

## SUJET D'ANALYSE

**Etude du fonctionnement de l'algorithme de classement des pages Web utilisé par le moteur de recherche Google**

Présenté par :  
**Kouamé Gérard Kra**  
**Seyed Hossein Seyed Nahrmiani**  
**Koffi Roland Wotobe**

Encadrant :  
**M Rubenthaler**

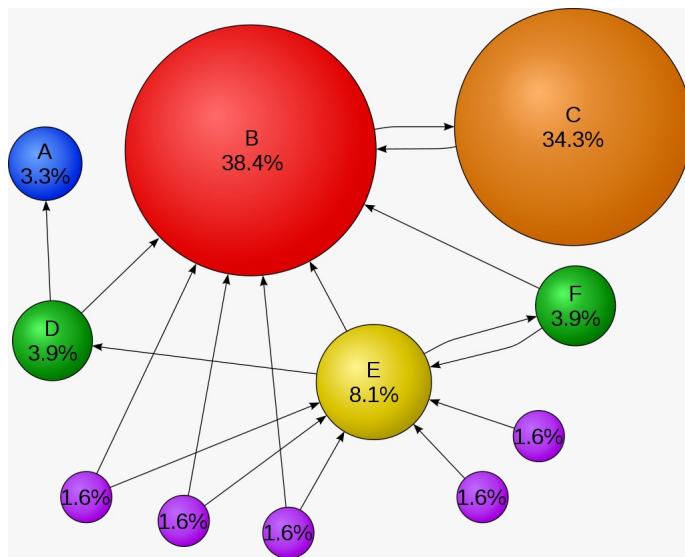


FIGURE 1 – Une simple illustration de l'algorithme Pagerank. Le pourcentage indique l'importance perçue, et les flèches représentent les hyperliens.

Source :<https://en.wikipedia.org/wiki/PageRank>

# Table des matières

1	INTRODUCTION	2
2	DEFINITIONS DES MOTS-CLES	2
3	ORIGINE DE PAGERANK	3
3.1	Approche récursive	3
3.2	Promenade aléatoire	3
3.3	Loi de transition	5
4	PAGERANK	5
4.1	Fonctionnement de PageRank	6
5	MISE EN ŒUVRE	7
6	Conclusion	9

## 1 INTRODUCTION

Dans le vaste espace numérique qu'est l'Internet, la quête d'informations commence souvent par une simple recherche. Etant donné qu'il y a une multitude d'informations sur le web, le besoin de gagner du temps et la pertinence des informations deviennent alors nécessaires pour chaque requête. C'est l'une des préoccupations fondamentales qui incita Larry Page et Sergueï Brin à concevoir le moteur de recherche Google. Depuis sa conception en 1998, Google a connu plusieurs améliorations et cela fait de lui aujourd'hui le moteur de recherche le plus utilisé au plan mondial. Cependant, la plupart des réformes de Google demeurent jusqu'à présent des secrets. Par contre, selon l'un des articles publiés par les fondateurs, le pilier de son succès est une judicieuse modélisation mathématique que nous retracsons ici.

## 2 DEFINITIONS DES MOTS-CLES

Dans cette section nous définissons quelques mots essentiels autour desquels notre présentation va s'articuler :

- Classement des pages Web : Le processus d'ordonnancement des résultats de recherche en fonction de leur pertinence et de leur importance.
- Importance d'une page web : La mesure de la pertinence et de l'autorité d'une page Web dans le contexte d'une recherche en ligne.
- Graphe du Web : Représentation graphique des pages Web interconnectées, où les différents noeuds représentent les pages web et les flèches entre les pages représentent les liens entre ces dernières.
- Algorithmes de recherche : De façon générale un algorithme est une suite finie et non ambiguë d'instructions et d'opérations permettant de résoudre un problème concret. En particulier un algorithme de recherche est une méthode informatique utilisée pour trouver des informations précises dans un ensemble de données.

Dans la suite de cette présentation, nous utiliserons le graphe du web ci-dessous.

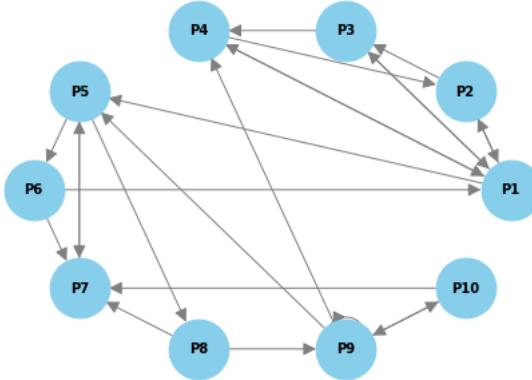


FIGURE 2 – Exemple de graphe du web

### 3 ORIGINE DE PAGERANK

Le modèle PageRank est un modèle beaucoup plus raffiné utilisé par Google pour la classification des pages web. Google fournit des résultats de recherche qui correspondent aux termes de recherche de l'utilisateur. En coulisse, Google maintient un classement parmi les sites web pour s'assurer que les sites "meilleurs" ou "plus importants" apparaissent tôt dans les résultats de recherche. Au lieu de résoudre le problème dans son ensemble en une seule fois, ce classement est d'abord établi de manière globale (indépendamment des termes de recherche), et seulement plus tard, les sites web correspondant à la requête de recherche sont triés selon un certain classement. Dans cette section, nous nous concentrerons sur la partie classement, nous notons les pages web par  $P_1, P_2, P_3, \dots, P_n$  et nous écrivons  $j \rightarrow i$  si la page  $P_j$  cite la page  $P_i$ .

#### 3.1 Approche récursive

L'approche récursive consiste à dire qu'une page  $P_i$  paraît importante si beaucoup de pages importantes la citent. Ceci nous mène à définir l'importance  $m_i$  de manière récursive comme suit :

$$m_i = \sum_{j \rightarrow i} \frac{1}{l_j} m_j \quad (1)$$

où  $l_j$  dénote le nombre de liens émis.

Remarque :

- Ici le poids du vote  $j \rightarrow i$  est proportionnel au poids  $m_j$  de la page émettrice.
- (1) est un système de  $n$  équations linéaires à  $n$  inconnues. Dans notre exemple, où  $n=10$ , il est déjà pénible à résoudre à la main, mais encore facile sur ordinateur. Pour les graphes beaucoup plus grands nous aurons besoin de méthodes spécialisées.

#### 3.2 Promenade aléatoire

Avant de tenter de résoudre l'équation (1), essayons d'en développer une intuition. Pour ceci imaginons un surfeur aléatoire qui se balade sur internet en cliquant sur les liens au hasard. Comment évolue sa position ? À titre d'exemple, supposons que notre surfeur démarre au temps  $t = 0$  sur la page  $P_1$ . De là les liens pointent vers  $P_2, P_3, P_4, P_5$  donc au temps  $t = 1$  le surfeur se retrouvent sur chacune de ces pages avec probabilité  $\frac{1}{4}$ . Le programme ci-dessous nous donne les probabilités à  $10^{-3}$  près) :

## Programme Python :

```
import numpy as np
import matplotlib.pyplot as plt
def pi(P):
    n = P.shape[0]
    A = np.transpose(P) - np.eye(n)
    A[0,:] = np.ones(n)
    b = np.zeros(n)
    b[0] = 1
    return np.linalg.solve(A, b)

def distri_stat(P, mu, epsilon, n):
    p_i = pi(P)
    current = mu
    print("t=0 ===> ", current)
    for i in range(n):
        new_distri = np.dot(current,P)
        diff = np.linalg.norm(p_i-new_distri,1)
        conv.append(diff)
        ind.append(i)
        print("t="+str(i+1) + " ===> ", new_distri)
        print("-----")
        if np.linalg.norm(new_distri-p_i,1) < epsilon:
            break
        current = new_distri

    return current

P = np.array([[0, 1/4, 1/4, 1/4, 1/4, 0, 0, 0, 0, 0], 
[1/2, 0, 1/2, 0, 0, 0, 0, 0, 0, 0], 
[1/2, 0, 0, 1/2, 0, 0, 0, 0, 0, 0], 
[1/2, 1/2, 0, 0, 0, 0, 0, 0, 0, 0], 
[0, 0, 0, 0, 1/3, 1/3, 1/3, 0, 0, 0], 
[1/2, 0, 0, 0, 0, 0, 1/2, 0, 0, 0], 
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0], 
[0, 0, 0, 0, 0, 1/2, 0, 1/2, 0], 
[0, 0, 0, 1/4, 1/4, 0, 0, 0, 1/4, 1/4], 
[0, 0, 0, 0, 0, 1/2, 0, 1/2, 0]]) 

mu = np.array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0])

X = distri_stat(P, mu, 5e-3, 30)
```

---

	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	$P_7$	$P_8$	$P_9$	$P_{10}$
$t = 0$	1.00	.000	.000	.000	.000	.000	.000	.000	.000	.000
$t = 1$	.000	.25	.25	.25	.25	.000	.000	.000	.000	.000
$t = 2$	.375	.125	.125	.125	.000	.083	.083	.083	.000	.000
$t = 3$	.229	.156	.156	.156	.177	.000	.083	.000	.042	.000
$t = 4$	.234	.135	.135	.146	.151	.059	.059	.059	.01	.01
$t = 5$	.238	.132	.126	.129	.120	.05	.115	.05	.037	.003
$\vdots$										
$t = 25$	.182	.098	.095	.106	.193	.064	.135	.064	.051	.013
$t = 26$	.181	.098	.095	.106	.193	.064	.135	.064	.051	.013

Remarquons que selon la position initiale du surfeur, le nombre d’itérations peut varier mais tout en gardant la distribution stationnaire. C’est le cas de l’exemple ci-dessous où le surfeur commence initialement sur la page  $P_5$ . De là les liens pointent vers  $P_6$ ,  $P_7$ ,  $P_8$  donc au temps  $t = 1$  le surfeur se retrouvent sur chacune de ces pages avec probabilité  $\frac{1}{3}$ . En adaptant le programme 1 ci-dessus, on obtient également les probabilités suivantes à  $10^{-3}$  près :

	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	$P_7$	$P_8$	$P_9$	$P_{10}$
$t = 0$	.000	.000	.000	.000	1.00	.000	.000	.000	.000	.000
$t = 1$	.000	.000	.000	.000	.000	.333	.333	.333	.000	.000
$t = 2$	.167	.000	.000	.000	.333	.000	.333	.000	.167	.000
$t = 3$	.000	.042	.042	.083	.417	.111	.111	.111	.042	.042
$t = 4$	.139	.042	.021	.031	.122	.139	.271	.139	.089	.01
$t = 5$	.116	.05	.056	.067	.327	.041	.185	.041	.096	.022
$\vdots$										
$t = 27$	.180	.097	.093	.105	.194	.065	.136	.065	.052	.013
$t = 28$	.180	.097	.093	.105	.194	.065	.136	.065	.052	.013

Le modèle du surfeur aléatoire peut sembler étonnant, mais en absence d’information plus précise, le recours aux considérations probabilistes se révèle souvent très utile !

### 3.3 Loi de transition

Comment formaliser la diffusion illustrée ci-dessus ? Supposons qu’au temps  $t$  notre surfeur aléatoire se trouve sur la page  $P_j$  avec une probabilité  $p_j$ . La probabilité de partir de  $P_j$  et de suivre le lien  $j \rightarrow i$  est alors  $\frac{1}{l_j}m_j$ . La probabilité d’arriver au temps  $t + 1$  sur la page  $P_i$  est donc

$$p'_i = \sum_{j \rightarrow i} \frac{1}{l_j} m_j \quad (2)$$

Étant donnée la distribution initiale  $p$ , la loi de transition (2) définit la distribution suivante  $p'$  telle que  $p' = T(p)$ . C’est ainsi que l’on obtient la ligne  $t + 1$  à partir de la ligne  $t$  dans nos exemples. (En théorie des probabilités, ceci est appelé une chaîne de Markov.) La mesure stationnaire est caractérisée par l’équation d’équilibre  $m = T(m)$ , qui correspond justement à notre équation de départ (1).

## 4 PAGERANK

Le PageRank ou PR est un algorithme d’analyse de liens concourant au système de classement des pages Web utilisé par le moteur de recherche Google. Il mesure quantitativement la popularité d’une page web. Le PageRank n’est qu’un indicateur parmi d’autres dans l’algorithme qui permet de classer les pages du Web dans les résultats de recherche de Google.

## 4.1 Fonctionnement de PageRank

L'idée principale derrière PageRank est de mesurer l'importance ou la popularité d'une page en analysant les liens entrants et sortants de cette dernière. Pour ce faire, PageRank se base sur quelques concepts que nous définissons ci-dessous.

- **La notion de vote** : PageRank considère un lien d'une page I vers une page J comme un vote de la page I pour la page J. Plus il y a de liens pointant vers une page, plus cette page est considérée comme importante ou influente.
- **La pondération des votes** : Tous les votes ne sont pas égaux. L'importance d'une page est déterminée par le nombre et la qualité des pages qui pointent vers elle. Si une page importante (avec un score élevé elle-même) pointe vers une autre page, ce vote sera plus significatif qu'un lien provenant d'une page moins importante.
- **L'algorithme itératif** : PageRank fonctionne de manière itérative. Au départ, chaque page se voit attribuer un score égal. Ensuite, ces scores sont mis à jour itérativement en fonction des votes des pages qui y pointent et de leur propre importance. Ce processus se répète jusqu'à ce que les scores convergent vers une valeur stable (cette valeur stable est en fait la distribution stationnaire de la chaîne de Markov associée à notre graphe du web ci-dessus).
- **La matrice de transition** : On peut représenter les liens entre les pages sous forme d'une matrice de transition, où chaque élément de la matrice représente la probabilité de passer d'une page à une autre en suivant les liens. En appliquant des calculs matriciels itératifs, on peut calculer les scores de PageRank pour chaque page.
- **Damping factor et surfer aléatoire** : PageRank utilise également un facteur d'amortissement pour modéliser le comportement des utilisateurs qui pourraient simplement naviguer de manière aléatoire sans suivre de liens. Cela permet d'éviter les boucles infinies et de garantir la convergence de l'algorithme. (Dans notre graphe du web on peut avoir des sites qui ne pointent vers aucun autre site. Le damping factor permet donc de relier tout notre graphe malgré l'existence de tels sites web ainsi on est toujours sûr de pouvoir se déplacer sur le graphe)

Le calcul des scores PageRank pour notre graphe du web peut être représenté sous forme matricielle.

$$v = M \times v \quad (3)$$

C'est une équation aux vecteurs propres où  $v$  est le vecteur contenant les scores PageRank des différentes pages du graphe et  $M$  la matrice de transition associée au graphe du web. L'un des inconvénients du modèle récursif est que l'on peut tomber dans un trou noir. Dans notre contexte un trou noir est une page web qui ne pointe vers aucune autre. Ainsi lorsqu'on se déplace sur le graphe du web quand on rencontre un trou noir on y reste indéfiniment ce qui ne donne pas des résultats satisfaisants. Pour palier à ce problème Google utilise le modèle suivant :

- avec une probabilité fixée  $c$  le surfeur abandonne sa page actuelle  $P_j$  et recommence sur une des  $n$  pages du web choisie de manière équiprobable ;
- sinon, avec probabilité  $1 - c$ , le surfeur suit un des liens de la page  $P_j$ , choisi de manière équiprobable.

Cette astuce du *damping factor* évite de se faire piéger par une page sans issue et garantit d'arriver n'importe où dans le graphe. Dans ce modèle la transition est donnée par :

$$p'_i = \frac{c}{n} + \sum_{j \rightarrow i} \frac{1 - c}{l_j} \times p_j \quad (4)$$

## 5 MISE EN ŒUVRE

Dans cette partie nous allons mettre en oeuvre l'algorithme sur le graphe qui nous sert d'exemple. Nous effectuerons donc le calcul des scores PageRank et nous parlerons aussi de la convergence de l'algorithme. Le programme ci-dessous nous donne les probabilités à  $10^{-3}$  près) et le graphe associé :

---

Programme Python :

```
import numpy as np
import matplotlib.pyplot as plt

def distri_stat(P, mu, epsilon, n, damping_factor):
    current = mu
    conv = []
    ind = []
    print(p_i)
    for i in range(n):
        new_distri = (1-damping_factor)*np.dot(current, P) + damping_factor/10
        diff = np.linalg.norm(new_distri - current, 1)
        conv.append(diff)
        ind.append(i)
        print("t=" + str(i + 1) + " ==> ", new_distri)
        print("-----")
        if diff < epsilon:
            break
        current = new_distri
    return current, conv, ind

def plot_convergence(mu, ax, damping_factor):
    X, C, I = distri_stat(P, mu, 1e-3, 30, damping_factor)
    F = [(1 - damping_factor)**i for i in I]
    ax.plot(I, C, 'o', label='Ecart entre les distributions')
    ax.plot(I, F, linestyle='-', color='red', label='(1 - c)^n')
    ax.set_xlabel('Iterations')
    ax.set_ylabel('Ecart entre les distributions')
    ax.set_title(f'Convergence de l\'algorithme avec damping factor={damping_factor}')
    ax.grid(True)
    ax.legend()

P = np.array([[0, 1/4, 1/4, 1/4, 1/4, 0, 0, 0, 0, 0],  
             [1/2, 0, 1/2, 0, 0, 0, 0, 0, 0, 0],  
             [1/2, 0, 0, 1/2, 0, 0, 0, 0, 0, 0],  
             [1/2, 1/2, 0, 0, 0, 0, 0, 0, 0, 0],  
             [0, 0, 0, 0, 1/3, 1/3, 1/3, 0, 0, 0],  
             [1/2, 0, 0, 0, 0, 0, 1/2, 0, 0, 0],  
             [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],  
             [0, 0, 0, 0, 0, 1/2, 0, 1/2, 0, 0],  
             [0, 0, 0, 1/4, 1/4, 0, 0, 0, 1/4, 1/4],
```

```
[0, 0, 0, 0, 0, 0, 1/2, 0, 1/2, 0]])
```

```
mu1 = np.array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0])
mu2 = np.array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0])
```

```
fig, axs = plt.subplots(1, 2, figsize=(12, 4))
```

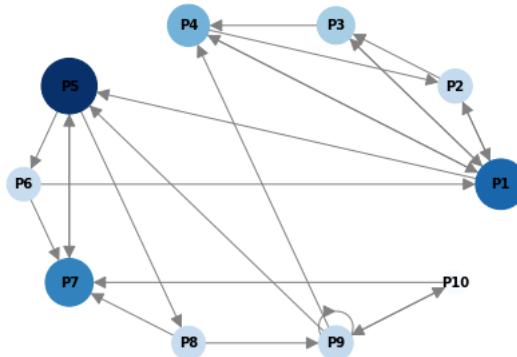
```
plot_convergence(mu1, axs[0], damping_factor=0.15)
axs[0].set_title('Initial Distribution: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]')
```

```
plot_convergence(mu2, axs[1], damping_factor=0.15)
axs[1].set_title('Initial Distribution: [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]')
```

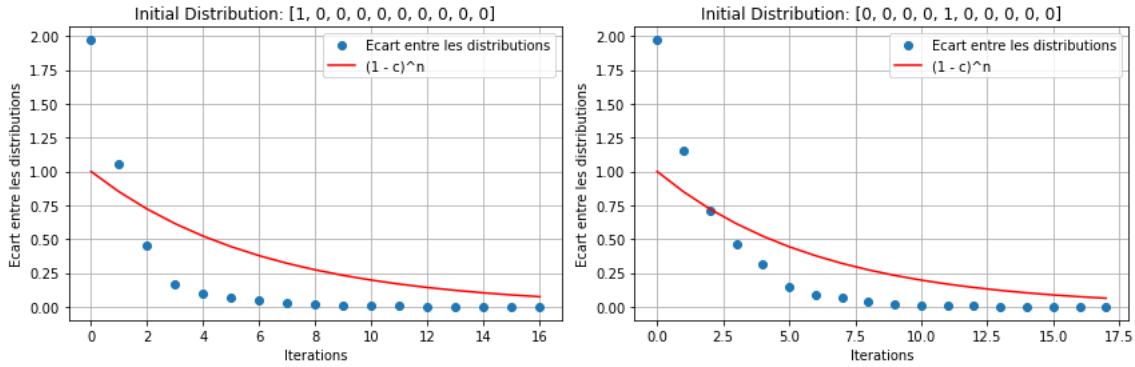
```
plt.tight_layout()
plt.show()
```

---

	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	$P_7$	$P_8$	$P_9$	$P_{10}$
$t = 0$	1.00	.000	.000	.000	.000	.000	.000	.000	.000	.000
$t = 1$	.015	.228	.228	.228	.228	.015	.015	.015	.015	.015
$t = 2$	.311	.115	.115	.118	.034	.079	.099	.079	.031	.018
$t = 3$	.197	.131	.130	.137	.172	.025	.1	.025	.063	.022
$t = 4$	.195	.115	.113	.125	.155	.064	.094	.064	.048	.028
$t = 5$	.192	.110	.105	.114	.146	.059	.125	.059	.064	.025
$\vdots$										
$t = 17$	.165	.094	.090	.103	.180	.066	.135	.066	.071	.03
$t = 18$	.165	.094	.090	.103	.180	.066	.135	.066	.071	.03



Ainsi nous obtenons les graphes suivants.



Nous remarquons que l'évolution de l'écart entre la distribution stationnaire et les distributions de probabilités au fil du temps décroît progressivement pour tendre vers 0. Cette évolution est équivalente à celle de la fonction  $(1 - c)^n$ . Cependant la fonction  $(1 - c)^n$  décroît plus vite que l'écart entre les distributions. C'est ce qui nous amène au théorème suivant.

#### Théorème du point fixe :

Considérons un graphe fini quelconque et fixons le paramètre  $c$  tel que  $0 < c \leq 1$ . Alors, l'équation (4) admet une unique solution vérifiant  $m_1 + \dots + m_n = 1$ . Dans cette solution  $m_1, \dots, m_n$  sont tous strictement positifs.

Pour toute distribution de probabilité initiale sur le graphe, le processus de diffusion (4) converge vers cette unique mesure stationnaire  $m$ . La convergence est au moins aussi rapide que celle de la suite géométrique  $(1 - c)^n$  vers 0.

Le théorème du point fixe assure donc que cette équation admet une unique solution, et justifie l'algorithme itératif (4) pour l'approcher.

## 6 Conclusion

Pour être utile, un moteur de recherche doit non seulement énumérer les résultats d'une requête mais les classer par ordre d'importance. Or, estimer la pertinence des pages web est un profond défi de modélisation. L'algorithme de PageRank, a révolutionné le domaine de la recherche sur Internet en fournissant une méthode efficace et pertinente pour évaluer la pertinence des pages web. En se basant sur le principe de la notoriété des pages liées, le PageRank a contribué à améliorer la qualité des résultats de recherche en classant les pages en fonction de leur importance dans le réseau. Son utilisation par le moteur de recherche Google a été cruciale pour améliorer la précision des résultats et a été l'un des facteurs clés du succès de la société. Toutefois, il est important de noter que, malgré sa puissance, le PageRank n'est qu'un aspect parmi d'autres des algorithmes de recherche modernes, qui intègrent désormais une multitude de critères pour fournir des résultats toujours plus personnalisés et pertinents.

**REFERENCES :**

<http://images.math.cnrs.fr/Comment-Google-classe-les-pages.html?lang=fr>  
Cours de ETH Zurick : Chapter 11 <https://disco.ethz.ch/courses/ti2/>  
The Anatomy of a Large-Scale Hypertextual Web Search Engine  
Langage utilisé : Python