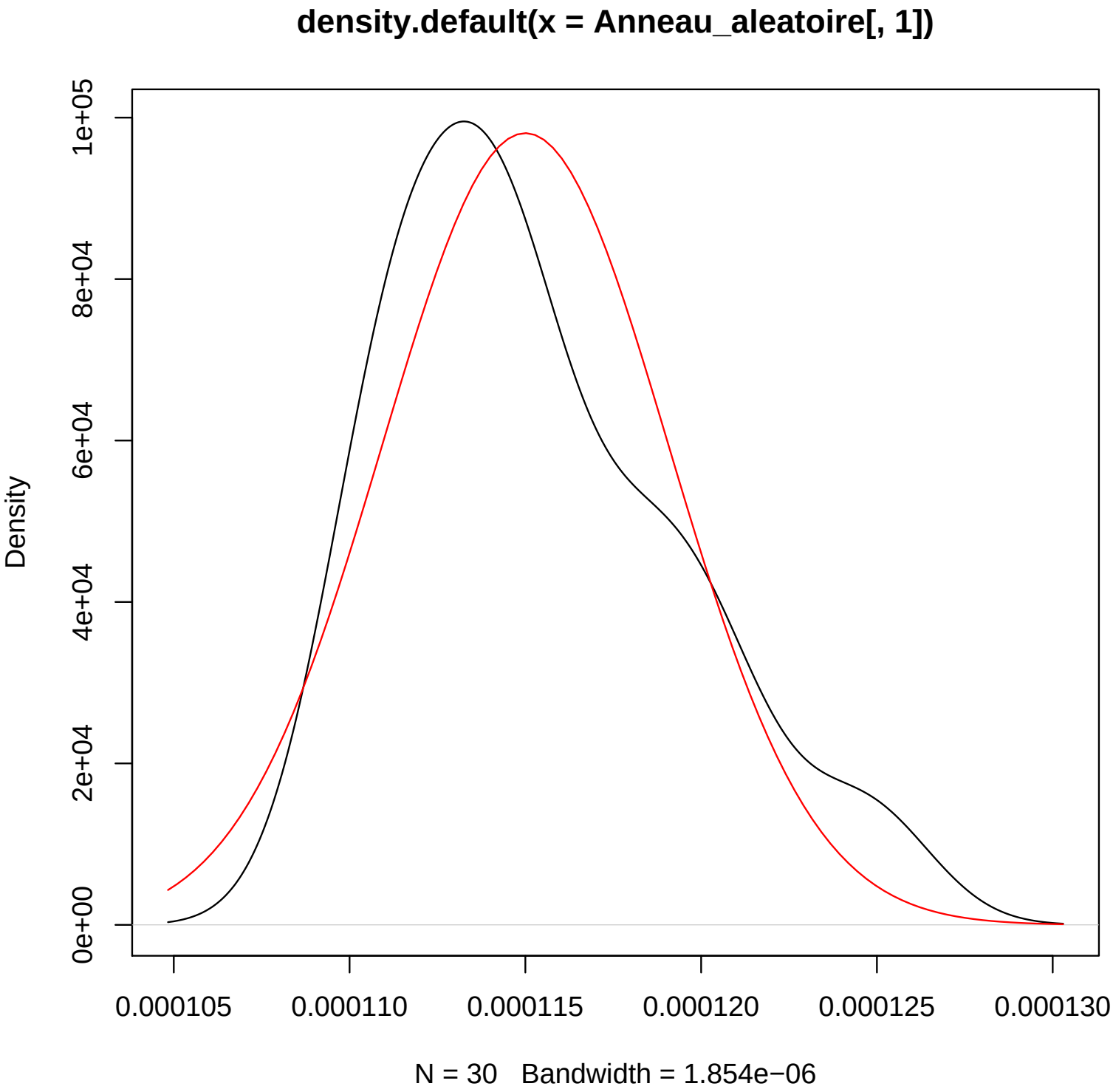
La cible ne peut voir que les pages qui pointe directement sur elle, c'est à dire les sommets en rouge. La seconde structure est légèrement moins efficace mais elle rend plus difficile la detection de l'arbre par la cible. En effet les sommets ne sont plus reliées ensemble en forme d'arbre du point de vu la cible. C'est donc une stratégie possible.

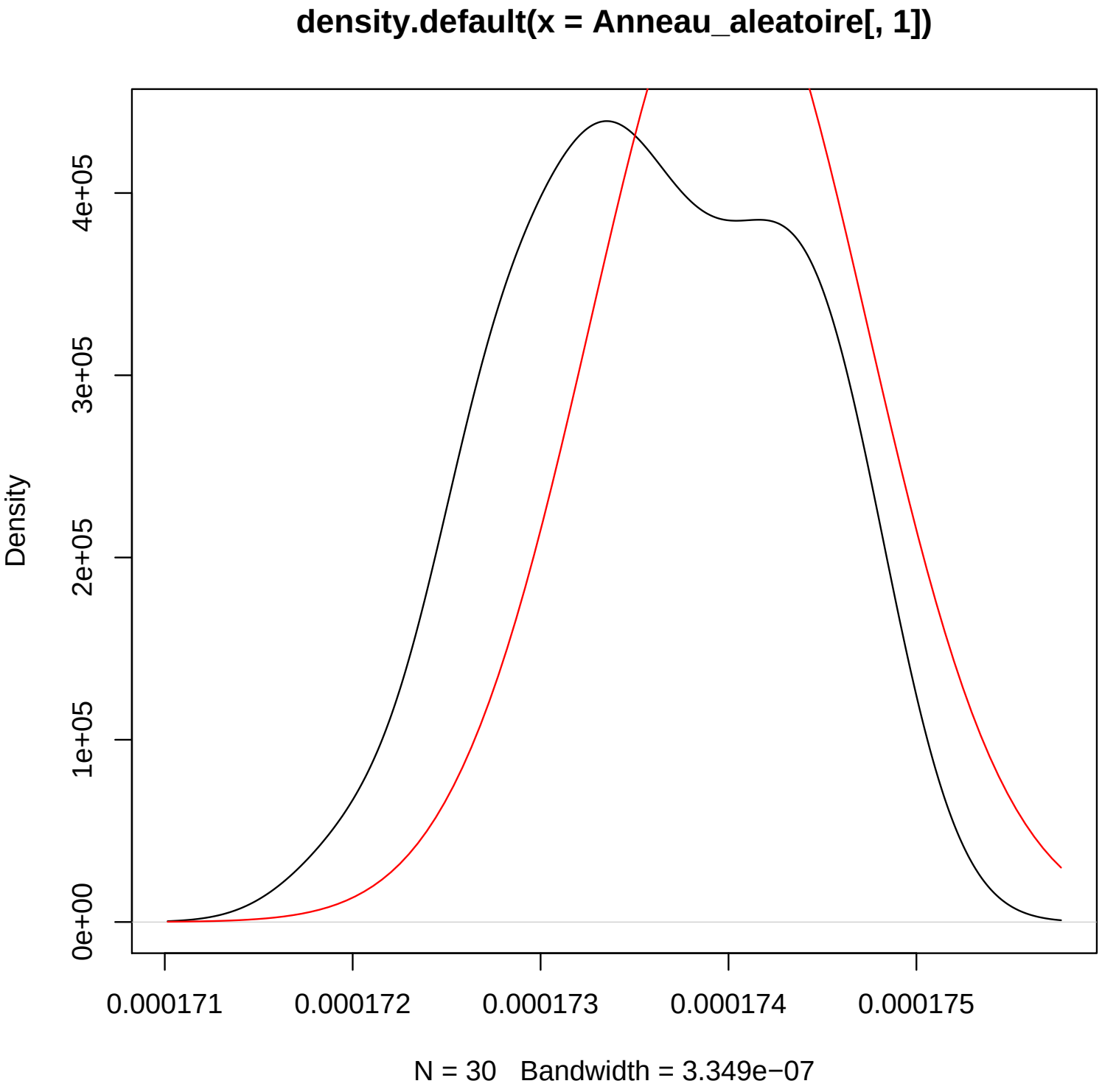
On observe que les valeurs d'efficacité les plus grandes sont pour l'arbre a attaquant unique. Cela s'explique par le fait que un seul sommet est relié a la cible dans cette structure. Cependant les pourcentages de modifications sont bien plus faible que pour l'arbre (38% contre 72% pour une cible forte à 100 sommets attaquants). Mais cette structure reste très utile pour des cibles moyennes ou faible. Ainsi une startégie interessante pour les cibles faibles serait d'utiliser un arbre a attaquant unique. Son avanatge majeur est qu'il est tres difficile a détecter car un seul sommet est directement relié a la cible.

7 Conclusions

8 Annexe


density.default(x = Anneau_aleatoire[, 1])
Density
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
6e+05
0.0001705
0.0001715
0.0001725
0.0001735
N = 30 Bandwidth = 2.625e−07


density.default(x = Anneau_aleatoire[, 1])
Density
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
0.000105
0.000110
0.000115
0.000120
0.000125
0.000130
N = 30 Bandwidth = 1.854e−06


density.default(x = Anneau_aleatoire[, 1])
Density
0e+00
1e+05
2e+05
3e+05
4e+05
0.000171
0.000172
0.000173
0.000174
0.000175
N = 30 Bandwidth = 3.349e−07