# Projet INF442
# PageRank*

Difficulté $*$

Pierre-Louis Poirion
poirion@lix.polytechnique.fr

X2014
Session 2016

## 1 Principe de PageRank

On considère que le web est une collection de $N \in \mathbb{N}$ pages, avec $N$ très grand. La plupart de ces pages incluent des liens vers d'autres pages, on dit qu'elles pointent vers ces autres pages. L'idée de base utilisée pour classer ces pages consiste à considérer que plus une page est pointées, plus elle a de chances d'être fiable et intéressante pour l'utilisateur. Le "PageRank" d'une page est une valeur qui permet de classer les pages web par ordre de pertinence.

Plus formellement, on représente le web sous la forme d'un graphe orienté $G = (V, E)$ où $V = \{1, ..., N\}$ et $(i, j) \in E$ si la page $j$ pointe sur la page $i$. Supposons que l'utilisateur choisisse chaque lien indépendamment des pages précédemment visitées. Le déplacement de l'utilisateur sur le web est ainsi un processus de Markov. Le pagerank correspond alors à la probabilité stationnaire $r \in \mathbb{R}^N$d'une chaîne de Markov.

Soit $C \in M_N(\mathbb{R})$ la matrice d'adjacence du graphe. Pour tout $j \in V$, soit $N_j = \sum_{k=1}^{N} C_{kj}$ le nombre total de liens sortant de la page $j$. On peut alors construire la matrice $Q \in M_N(\mathbb{R})$ où
$$Q_{ij} = \begin{cases} \frac{C_{ij}}{N_j} & \text{si } N_j > 0 \\ 0 & \text{sinon} \end{cases}$$
Le pagerank est donc le vecteur $r \in \mathbb{R}^N$ vérifiant $r = Qr$. On cherche donc le vecteur propre associé à la valeur propre 1 de la matrice $Q$.

**Question 1** Écrire la matrice d'adjacence associée aux exemples des figures 1 et 2.

* http://en.wikipedia.org/wiki/PageRank

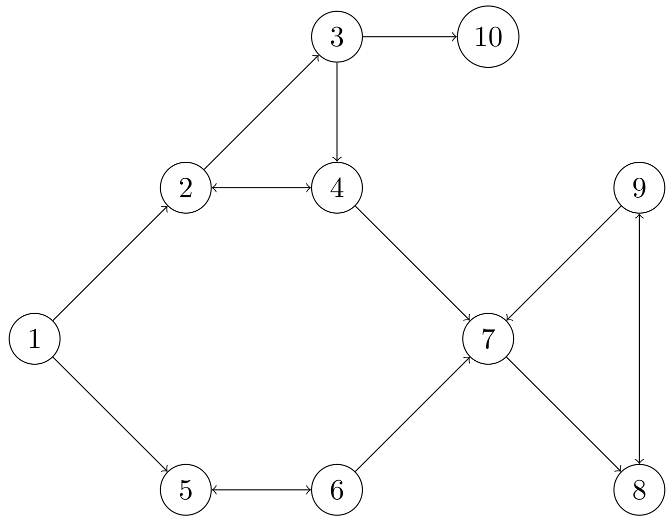


Figure 1: Exemple 1

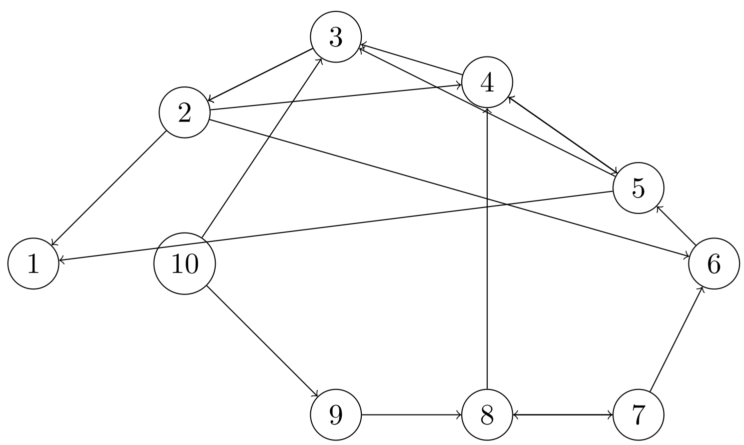


Figure 2: Exemple 2

## 2 Calcul de $r$

La matrice $Q$ peut ne pas admettre 1 comme valeur propre. Pour contourner cela, on introduit la matrice $P$ définie par :

$$P = Q + \frac{1}{N} e d^\top,$$

où $e = (1, ..., 1)$ et $d_j = \begin{cases} 1 & \text{si } N_j = 0 \\ 0 & \text{sinon} \end{cases}$

On prouve facilement que $P$ admet toujours 1 comme valeur propre.

On modifie à nouveau $P$ pour que 1 soit valeur propre simple. Soit $0 < \alpha < 1$, on introduit la matrice $P_\alpha$ par

$$P_\alpha = \alpha P + (1-\alpha)\frac{1}{N}ee^\top$$

**Question 2** Écrire une fonction prenant en paramètre la matrice d'adjacence d'un graphe et un réel $\alpha$, et qui retourne la matrice $P_\alpha$.

Au final l'algorithme *PageRank* revient à déterminer, pour $0 < \alpha < 1$ fixé, le vecteur $r_\alpha \in \mathbb{R}^N$ associé à la valeur propre 1, qui satisfait $1 = \rho(P_\alpha)$ ($\rho(P_\alpha)$ indique la plus grande valeur propre de $P_\alpha$). En pratique, on calcule $r_\alpha$ par l'algorithme des puissances itérées :
Idée : à l'étape $k$ ayant un vecteur $r_\alpha^k$ de norme 1, on calcule $P_\alpha r_\alpha^k$ qu'on normalise ensuite.

**Algorithm 1** Méthode des puissances itérées

Entrée : Matrice $P_\alpha$, marge d'erreur $\epsilon$
vecteur initial $r$ non nul de norme 1.
**while** $\|r^{(t)} - r^{(t+1)}\|_1 \leq \epsilon$ **do**
    $r^{t+1} = P_\alpha r^t$
    $r^{t+1} = \frac{r^{t+1}}{\|r^{t+1}\|_1}$
**end while**
Return $r$

où $\|x\|_1 = \sum |x|^i$

**Question 3** Écrire une fonction qui exécute en parallèle la méthode des puissances itérées.

**Question 4** Trouver le Pagerank du graphe de l'exemple pour différentes valeurs de *alpha*. Que constatez vous ?

En pratique la matrice d'adjacence des graphes est creuse, en revanche ce n'est pas le cas, en générale, de la matrice $P_\alpha$. Aussi, il est en pratique hors de question d'assembler la matrice$P_\alpha$. On peut montrer cependant que pour tout vecteur $z \in \mathbb{R}^N$, on a :

$$P_\alpha z = \alpha Q z + \frac{1}{N}\left(\alpha d.z + (1-\alpha)e.z\right)e$$

On s'est donc ramené à un produit matrice vecteur, ou la matrice ($Q$) est, cette fois ci, creuse.

**Question 4** Écrire une fonction qui calcule en parallèle un produit matrice creuse, vecteur

**Question 5** Modifiez l'algorithme PageRank en conséquence.

**Question 6** Testez votre algorithme sur des graphes creux de grandes tailles ($N > 100$) générés aléatoirement. On pourra chercher à faire varier la densité, $p$, (c-à-d la probabilité $p$ que l'arc $(i, j)$ appartienne à $E$) du graphe et à comparer la vitesse des deux algorithmes de la question 4 et de la question 6.