Graphes « petit monde », HITS, PageRank

Préambule

Le choix du langage de programmation est laissé à l'appréciation des étudiants. Un compte-rendu synthétique du TP devra être rendu pour le 25 janvier 2008.

1 Graphe aléatoire

Lois exponentielles

Nous avons vu en cours que les lois exponentielles se décrivent de la manière suivante, pour un γ fixé :

- cas discret : $P(K = k) = C_\gamma k^{-\gamma}, k = 1, 2, \ldots$
- cas réel densité de probabilité : $p(x) = D_\gamma x^{-\gamma}, x \geq 1$

où C_γ et D_γ sont des constantes de normalisation permettant de s'assurer que :

$$\sum_k P(K = k) = 1, \qquad \int_1^{+\infty} p(x)dx = 1.$$

1. Quelle est la valeur minimale que peut prendre γ ?
2. Donner une expression de C_γ. Donner la valeur de D_γ
3. Ecrire un programme qui permet de générer une suite de nombres entiers suivant une loi exponentielle, γ étant donné (i.e. paramètre de la fonction). On peut utiliser le principe de la roue biaisée.
4. Faire un graphe (en utilisant gnuplot ou un tableur) qui trace la distribution théorique des nombres générés et la distribution empirique, pour différentes longueurs de séquences générées.

Construction d'un graphe aléatoire : les riches deviennent plus riches

Voici une méthode simple de construction de graphe aléatoire de type « petit monde »[1]. La méthode est définie par 2 paramètres : M_0 et $m \leq M_0$. A chaque étape de l'algorithme un nouveau nœud et m nouveaux liens (non orientés) sont créés : ces nouveaux liens viennent relier le nouveau nœud à m nœuds parmi ceux qui existaient à l'étape précédente ou bien à lui-même (petite modification par rapport à la méthode initiale). Le choix des nœuds avec lesquels le nouveau nœud v va se lier se fait en fonction de la probabilité $k_w/(\sum_r k_r - 1)$ pour les nœuds existants et $1/(\sum_r k_r - 1)$ pour v, où k_w désigne le degré de w. Il peut y avoir des arêtes multiples et des boucles.

1. L'algorithme tel qu'il est décrit est incomplet, notamment pour son initialisation : expliquez ce qu'il se passe si l'algorithme décrit tel quel n'est pas modifié. On propose comme modification de supposer une distribution de probabilité uniforme (pour la création d'arête) pour les points de M_0 tant qu'il existe encore un nœud ayant un degré nul.
2. Programmer cet algorithme.
3. Générer des graphes aléatoires et estimer le coefficient γ de la loi exponentielle que suit le degré des sommets.
4. Programmer une méthode qui permet d'orienter les arêtes de façon aléatoire.
5. Au lieu d'ajouter un nœud à chaque étape on peut choisir, pour p et q donnés de
   - avec probabilité p : créer un nouveau nœud et m arêtes comme décrit précédemment,
   - avec probabilité q : ajouter m arêtes (sans créer de nœud) en utilisant la distribution préférentielle décrite ci-dessus,
   - avec probabilité $1-p-q$: rerouter m arêtes toujours en utilisant la distribution préférentielle décrite ci-dessus.

   Implémenter cette stratégie et estimer les caractéristiques des graphes obtenus en fonction de M_0, m, p et q.

2 HITS et PageRank

1. Programmer les algorithmes HITS et PageRank.
2. Les faire fonctionner sur le graphe vu en cours.
3. Simuler les marches aléatoires en fonction des matrices stochastiques primitives vues en cours et estimer empiriquement les probabilités PageRank en simulant des marches de différentes longueurs.
4. Supprimer et rerouter aléatoirement des liens du graphe vu en cours et regarder l'évolution des scores PageRank.
5. Faire le même type d'expériences avec des graphes aléatoires plus grands. Estimer la vitesse de convergence de l'algorithme.

Références

[1] Albert R. Barabasi. Emergence of scaling in random networks -. *Science*, pages 509–512, 1999.