

Aspects computationnels de l'algorithme PageRank

Martin Medina et Étienne Mitchell-Bouchard

24 mars 2025

1 Introduction

1.1 Contexte

Avec la popularité croissante du Web et l'arrivée des moteurs de recherche (*Search Engines* en anglais) au début des années 1990, la façon dont les gens accèdent et consultent l'information a radicalement changé. Tandis qu'avant les seules sources de consultation étaient les livres physiques et les encyclopédies, maintenant une quantité inimaginable de connaissances sont à quelques clics de distance. Ce sont les moteurs de recherche qui ont facilité la recherche et l'accès à cette vaste quantité de connaissances sur le Web. Cependant, c'est justement cette énorme quantité de ressources qui pose un grand problème et qui continue à être d'actualité: dans quel ordre devrait-on afficher les résultats lors d'une requête d'un utilisateur? Les méthodes essayant de résoudre ce problème sont appelées les méthodes de Récupération d'Information (*Information Retrieval methods* en anglais [1]) et la course pour créer la méthode la plus efficace en termes de précision et de vitesse changera l'industrie pour toujours.

La première méthode de Récupération d'Information suggérée a été LSI (*Latent Semantic Indexing*). Elle utilisé la Décomposition en Valeurs Singulières (SVD - *Singular Value Decomposition*) [1], une technique d'analyse matriciel qui réduit la dimensionnalité d'une matrice terme-document, matrice qui associe à chaque terme (mot) une collection de tous les documents qui le contiennent, tout en conservant l'essentiel de l'information. Plus simplement, la SVD est synonyme d'avoir une grande pile de cubes de différentes couleurs (représentant tous les mots et documents), mais au lieu de garder chaque petit cube individuellement (matrice au complet), ils sont regroupés par similitude pour former moins de cubes mais plus grands (plus représentatifs); c'est la réduction de la dimensionnalité. Cela permet de garder l'essentiel des informations tout en réduisant la quantité totale à manipuler, ce qui, en pratique, permettait de transformer des mots en concepts qui regroupent des termes en commun. Par exemple, les synonymes de "maison" comme domicile, habitation, résidence ou appartement, seront dans le même groupe. Mais aussi, les mots utilisés dans plusieurs contextes comme "battre" (frapper, compétition, battre des oeufs, etc.) seront dans des groupes séparés. Cette méthode fonctionne bien pour suggérer une collection plus large et plus pertinente de documents à la requête de l'utilisateur que simplement chercher les mots de la requête dans la matrice terme-documents, cependant elle a deux grands problèmes:

1. Le calcul pour retrouver les groupes de pages correspondant à la requête doit être effectué à chaque requête, ce qui est extrêmement lent et qui limite aussi la méthode LSI à des collections de documents relativement petites (ce qui n'est pas pertinent dans ce cas considérant le nombre de pages sur le Web) [1].
2. Il faudrait encore trouver la façon de trier les pages suggérées à l'utilisateur pour que les plus pertinentes apparaissent en premier.

Le premier problème est très important car les utilisateurs donnent une grande importance à la vitesse des services, dans ce cas-ci l'affichage des pages. De même, il mentionne la plus grande contrainte

de ce problème: le Web, qui présente plusieurs caractéristiques uniques qui en font une collection de documents particulièrement difficile à analyser [1-2]:

- Le nombre de documents dans le Web est immense (plus de 50 milliards en février 2024 [18]), ce qui requiert d'une proposition très efficace.
- Les documents web ne sont pas soumis à un processus de révision, ce qui donne lieu à du contenu redondant, de très mauvaise qualité et même parfois à de fausses informations.
- Le web est en constante évolution, des milliers de pages sont modifiées, ajoutées ou supprimées à chaque jour, ce qui en fait du Web une source très dynamique.

D'autre part, le deuxième problème est le plus important, même autant que la pertinence des pages suggérées puisque les utilisateurs regardent normalement uniquement les 5-10 premières pages suggérées par le moteur de recherche [1]. Ils regardent très rarement, voir jamais, toutes les pages suggérées, donc si les pages les plus pertinentes se trouvaient à la fin de la liste, l'utilisateur ne connaîtra jamais leur existence. Il fallait donc trouver un critère qui permette de classifier les pages en ordre d'importance, mais comment faire? Cette question nous mène au troisième problème de la méthode LSI: elle se centre uniquement sur le contenu des documents. On aimerait donc une méthode qui prenne aussi en compte la qualité et la fiabilité (importance) des documents dans le Web. Heureusement, il existe une caractéristique du Web qui permet de faire cela: sa structure d'hyperliens.

Sur le Web, une page peut faire référence à une autre avec un hyperlien, qui pourrait être représenté comme une flèche qui pointe vers la page référencée. Un utilisateur peut utiliser ces hyperliens pour se déplacer d'une page à une autre, et de cette façon se déplacer sur le Web. En effet, cette structure créée par les hyperliens des pages du Web reliant les différentes pages entre elles donne une information très précieuse qui sera exploitée par Google pour concevoir la méthode de Récupération d'Information PageRank. Lors de la création de la méthode, des enjeux économiques énormes étaient en jeu, et, si leur idée fonctionnait, ils auraient pu contrôler le marché et générer des milliards de dollars [3].

1.2 PageRank

En 1998, les fondateurs de Google: Larry Page et Sergey Brin, ont eu une idée pour réussir à suggérer et trier des pages lors de la recherche d'un utilisateur mieux que personne. Leur objectif était de réussir à quantifier l'importance des pages dans le Web et de cette façon pouvoir trier les pages suggérées à l'utilisateur pour que les résultats les plus utiles soient dans les premières positions lorsqu'ils effectuent une recherche.

Pour faire cela, leur idée était de quantifier/approximer l'importance de chaque page par une valeur en fonction du nombre et de la qualité des hyperliens dans le Web qui pointent vers elle. Leur théorie était que plus les pages qui font référence à une page (ont un hyperlien vers cette page) sont nombreuses, plus cette page devrait être importante. En conséquent, si une page importante fait référence vers une autre page, cette autre page devrait, en théorie, être importante elle aussi. Donc, les hyperliens pointant vers une page seraient interprétés comme des "votes d'importance" vers celle-ci et les hyperliens partant de pages considérées comme importantes auront plus de poids dans la votation que les hyperliens partant de pages moins importantes [3]. On remarque que cette définition de l'algorithme pointe vers un calcul récursif, ce qui est le cas et nous en parlerons d'avantage dans la section suivante.

Une comparaison peut être faite avec le monde de la recherche scientifique. Si un article est référencé par beaucoup d'articles, alors cet article doit être important puisque son contenu a été utile dans beaucoup d'autres recherches. Également, si cet article considéré comme important fait référence à un autre article, cet autre article devrait aussi être important puisqu'un article important a utilisé son contenu pour être rédigé.

Donc, Larry et Sergey voulaient attribuer à chaque page du Web une note d'importance (appelée *PageRank*) qui représente l'importance de cette page dans le Web. Une page avec un PageRank plus haut sera considérée comme plus importante qu'une page avec un PageRank plus bas, et donc sera

dans une position plus haute lors de l'affichage des pages suggérées. Cette méthode de Récupération d'Information sera aussi appelée PageRank et c'est celle qui sera à la base de leur premier moteur de recherche. Si leur idée pour mesurer l'importance d'une page s'avérait bonne, ils auraient trouvé une solution au troisième problème de LSI. En effet, "PageRank utilise non pas le contenu (textuel ou visuel) des pages, mais la structure des liens entre elles" [2].

Il est important de remarquer que PageRank est un algorithme qui mesure l'importance des pages et non leur pertinence, deux concepts qui sont différents. Tandis que la pertinence garantit que les résultats correspondent à la requête de l'utilisateur, l'importance (PageRank) propulse les pages autoritaires (fiables et de haute qualité) plus haut dans les résultats lors d'une requête. Voici plus de détails sur ces concepts:

- **L'importance** d'une page est déterminée par sa popularité, qualité et fiabilité dans le Web (indépendamment de la requête de l'utilisateur). Dans le cas de PageRank, elle est calculée en fonction du nombre et de la qualité (poids) des hyperliens pointant vers elle. Les pages avec un grand nombre de liens entrants provenant de pages fiables et de haute qualité sont considérées comme plus importantes et seront affichées en premier à l'utilisateur (premières position).
- La **pertinence** d'une page, en revanche, fait référence à la manière dont le contenu d'une page correspond à une requête spécifique (basée sur des mots-clés, le contenu, etc.). Il existe des méthodes de Récupération d'Information qui sont centrées sur la pertinence des pages, comme HITS, mais leur principal problème est le même que celui de LSI: leur dépendance à la requête les obligent à faire tous les calculs à chaque requête, ce qui est très lent et coûteux considérant la taille du Web. Pour plus d'informations sur HITS, consultez [1].

Cette notion d'indépendance à la requête permettra à Google de calculer une unique fois le PageRank de toutes les pages du Web et ensuite de les cataloguer dans un fichier terme-document comme celui de la méthode LSI. Au moment où un utilisateur fait une requête, une liste triée en ordre décroissant des pages liées aux termes de la requête est présentée par PageRank, et elle sera affichée à l'utilisateur presque instantanément. L'indépendance à la requête est donc une solution au premier problème de LSI, et de toutes les méthodes centrées sur la requête, puisqu'elle permet de donner une liste de pages très rapidement lors de différentes requêtes des utilisateurs. Un unique PageRank pour chaque page du Web calculé au début est valide pour plusieurs requêtes différentes, ce qui fait en sorte que la vitesse d'affichage des pages à l'utilisateur dépend uniquement de l'algorithme de recherche des pages dans le fichier terme-document, et de l'algorithme de tri utilisé pour trier les pages sélectionnées en fonction de leur PageRank.

C'est aussi pertinent de clarifier que, même si les deux concepts sont fortement reliés, le PageRank d'une page n'est pas uniquement calculé en fonction du nombre d'hyperliens pointant vers la page. En effet, "Un simple comptage des hyperliens entrants ignore la thèse de PageRank selon laquelle une page est importante si elle est référencée (pointée à l'aide d'un hyperlien) par d'autres pages importantes. Le comptage des hyperliens entrants mesure uniquement la quantité, et non la qualité de ces liens." [1]. Le calcul du PageRank des pages du Web est bien plus compliqué que juste compter le nombre d'hyperliens pointant vers chaque page.

2 Modélisation

2.1 Le Web

Le Web, de façon très simplifiée, peut être vu comme une collection de pages qui sont reliées entre elles par des hyperliens. On peut donc modéliser la structure du Web à l'aide d'un graphe orienté.

Définition 1: Un **graphe orienté** est un ensemble de points, appelés **noeuds**, reliés par des lignes, appelées **arêtes**, qui ont un sens unique, comme des flèches. Si deux noeuds sont reliés par une arête, on peut se déplacer d'un noeud à un autre seulement dans la direction indiquée par l'arête.

Dans le cas du Web, chaque page sera représentée par un noeud et chaque lien sera représenté par une arête orientée. Par exemple, si une page A possède un lien pointant vers une page B , alors, dans le graphe, le noeud A sera relié au noeud B par une arête orientée dans le sens $A \rightarrow B$. Le résultat de cette modélisation sera un énorme graphe qui représente la structure des hyperliens du Web. Il est important de remarquer que les hyperliens sur une page qui pointe vers cette même page ne sont pas pris en compte, sinon le PageRank des pages serait facilement manipulable: le créateur d'une page pourrait ajouter une très grande quantité d'hyperliens pointant vers sa propre page ce qui augmentera son PageRank. Cette notion de possibilité de manipulation du PageRank doit être évitée le plus possible, nous en parlerons dans la Section 4.

Voici un exemple simple d'un ensemble de 5 pages:

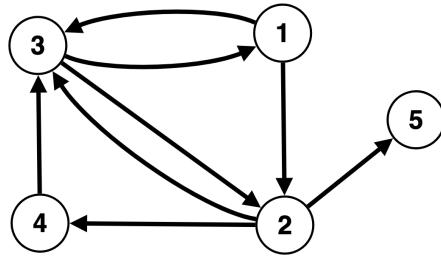


Figure 1: Exemple de graphe orienté

Dans cet exemple, la page 1 possède un lien qui dirige vers la page 3 et un autre vers la page 2, la page 2 contient trois liens pointant vers les pages 3, 4 et 5, la page 3 possède deux liens dirigeant vers les pages 1 et 2, la page 4 possède un lien vers 3, et finalement la page 5 ne possède aucun lien.

La notion du poids des votes des différentes pages en fonction de leur propre importance, peut être visualisé de la façon suivante. Pour cet exemple, imaginons que la page 3 est considérée comme plus importante que les autres (puisque c'est celle vers laquelle le plus de pages pointent), et donc ses liens auront plus de valeur dans le calcul du PageRank que ceux des autres pages.

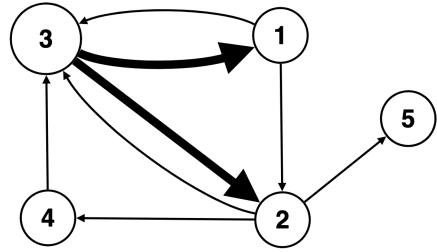


Figure 2: Représentation de la notion de poids

2.2 Intuitions pour le calcul de PageRank

Comme mentionné précédemment, l'idée de PageRank est que, dans le calcul de l'importance des pages, les votes (liens) provenant de sites importants auront plus de poids que ceux provenant de sites moins importants. De plus, la valeur d'un vote provenant d'une page doit être atténuée par le nombre de sites vers lesquels cette source fait référence, car plus elle pointe vers différents sites, moins

chaque lien a d'importance. Cela revient donc à pondérer la valeur de chaque hyperlien d'une page par le nombre total d'hyperliens qu'elle contient. On remarque que la définition de PageRank est récursive: le calcul de l'importance PageRank d'une page dépend du PageRank (importance) des pages qui pointent vers elle, qui dépend à leur tour du PageRank des pages qui pointent vers elles, et ainsi de suite.

Notons $\text{PR}(i)$ le PageRank de la page i . Le calcul du PageRank de la page **3** de la Figure 1 sera donné par la formule suivante:

$$\text{PR}(3) = \frac{\text{PR}(1)}{2} + \frac{\text{PR}(2)}{3} + \frac{\text{PR}(4)}{1}$$

Dans ce cas, pour le calcul du PageRank de la page **3**, la page **4** transférera toute la valeur de son PageRank puisque **3** est la seule page vers laquelle elle pointe, tandis que les pages **1** et **2** devront répartir leur PageRank entre toutes les pages vers lesquelles elles pointent (la page **1** transférera la moitié vers la page **3** et la page **2** juste un tiers). De plus, on constate effectivement qu'une page aura une PageRank plus élevé si un plus grand nombre de pages possèdent des hyperliens vers elle et, plus important, si les pages pointant vers elle sont importantes (PageRank élevé) car le calcul du PageRank est fait directement en fonction de celui des autres pages. L'importance des pages pointant vers une page ont donc une influence sur l'importance de celle-ci.

De forme générale, le PageRank d'une page i peut être représenté par la formule suivante:

$$\text{PR}(i) = \sum_{j \in W_i} \frac{\text{PR}(j)}{L(j)} \quad (1)$$

où W_i représente l'ensemble des pages du Web qui possèdent un hyperlien vers la page i et $L(j)$ correspond au nombre d'hyperliens que la page j possède.

Comme nous l'avons mentionné précédemment, le PageRank est une définition récursive, donc son calcul aura forcément besoin d'être réalisé en itérations. Cette formule n'est pas facilement calculable en elle-même, nous devons donc modéliser ce problème de manière à pouvoir la calculer de façon plus simple et itérative.

2.3 Chaînes de Markov

Définition 2: Une **chaîne de Markov** est un modèle mathématique composé d'un ensemble d'états (noté \mathcal{X}) et des transitions qui connectent les états. Chaque transition possède une probabilité, ce qui veut dire que, depuis un état donné, on peut se déplacer vers un autre avec la probabilité associée à cette transition. La probabilité de transition d'un état i vers l'état j à l'instant n s'écrit $P[X_{n+1} = j | X_n = i]$, où $X_n = i$ signifie qu'à l'instant n le système se trouve dans l'état i . La propriété markovienne des chaînes de Markov signifie que la probabilité de transitionner vers un état dépend uniquement de l'état actuel, sans tenir compte des états précédents (les chaînes de Markov n'ont pas de mémoire [2]). Formellement, cela se traduit par:

$$P[X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \dots, X_1 = i_1, X_0 = i_0] = P[X_{n+1} = j | X_n = i]$$

Définition 3: Les probabilités de transition d'une chaîne de Markov peuvent toutes être représentées dans une matrice appelée la **matrice des probabilités de transition** de la chaîne. Cette matrice, notée P , est une matrice de dimension $n \times n$, où n est le nombre d'états de la chaîne. L'élément $P_{i,j}$ (situé à la i -ème ligne et la j -ème colonne) correspond à la probabilité de transition de l'état i vers l'état j en une étape:

$$P_{i,j} = P[X_{n+1} = j | X_n = i]$$

Cette matrice doit être stochastique, ce qui veut dire que tous ses éléments doivent être positifs et la somme des probabilités sur chaque ligne est toujours égale à 1. Formellement, on écrit:

$$P_{i,j} \geq 0, \forall i, j \quad \text{et} \quad \sum_{j=1}^n P_{i,j} = 1, \forall i \quad (2)$$

De plus, à l'aide de cette matrice, on peut obtenir la probabilité totale d'être à l'instant n dans l'état i avec la formule suivante:

$$P[X_n = i] = \sum_{j=1}^n P(X_{n-1} = j) P_{i,j} \quad (3)$$

Cela équivaut à prendre en compte toutes les transitions possibles vers i et à les pondérer en fonction de leur probabilité.

On a vu que dans un graphe orienté on peut se déplacer d'un noeud à un autre à l'aide des arêtes tout en respectant leur orientation. Une chaîne de Markov peut donc être représentée par un graphe orienté sous certaines conditions:

- 1. Transitions indépendantes des états précédents:** la probabilité de transition vers un état futur ne dépend que de l'état actuel, et non de l'historique des états précédents.

→ Cela est vrai dans notre graphe représentant la structure d'hyperliens du Web car le déplacement d'un utilisateur d'une page à une autre dépend uniquement des hyperliens (transitions disponibles) présents dans la page actuelle sur laquelle il se trouve, et ne dépend effectivement pas des liens des pages qu'il a visité dans le passé.

- 2. Valeurs des arêtes pondérées par des probabilités:** chaque transition entre deux états est associée à une probabilité, qui correspond au poids de l'arête reliant les noeuds.

→ Comme on l'a mentionné précédemment dans l'intuition de la formule de PageRank (Section 2.2), la valeur de chaque hyperlien d'une page doit être pondérée par le nombre total d'hyperliens qu'elle contient. Cela transforme les poids des arêtes de notre graphe en probabilités de transition. En effectuant cette pondération, on suppose que tous les hyperliens d'une page sont également probables d'être cliqués par l'utilisateur lorsqu'il se trouve sur cette page. Étant sur une page, l'utilisateur choisi donc avec une probabilité uniforme vers qu'elle prochaine page se déplacer (parmi les hyperliens dans la page). On appelle cela un promeneur impartial [1-2].

D'autres distributions de probabilité peuvent être utilisées pour pondérer le poids des hyperliens d'un noeud [1], mais il faut faire attention, car cela peut causer des problèmes. Par exemple, si l'on décide que les probabilités de transition depuis un noeud sont pondérées en fonction du nombre de visites de chaque page au cours des derniers mois, l'hyperlien de la page ayant le plus grand nombre de visites sera choisi plus souvent par l'utilisateur (car cet hyperlien a la probabilité la plus élevée parmi tous les autres hyperliens de la page). Cela augmentera le nombre de visites vers cette page plus rapidement que pour les autres, rendant chaque fois plus probable que l'utilisateur choisisse ce lien, et ainsi de suite, jusqu'à ce qu'éventuellement cette page ait une probabilité de 1 d'être choisie. Dans ce travail, on utilisera toujours une distribution uniforme pour les probabilités de transition.

Pour l'instant, on a la matrice de transitions suivante pour notre exemple:

$$P = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4)$$

3. La somme des probabilités de transition depuis un état donné est égale à 1: la somme des poids des arêtes sortantes d'un noeud est toujours égale à 1, car le système doit forcément évoluer vers un état (à chaque instant, une transition doit être obligatoirement effectuée).

→ Grâce à la pondération effectuée pour satisfaire la condition précédente, cette condition est presque toujours respectée pour tous les noeuds de notre graphe. Cependant, si le graphe contient un noeud qui ne possède pas d'hyperliens (page 5 dans notre exemple), les probabilités de transition de cette page (5-ème ligne de P) seront toutes nulles et leur somme sera donc égale à 0 ($\sum_{j=1}^5 P_{5,j} = 0$). Pour régler ce problème, on ajoutera à tous les noeuds du graphe qui ne contiennent pas d'hyperliens, des arêtes (hyperliens) fictives vers tous les noeuds du graphe (chacune avec une probabilité de transition uniforme: $\frac{1}{n}$).

Graphiquement, cela peut être visualisé de la façon suivante:

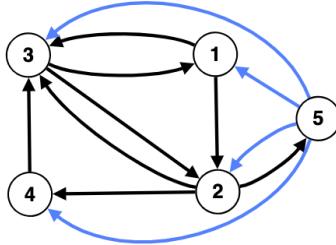


Figure 3: Graphe avec hyperliens fictifs ajoutés

où les arêtes en bleu représentent les hyperliens fictifs ajoutés.

Mathématiquement, cela équivaut à remplacer chaque ligne de zéros dans la matrice P par $\frac{e}{n}$, où e est un vecteur ligne de dimension $1 \times n$ dont toutes les composantes sont égales à 1. On obtient la matrice \bar{P} suivante:

$$\bar{P} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix} \quad (5)$$

Cette matrice satisfait toutes les équations de (2), \bar{P} est une matrice stochastique. Par la suite, nous expliquerons pourquoi cette modification est importante.

Les trois conditions sont respectées par notre nouveau graphe représentant la structure d'hyperliens du Web, alors ce graphe peut être représenté par une chaîne de Markov où les noeuds représentent l'ensemble d'états (pages du Web) et les arêtes représentent les transitions possibles. De plus, la matrice \bar{P} est la matrice des probabilités de transition de la chaîne de Markov représentant le graphe.

Maintenant, imaginons un utilisateur qui se promène dans cette chaîne de Markov représentant la structure du Web. S'il se promène assez longtemps, les pages qu'il aurait visité le plus seraient, en théorie, les pages les plus importantes. En effet, plus il y a de pages pointant vers une, plus il y a de chances que l'utilisateur visite la page [6]. Si l'utilisateur se déplace dans la chaîne pendant une durée infinie et on mesure la proportion de temps (étapes) qu'il est dans chaque page (nombre de fois qu'il visite chaque page), on obtiendra une mesure de l'importance pour chaque page qui correspondra au PageRank de chaque page. Donc, plus une page est importante, plus l'utilisateur la visitera souvent et, par conséquent, plus souvent il utilisera les hyperliens partant de cette page pour se déplacer vers une autre. De cette façon, le concept selon lequel un lien provenant d'une page importante a plus de poids qu'un autre provenant d'une page moins importante est préservé.

Calculer la proportion du temps pendant lequel l'utilisateur, se déplaçant à l'infini dans la chaîne, est dans chaque état (page), revient à calculer le vecteur des probabilités stationnaires de la chaîne.

Propriété 1: Soient P la matrice des probabilités de transition d'une chaîne de Markov et $P^{(n)}$ la n -ème puissance de celle-ci. L'élément $P_{i,j}^{(n)}$, composante (i, j) de la n -ème puissance de P , correspond à la probabilité de passer de l'état i à l'état j en n étapes. Lorsque n devient suffisamment grand, $P^{(n)}$ tend à converger vers une matrice où chaque ligne est égale au vecteur des probabilités stationnaire de la chaîne, noté π .

Définition 4: Le vecteur des probabilités stationnaires π d'une chaîne de Markov est un vecteur ligne de dimension $1 \times n$ qui représente la distribution des probabilités des états à l'équilibre, c'est-à-dire la proportion du temps (étapes) pendant lequel l'utilisateur, se déplaçant à l'infini dans la chaîne, est dans chaque état. Si P est la matrice des probabilités de transition de la chaîne, le vecteur stationnaire π satisfait les équations suivantes:

$$\pi P = \pi \quad \text{et} \quad \pi e^T = 1$$

ce qui signifie que la distribution des probabilités reste inchangée après une étape de transition (le vecteur π a convergé) et que ses composantes somment à 1.

Ce vecteur sera appelé le vecteur PageRank et chaque composante π_i correspond au PageRank de la page i . De plus, il satisfait les deux conditions suivantes:

$$\pi_j = \sum_{i=1}^n \pi_i P_{i,j} \quad , \forall j \in \mathcal{X}, j \in \{1, 2, \dots, n\}$$

qui montre que le PageRank de chaque page est calculé en fonction du PageRank des autres, et

$$\sum_{i=1}^n \pi_i = 1$$

qui montre que le vecteur PageRank π est bien une distribution de probabilités car ses composantes somment à 1.

En général, π ne peut pas être calculé exactement [1], mais, il peut être approximé par des multiplications successives de matrices-vecteurs. De la même façon que les colonnes de $P^{(n)}$ qui convergent vers π lorsque n tends vers l'infini, le vecteur π peut être approximé itérativement de la façon suivante:

- ~ Soit $\pi_{(0)}$ le vecteur initial choisi aléatoirement. Dans cette application, on initialisera $\pi_{(0)} = \frac{e}{n}$ pour attribuer une probabilité uniforme à toutes les pages d'être l'état initial de l'utilisateur (depuis lequel il va commencer). Cependant, cette initialisation n'a pas d'importance pour le résultat final de π .
- ~ En utilisant la Propriété 1, on obtient:

$$\pi_{(t)} = \pi_{(0)} P^t$$

- ~ Ensuite, grâce à (3), on a:

$$\pi_{(t)} = \pi_{(t-1)} P \tag{6}$$

- ~ Finalement, en utilisant la propriété de la convergence de $P^{(n)}$, on a:

$$\pi = \lim_{t \rightarrow \infty} \pi_{(t)} \tag{7}$$

Cependant, la convergence de cette méthode est assez lente, ce qui est problématique pour une matrice de la taille du Web, car chaque calcul est très coûteux. On aimerait donc réaliser le moins de calculs possible. Pour cela, nous utiliserons une condition d'arrêt (seuil de tolérance) pour accepter la convergence de (7) et minimiser ainsi le plus possible le nombre d'itérations (calculs). Cette méthode itérative est appelée la méthode de la puissance, et on l'expliquera en détail à la Section 3.3.

Finalement, en revenant à la condition 3), les états ne possédant aucun lien sont appelés **états absorbants**. Il est important de les éliminer, dans ce cas en ajoutant des liens fictifs, car sinon, lors du calcul de π , ces états absorberont tout le PageRank des autres pages. En effet, si aucun lien fictif n'est créé, lorsque l'utilisateur entre dans cette page, il ne pourra pas en sortir et restera dans cet état indéfiniment. Cela, à long terme, signifiera que l'utilisateur aura passé tout son temps dans cette page, rendant les états absorbants responsables de l'ensemble du PageRank. Si nous ne corrigions pas ce problème dans notre exemple où l'état 5 est absorbant, et que nous lui ajoutons un lien avec probabilité 1 vers lui-même pour obtenir une matrice stochastique, nous obtiendrons $\pi = (0, 0, 0, 0, 0, 1)$, ce qui n'est évidemment pas le PageRank correct.

Cela n'est pas problématique dans ce cas particulier, car dans cet exemple il y a un seul état absorbant et il n'est pas très important (il est simplement pointé par un lien). Cependant, seulement le retirer du graphe n'est pas une bonne idée. En effet, dans un graphe de la taille du Web, il pourrait y avoir plusieurs états formant un ensemble d'états absorbants dans lequel il pourrait y avoir des milliers de pages et, parmi elles, des pages très importantes qu'il faut ordonner et qu'il faut considérer dans le calcul du PageRank. [2] Pour cette raison, il est nécessaire de traiter les états absorbants en leur fournissant manuellement une sortie.

3 Résolution de base

3.1 Calcul d'un vecteur propre

Propriété 2: Si λ est une valeur propre d'une matrice de transition P , $n \times n$, alors $|\lambda| \leq 1$. De plus, il existe un vecteur propre associé à la valeur propre $\lambda = 1$ dont toutes les composantes sont positives ou nulles. [2]

Propriété 3: La matrice des probabilités de transition P d'une chaîne de Markov possède 1 parmi ses valeurs propres. [2] Donc, d'après la Propriété 2, la valeur propre dominante de toute matrice stochastique P est $\lambda = 1$. [1]

Par conséquent,

Propriété 4: Si l'itération (6) pour calculer π converge, elle convergera bien vers le vecteur propre unique de P pour la valeur propre 1: π , le vecteur de probabilités stationnaires de la chaîne de Markov. [1]

Alors le calcul du vecteur PageRank (π) revient à trouver le vecteur propre de la matrice des probabilités de transition P pour la valeur propre 1. On sait qu'on peut l'approximer itérativement avec (6), mais existe-t-il un meilleur algorithme? Malheureusement, il n'en existe pas encore. À cause de l'immense taille de la matrice des probabilités de transition du Web (plus de 50 milliards de lignes), le calcul itératif avec la méthode de la puissance (qui est basée sur (6) et (7)) est le meilleur choix possible. En effet, "Les méthodes directes (même celles optimisées), ainsi que les *eigensolvers*, ne peuvent pas gérer la taille énorme de cette matrice" [1].

Cependant, avant de calculer π avec la méthode de la puissance, il y a un dernier problème qu'il faut régler.

3.2 Ajustement de la matrice P

Comme on l'a vu dans la Section 2.3, la matrice de transition P (4) fabriquée directement avec le graphe d'exemple (Figure 1) n'est pas stochastique à cause de la présence d'un état absorbant. Il faut donc la modifier en remplaçant par $\frac{e}{n}$ les lignes pour lesquelles toutes les composantes sont nulles.

Ainsi, on obtient une matrice stochastique \bar{P} (5) qui est la matrice des probabilités de transition de la chaîne de Markov qui représente la version modifiée du graphe précédent (Figure 3). Cependant, cette matrice \bar{P} a un problème: elle n'est pas forcément irréductible (voir ci-bas). En réalité, "La structure du Web est telle que \bar{P} est presque certainement réductible." [1].

Définition 5: Une chaîne de Markov est **irréductible** lorsque pour tous les états de la chaîne, il est possible de passer d'un état à un autre, même si cela peut prendre plusieurs étapes (tous les états ont un chemin qui les lie entre eux). Plus spécifiquement, une chaîne de Markov est irréductible si un utilisateur qui se promène dans la chaîne est capable de commencer à un état initial et revenir à celui-ci en passant par tous les autres états avant de revenir. Cela veut dire qu'il n'y a pas d'états "isolés" que l'on ne pourrait jamais atteindre à partir d'un autre état.

Proposition 1: La matrice des probabilités de transition d'une chaîne de Markov irréductible est dite irréductible si la chaîne de Markov est irréductible.

Définition 6: Une chaîne de Markov irréductible est **récurrente positive** si elle possède un nombre fini d'états [15]. Cela sera toujours le cas sur le Web puisque, même si le nombre est très grand, il possède (et possédera toujours) un nombre fini de pages (noeuds).

Propriété 5: Le vecteur de probabilités stationnaires π d'une chaîne de Markov irréductible et récurrente positive est unique.

Finalement, des Propriétés 2, 3, 4 et 5 on peut dériver le théorème suivant:

Théorème 1 (Perron-Frobenius): Soit A la matrice des probabilités de transition d'une chaîne de Markov, donc une matrice carrée et positive (tous ses éléments sont ≥ 0). Si A est irréductible, alors elle possède une valeur propre dominante unique (la plus grande en valeur absolue), et un unique vecteur propre positif dont les éléments somment à 1 (π tel que $\sum_{i=1}^n \pi_i = 1$) correspondant à cette valeur propre. De plus, le théorème assure que l'algorithme itératif de la méthode de la puissance convergera vers ce vecteur propre dominant, à condition que la matrice A soit irréductible. Cet unique vecteur propre correspond au vecteur des probabilités stationnaires de A (Propriété 6).

Donc, sachant que \bar{P} est positive, que 1 est une de ses valeurs propres (Propriété 3) et qu'elle est en plus la valeur propre dominante (Propriété 2), il faudrait juste que \bar{P} soit irréductible pour que le vecteur propre associé à la valeur propre 1, qui correspond au vecteur PageRank (Propriété 4), soit unique et puisse être calculé itérativement avec la méthode de la puissance (puisque il convergera vers une solution unique).

Maintenant, on peut voir effectivement pourquoi le fait que \bar{P} ne soit pas irréductible est un problème. Si \bar{P} n'est pas irréductible, son vecteur PageRank pourrait ne pas être unique et son approximation avec la méthode de la puissance n'est pas possible (le vecteur propre ne convergera jamais).

Pour forcer l'irréductibilité de la chaîne de Markov représentant la structure du Web et ainsi pouvoir utiliser la méthode de la puissance pour calculer son vecteur des probabilités stationnaires (qui équivaut au vecteur PageRank), Brin et Page ont décidé d'ajouter une probabilité à toutes les composantes de la matrice des probabilités de transition afin de rendre tout état directement atteignable depuis tous les autres états. L'idée derrière cela était de simuler le comportement d'un utilisateur qui effectuerait une recherche dans la barre de navigation pour aller vers une autre page, au lieu d'utiliser les hyperliens. Cela correspondrait à ajouter une probabilité que l'utilisateur se téléporte vers n'importe quelle page à n'importe quel moment. Vu que maintenant on pourrait atteindre n'importe quelle page dans une étape, la chaîne est bien évidemment irréductible (par définition), donc la matrice des probabilités de transition associée l'est aussi (Proposition 1).

Originalement, Google a décidé que toutes les pages avaient la même probabilité d'être sélectionnées lors d'une téléportation. Pour modéliser ce comportement, ils ont additionné une **matrice de perturbation** $E = \frac{e^T e}{n}$ (matrice $n \times n$ avec tous ses éléments égal à $\frac{1}{n}$) à leur matrice \bar{P} . De plus, afin d'avoir un meilleur contrôle sur le poids de cette probabilité de téléportation, ils ont introduit un

scalaire $\alpha \in [0, 1]$ appelé **facteur d'amortissement**. En combinant ces éléments, ils ont obtenu la nouvelle matrice des probabilités de transition suivante :

$$\bar{P} = \alpha \bar{P} + (1 - \alpha)E$$

On peut remarquer dans la formule que le rôle joué par le facteur d'amortissement est très important. En effet, on constate que si $\alpha = 1$, la matrice de perturbation est complètement ignorée et, en conséquence, aucune téléportation ne sera effectuée. C'est comme si aucun ajustement n'aurait été effectué, la matrice risque donc de ne pas être réductible. Par contre, si $\alpha = 0$, la matrice \bar{P} est complètement ignorée et l'utilisateur pourra uniquement se déplacer avec des téléportations vers n'importe quelle page, avec une probabilité uniforme de $\frac{1}{n}$. Dans ce cas, la structure d'hyperliens originale du Web sera complètement ignorée. Nous parlerons plus en détail du rôle que joue le facteur d'amortissement dans la convergence de PageRank dans la Section 5.1.

Si on voulait porter cette modification dans la formule (1), on obtient la formule originale de PageRank proposée par Page et Brin :

$$\text{PR}(i) = \frac{1 - \alpha}{n} + \alpha \sum_{j \in W_i} \frac{\text{PR}(j)}{L(j)}$$

Effectivement, on peut voir que si une page ne contient aucun hyperlien, son PageRank ne sera pas égal à 0, mais plutôt à $\frac{1-\alpha}{n}$. Cette valeur correspond à celle ajoutée à \bar{P} par $(1 - \alpha)E$ dans la représentation matricielle.

Finalement, un peu plus tard, Google a modifié sa distribution uniforme des probabilités de téléportation pour adopter une approche plus réaliste et moins démocratique en utilisant la matrice de perturbation $E = e^T v$, où v est un vecteur ligne positif de dimension $1 \times n$ qui contient des probabilités "personnalisées" de téléportation pour chaque page [1]. Ce vecteur est appelé **vecteur de personnalisation** et permet de manipuler (augmenter ou diminuer) la probabilité de téléportation vers certaines pages en particulier (choisies). Malgré l'existence d'autres méthodes pour forcer l'irréductibilité, Google continue de privilégier l'approche $E = e^T v$ [1]. Ce vecteur de personnalisation, bien qu'il soit plus flexible que la matrice de perturbation uniforme, a été fortement critiqué en raison du manque d'impartialité dans l'ordre des pages qu'il génère. Nous discuterons de cela plus en détail dans la Section 4.2.

Appliquons maintenant cet ajustement à la matrice \bar{P} (5) notre exemple de la Figure 1 :

- Notre matrice de perturbation sera :

$$E = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

- On prendra $\alpha = 0.85$ comme facteur d'amortissement. Les dernières communications de Google ont indiqué qu'ils utilisent aussi un facteur d'amortissement $\alpha = 0.85$. Ils affirment que cela permet de prendre en compte correctement l'importance de chaque page, tout en réduisant au maximum le taux de convergence, c'est-à-dire le nombre d'itérations nécessaires à la méthode de la puissance pour que le vecteur propre converge.
- On obtient alors la nouvelle matrice des probabilités de transition suivante :

$$\bar{P} = 0.85 \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix} + 0.15 \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix} = \begin{bmatrix} \frac{3}{100} & \frac{91}{200} & \frac{91}{200} & \frac{3}{100} & \frac{3}{100} \\ \frac{3}{100} & \frac{3}{100} & \frac{47}{150} & \frac{47}{150} & \frac{47}{150} \\ \frac{91}{200} & \frac{91}{200} & \frac{3}{100} & \frac{3}{100} & \frac{3}{100} \\ \frac{3}{100} & \frac{3}{100} & \frac{22}{100} & \frac{3}{100} & \frac{3}{100} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

C'est cette matrice \bar{P} , généralement appelée "la matrice Google" [1], qui est irréductible et que sa distribution stationnaire π est le vecteur PageRank. Comme mentionné précédemment, puisque la matrice est irréductible, ce vecteur pourra être trouvé (approximé) de manière itérative avec la méthode de la puissance.

3.3 Méthode de la puissance

Comme mentionné précédemment, pour trouver le vecteur stationnaire de la matrice irréductible \bar{P} (dont la taille, dans la réalité, est de l'ordre de milliards), Google a choisi la méthode de la puissance.

La méthode de la puissance consiste à approximer le vecteur propre dominant de \bar{P} en appliquant la formule (6) itérativement sur la matrice \bar{P} et un vecteur initial $\pi_{(0)}$, jusqu'à ce que la différence (écart) entre deux itérations soit inférieure à un seuil de tolérance β (souvent 1×10^{-6}) pour accepter la convergence. La méthode de la puissance itère donc jusqu'à ce que la condition $\|\pi_{(k)} - \pi_{(k-1)}\| < \beta$ soit satisfaite. Comme mentionné dans la Section 2.3, nous prenons un vecteur $\pi_{(0)}$ qui suit une distribution uniforme, donc $\pi_{(0)} = \frac{e}{n}$, et nous sommes assurés de converger vers le vecteur propre dominant correspondant au vecteur des probabilités stationnaires de \bar{P} , qui est le vecteur PageRank.

En résumé:

1. Le calcul initial est:

$$\pi_{(1)} = \pi_{(0)} \bar{P}$$

2. Ensuite, on répète le calcul suivant jusqu'à convergence de π (c'est-à-dire jusqu'à ce que $\|\pi_{(k)} - \pi_{(k-1)}\| < \beta$):

$$\pi_{(k)} = \pi_{(k-1)} \bar{P}$$

Un des avantages de la méthode de la puissance est que même avec un seuil de tolérance $\beta = 1 \times 10^{-6}$, le résultat sera presque identique à celui lorsque $\|\pi_{(k)} - \pi_{(k-1)}\| = 0$ (vecteur de probabilités stationnaires théorique). On va démontrer cela dans la Section 5.1. Ainsi, en acceptant la convergence à un seuil de tolérance, on économise des itérations (ce qui économise beaucoup de temps de calcul) et l'approximation réalisée par la méthode de la puissance continue à être très précise.

De même, comme mentionné dans la Section 2.3, le choix du vecteur initial $\pi_{(0)}$ n'a aucune influence sur le résultat (preuve dans [2]). Peu importe le vecteur initial choisi, tant que ses éléments sommrent à 1, le vecteur va converger vers le même vecteur de probabilités stationnaires (vecteur propre dominant). Dans ce cas, on a choisi une distribution uniforme juste pour représenter que toutes les pages ont la même probabilité d'être l'état initial.

En général, les principaux avantages de la méthode de la puissance sont [6] :

- Implémentation simple (Section 5.1).
- Itérations peu coûteuses grâce au produit matrice-vecteur creux (Sections 4.1 et 5.2).
- Convergence assurée avec une matrice irréductible (théorème de Perron-Frobenius).
- Taux de convergence indépendant de la dimension de la matrice (Section 5.1).
- Très grande précision et fiabilité numérique (Section 5.1).

Finalement, voici le vecteur des probabilités stationnaires donné par notre implémentation de la méthode de la puissance de la Section 5.1:

$$\pi \approx (0.1823, 0.2597, 0.3084, 0.1248, 0.1248) \quad (8)$$

On remarque que la page **3** possède le PageRank le plus élevé et est donc considérée comme la page la plus importante de notre graphe. De même, les pages **4** et **5** ont le PageRank le plus bas, elles sont

donc considérées comme les pages les moins importantes. Cela a du sens, car la page **3** est celle qui est pointée par le plus d'hyperliens (trois hyperliens), tandis que les pages **4** et **5** ne sont pointées que par un seul lien. La page **1** est également pointée par un seul lien, mais cet lien provient de la page **3**. Ainsi, grâce au principe selon lequel un vote provenant d'une page importante a plus de poids, son PageRank est plus élevé que celui des pages **4** et **5**, qui ne sont pas pointées par la page **3**. L'lien qui pointe vers ces dernières a donc moins de poids que celui qui pointe vers **1**, ce qui rend les pages **4** et **5** moins importantes que **1**.

3.4 Au moment d'une requête

PageRank n'est qu'une composante du complexe système de classement de Google. En effet, Google combine PageRank à une grande variété d'autres algorithmes qui mesurent des caractéristiques différentes des pages. PageRank est une mesure de l'importance des pages du Web ; ainsi, cette mesure est combinée à celle des autres algorithmes pour obtenir un classement global. Cependant, puisque dans ce cas on s'intéresse uniquement à PageRank, on va supposer qu'il est le seul algorithme utilisé et que le classement retourné par PageRank est le seul pris en compte pour déterminer l'ordre d'affichage des pages. Pour poursuivre, nous expliquerons le principe derrière le fonctionnement de Google avant qu'une requête soit effectuée par un utilisateur et au moment de celle-ci.

1. Préparation des pages avant qu'une requête soit effectuée:

- Le vecteur PageRank du Web est calculé. Ce vecteur est valide et peut être utilisé pour n'importe quelle requête grâce à son indépendance de celle-ci, comme mentionné dans les sections précédentes. Cependant, à cause de la volatilité du Web, également mentionnée précédemment, ce vecteur doit être actualisé en permanence. On discutera plus en détail des défis de l'actualisation de PageRank dans la Section 4.1. Le vecteur PageRank calculé aura la même forme que (8), mais il contiendra un élément par page. Puisque ce vecteur représente une distribution (distribution stationnaire de la chaîne), la somme de ses éléments sera égale à 1.
- Chaque fois qu'une page Web est ajoutée ou modifiée, il est également important d'actualiser le fichier terme-document. Ainsi, chaque fois qu'une page est ajoutée ou modifiée, son contenu est analysé et son nom est ajouté dans les listes de documents correspondant aux termes présents dans le contenu. De cette façon, le système d'indexation sera toujours à jour, permettant à l'algorithme de recherche de trouver de façon fiable toutes les pages contenant les termes de la requête de l'utilisateur.

2. Lorsqu'un utilisateur effectue une requête:

- Premièrement, un algorithme analyse la requête de l'utilisateur et repère tous les termes qu'elle contient. Ensuite, les pages présentes dans les listes reliées à chaque terme dans le fichier terme-document sont sélectionnées. Cela forme un sous-ensemble de pages appelé l'ensemble de pertinence pour la requête [1]. Ensuite, l'ensemble de pertinence est étendu pour établir certaines associations sémantiques et donc aussi inclure les documents qui contiennent des synonymes des termes, tel que faisait LSI (Section 1.1). Cet élargissement de l'ensemble de pertinence peut être fait en ajoutant les pages qui pointent vers les pages de l'ensemble de pertinence ou qui sont pointées par celles-ci. Cependant, Google utilise actuellement d'autres algorithmes plus avancés et complexes, tels que des modèles d'apprentissage automatique, pour traiter la requête et les termes qui la composent, en essayant également de comprendre le contexte et le sens des mots de la requête. Ces algorithmes complexes ont pour objectif d'améliorer l'expérience des utilisateurs et d'augmenter la pertinence des pages sélectionnées qui seront présentées à l'utilisateur.
- Une fois que les documents pertinents pour la requête de l'utilisateur sont sélectionnés (ensemble de pertinence), ils sont triés de façon décroissante en fonction du PageRank de chaque page et, finalement, ils sont affichés dans cet ordre à l'utilisateur. Ainsi, PageRank

ne dépend effectivement pas de la requête. Chaque document possède un score PageRank indépendant de toutes les requêtes.

Pour finaliser, notre exemple de la Figure 1, imaginons que l'ensemble de pertinence, noté \mathcal{N} , contient les pages **1**, **2**, **3** et **5**:

$$\mathcal{N} = [1, 2, 3, 5]$$

Le Pagerank correspondant à chaque page se trouve dans (8). On a:

- $\text{PR}(1) = \pi_1 = 0.1823$
- $\text{PR}(2) = \pi_2 = 0.2597$
- $\text{PR}(3) = \pi_3 = 0.3084$
- $\text{PR}(5) = \pi_5 = 0.1248$

Donc, lors de sa requête, l'utilisateur verra affichée en première position la page **3**, en deuxième position la page **2**, en troisième position la page **1**, et en dernière position la page **5**.

Finalement, on remarque qu'il peut y avoir deux pages avec le même PageRank (la page **4** et la page **5**). Dans ce cas, l'ordre d'affichage des pages pourrait être choisi au hasard. Toutefois, il est également possible d'utiliser d'autres critères, tels que la popularité mesurée par le nombre de visites ou la qualité du contenu évaluée par des algorithmes d'apprentissage automatique.

4 Avantages et Inconvénients de PageRank

L'implémentation de base décrite dans la section précédente n'est pas parfaite. Elle rencontre plusieurs problèmes, mais heureusement la grande majorité d'entre eux ont une solution. Cependant, l'algorithme de PageRank lui-même présente à la fois des points faibles et des points forts. Les points faibles sont difficiles à corriger et nécessitent de la recherche pour trouver de nouveaux algorithmes alternatifs à PageRank. Dans cette section, nous détaillerons ces aspects.

4.1 Problèmes de la résolution de base

1. Complexité temporelle:

- L'implémentation de la Section 3 se base sur la multiplication d'un vecteur $1 \times n$ et une matrice $n \times n$ sur plusieurs itérations. Le calcul et la preuve de la complexité seront détaillés dans la Section 5.1, mais cette multiplication se fait généralement en $\mathcal{O}(n^2)$. Cette complexité polynomiale est problématique surtout lorsqu'on manipule des matrices énormes, comme dans le cas de PageRank (pour le cas du Web, $n \approx 50$ milliards). Cela rend le calcul de PageRank extrêmement lent. En effet, le calcul de PageRank pour Google peut prendre des jours même avec des dizaines d'ordinateurs extrêmement puissants [1].
- Cela est à cause du fait que l'implémentation de base, qui utilise une multiplication matricielle traditionnelle, n'exploite pas au maximum les propriétés de la matrice des probabilités de transition du Web. Dans le Web, une page n'est pas reliée à toutes les autres pages. En effet, en moyenne une page contient environ 10 hyperliens [16]. Puisque, lorsqu'une page qui ne pointe pas vers une autre est représenté par un 0, la matrice des probabilités de transition du Web possède une énorme quantité d'éléments nuls. Pour être plus précis, il y a beaucoup plus d'éléments nuls que non nuls. La matrice des probabilités de transition du Web est donc une **matrice creuse**. Cette propriété peut être exploitée en optimisant la multiplication matricielle, qui aura maintenant une complexité linéaire de $\mathcal{O}(n)$. On présentera plus en détail cette méthode de multiplication matricielle dans la Section 5.2, mais en

résumé elle consiste à modifier la matrice originale pour ne conserver que les éléments non nuls, ce qui accélère la multiplication. Cependant, pour des matrices de la taille de celles de Google, même cette optimisation n'est pas suffisante, l'utilisation du parallélisme est donc impérative pour un problème de ce type [1].

2. Taux de convergence:

- Sachant que la multiplication matricielle est l'opération qui consomme le plus de temps dans le calcul de PageRank, le taux de convergence est un élément crucial de l'algorithme. On sait que la méthode de la puissance appliquée à la matrice \bar{P} est assurée converger vers le vecteur propre dominant de celle-ci, cependant la vitesse de cette convergence dépend principalement du facteur d'amortissement (démonstration dans la Section 5.1). En effet, la méthode de la puissance converge plus vite lorsque la différence entre la première valeur propre et la deuxième augmente. Cependant, la structure du Web provoque avec une forte probabilité que la première valeur propre $\mu_1 = 1$ et la deuxième $\mu_2 \approx 1$, ce qui fait que le taux de convergence de la méthode de la puissance diminue à la même vitesse que $\alpha k \rightarrow 0$ où k représente le nombre d'itérations de la méthode de la puissance [1]. Cela provoque que le nombre d'itérations de la méthode de la puissance avant la convergence augmente linéairement au fur et à mesure que le facteur d'amortissement augmente (Section 5.1). Cependant, Google affirme que un facteur d'amortissement grand conduit à un calcul plus réel de l'importance des pages. Alors, il y a un conflit, puisque on aimerait diminuer le plus possible le facteur d'amortissement pour accélérer la convergence et donc diminuer le nombre d'itérations et, en conséquence, le temps de calcul, mais aussi on aimerait augmenter le plus possible le facteur d'amortissement pour que la mesure de l'importance effectuée par PageRank soit le plus significatif (correct) possible. Il faut donc chercher un équilibre entre les deux éléments pour trouver une valeur du facteur d'amortissement optimal.
- En réalité, le choix du facteur d'amortissement dépend de l'objectif recherché avec l'algorithme: si l'on priorise la vitesse, on utilisera un facteur d'amortissement faible, tandis que si l'on priorise la fiabilité, un facteur d'amortissement proche de 1 sera préférable. Dans leur article original, Page et Brin avaient effectué leurs expériences avec $\alpha = 0.85$, forçant le promeneur à ignorer les liens de la page où il se trouvait 3 fois sur 20 [2]. Ils ont été satisfaits des résultats obtenus, et ce facteur d'amortissement a été utilisé par Google pendant longtemps, bien qu'ils ne le communiquent plus publiquement aujourd'hui.

4.2 Points faibles et limites de l'algorithme

1. Utilisation de l'importance plutôt que la pertinence:

Google, pour profiter des avantages de l'indépendance de la requête, a décidé de faire en sorte que PageRank mesure l'importance d'une page en fonction de la quantité et de la qualité des liens qui y pointent. Cependant, cela ignore la pertinence des pages par rapport à une requête spécifique. Cela pose un problème, car une page peut être très importante dans le Web global sans être vraiment pertinente pour une requête particulière, et vice-versa. Bien que ce soit un problème spécifique à PageRank, il s'agit en réalité d'un problème plus général lié au fait de centrer un algorithme uniquement sur l'importance, tout en ignorant la pertinence.

2. Mise à jour:

Comme mentionné dans les sections précédentes, le Web est en constante évolution, ce qui oblige Google à actualiser régulièrement le PageRank. Cependant, cela n'est pas une tâche facile en raison de la complexité des calculs pour une matrice de la taille du Web. Google doit donc décider à quelle fréquence recalculer son PageRank, car, en effet, à chaque recalculation, l'algorithme doit être exécuté depuis le début (bien qu'il y ait des recherches sur ce domaine, la réutilisation du vecteur PageRank précédent ne donne pas encore de bons résultats [1]). Si Google choisit de

recalculer le PageRank trop fréquemment, il pourrait gaspiller des ressources. En revanche, s'il ne le fait pas suffisamment souvent, il risque de fournir des résultats obsolètes aux utilisateurs, car le PageRank ne serait plus représentatif de l'importance des pages dans l'état actuel du Web.

Google a mentionné qu'il calcule le PageRank environ une fois toutes les quelques semaines pour l'ensemble des documents de sa collection Web. Cependant, des changements significatifs dans la structure du Web continuent de se produire, ce qui reste un dilemme actuel pour les ingénieurs de Google : combien de temps peut-on retarder l'énorme effort nécessaire pour une mise à jour complète avant que les résultats de Google ne deviennent trop obsolètes? [1]

3. Impartialité:

Comme on l'a vu dans la Section 3.2, Google utilise un vecteur de personnalisation au lieu de la matrice de perturbation uniforme. Cela a pour objectif de manipuler les probabilités de téléportation vers certaines pages et ainsi favoriser certaines entreprises, qui payent sûrement des sommes considérables à Google pour bénéficier de cette avantage. Cela donne à Google un contrôle presque total sur le PageRank des pages. En effet, si Google voulait augmenter le PageRank d'une page spécifique, il lui suffirait d'augmenter la probabilité de téléportation associée à cette page dans le vecteur de personnalisation. Payer une fortune à Google pour qu'il procède ainsi est très attractif pour les entreprises, car cela signifie que leur page apparaîtra plus souvent en tête des résultats, attirant ainsi un plus grand nombre de visiteurs, ce qui se traduit généralement par une augmentation des ventes et des revenus. Cette possibilité de manipulation du PageRank a déclenché de nombreuses discussions sur l'éthique de cette pratique, qui reste un sujet d'actualité. Certains affirment que cette méthode crée un conflit d'intérêts et compromet l'objectivité des résultats de recherche, car les pages les mieux positionnées ne sont pas uniquement celles qui sont les plus importantes, mais aussi celles qui bénéficient à Google économiquement. Cela mène à la question de si les résultats affichés par Google sont vraiment impartiaux, étant donné des enjeux économiques cachés qu'il y a derrière.

3. Manipulation:

Même si le classement généré par PageRank ne dépend pas uniquement du nombre d'hyperliens pointant vers une page, ils ont une très grande influence sur celui-ci. De plus, les liens pointant vers une page sont encore plus importants s'ils proviennent d'une page considérée comme importante. Cela a donné début à une stratégie de "vente de liens" [3] dans laquelle des pages importantes offraient aux propriétaires d'autres pages de mettre un lien vers leur page en échange d'une somme d'argent, souvent assez grande. Lorsque Google a appris de cette pratique, il a immédiatement menacé d'ignorer dans le calcul du PageRank les pages qui seraient découvertes en train de vendre des liens. [3] Cela, bien qu'a diminué la popularité de cette pratique, a empêché aux pages Web de faire de la publicité dans leur page en mettant des liens d'entreprises qui payaient un prix pour y apparaître, ce qui était tout à fait légitime. En effet, différentier entre une personne faisant de la vente de liens et une autre faisant de la publicité était très compliqué. Pour cela, Google a introduit l'attribut HTML *nofollow* aux balises de *<a>* (les hyperliens). Cette balise permettait de mettre des hyperliens dans une page HTML qui ne seraient pas pris en compte dans le calcul de PageRank (ils ne seront pas considérés comme des "votes"). De cette façon, Google a indiqué aux propriétaires des pages de utiliser l'attribut *nofollow* sur les liens payants (publicité). De cette façon, Google permettait la publicité mais non la vente de liens, qui avait comme objectif d'augmenter le PageRank d'une page.

```
1 <a href="http://www.example.com/" rel="nofollow">Visitez Exemple</a>
```

Listing 1: Exemple d'utilisation de l'attribut 'nofollow'

La manipulation du PageRank continue à être une préoccupation d'actualité pour Google. En effet, une grande partie des efforts dédiés à l'amélioration de leur moteur de recherche est dirigée vers la vente de liens et d'autres méthodes de manipulation de PageRank comme le *spam* (propriétaires de pages qui introduisaient une grande quantité de fois le lien vers leur page dans les

sections des commentaires d'une autre page style blog). [5] Une autre façon de mitiger l'impact de la vente de liens dans le classement d'une page, est l'utilisation de plusieurs algorithmes que Google utilise pour afficher les pages. Cette diversification de critères, permet de réduire les probabilités de réussite d'une page qui ait payé pour des liens à d'autres pages. Cependant, Google n'a pas encore gagné cette guerre contre la manipulation. En effet, des experts assurent que toutes ces pratiques ne sont devenues que "sousterraines" (cachées) et que leur quantité c'est multiplié depuis les dernières années. [8] Google continue à faire activement des progrès et de nouvelles méthodes pour éviter le plus possible la manipulation afin d'assurer la meilleure qualité de résultats aux utilisateurs, comme le *Penguin*: filtre appliqué aux résultats d'une recherche qui élimine les pages suspects de vente de liens ou manipulation. Cependant, de l'autre côté, les méthodes de manipulation de PageRank ne font que s'améliorer, ce qui oblige à Google de faire des progrès cachés de façon privée. En effet, google ne communique plus les critères qu'ils utilisent pour déterminer l'ordre d'affichage des pages ou toute autre information qui pourrait aider à comprendre comment manipuler ceci.

4.3 Points forts de l'algorithme

1. Utilisation de l'importance plutôt que la pertinence:

Même si cette pratique a des problèmes (Sections 4.2), considérer l'importance au lieu de la pertinence des pages a aussi des avantages. Comme mentionné dans les sections précédentes, se centrer sur l'importance au lieu que sur la pertinence rend PageRank indépendant des requêtes. Cela est très importants car ça permet à Google de calculer PageRank une seule fois et non à chaque requête. Cela se traduit par un affichage presque instantané des pages lors de la requête de l'utilisateur, et aussi par des économies en termes d'argent (puisque calculer PageRank consomme une quantité énorme de ressources que Google doit payer). Cette caractéristique est celle qui a provoqué le succès de Google avec PageRank, puisqu'ils résolvaient le principal problème des autres méthodes de Récupération d'Information de l'époque qui se concentraient sur la pertinence (comme HITS [1]).

2. Immunité partielle à la manipulation:

Bien que c'est possible de manipuler le PageRank d'une page comme mentionné dans la section précédente, il y a des études qui affirment que l'influence d'avoir quelques liens de plus pointant vers une page dans le PageRank n'est pas significatif à cause de l'immense quantité de pages dans le Web [1]. Donc, si bien c'est possible, la manipulation est dans la grande majorité des cas inutile. De même, le fait que Google n'affiche pas le PageRank des pages rend très difficile de savoir qu'elles pages sont celles qui sont vraiment importantes, donc ceux desquelles il vaudrait peut-être la peine d'acheter des liens. La majorité des cas où quelqu'un paye à une page pour mettre le lien de sa page, il sait même pas si la page est suffisamment importante (PageRank haut) pour que ce lien ait un minimum d'impact dans le PageRank de sa page.

3. Personnalisation:

Si bien l'utilisation du vecteur de personnalisation est controversée, il est aussi utile pour Google pour combattre la manipulation. En effet, si Google veut pénaliser une page qui est suspecte de vendre des liens, il doit uniquement descendre (même mettre à 0) la probabilité associée à cette page dans le vecteur de personnalisation. En effet, cela éliminera presque toute possibilité d'être téléporté à cette page, ce qui diminuera la proportion de temps que l'utilisateur qui se promène à l'infini passe dans cette page, ce qui diminuera alors son PageRank.

5 Aspects computationnels

5.1 Analyse computationnel de l'algorithme de base

Définition 7: La **complexité temporelle** d'un algorithme désigne la quantité de temps nécessaire pour exécuter cet algorithme en fonction de la taille de l'entrée. Elle est exprimée à l'aide de la notation Big-O (\mathcal{O}), qui indique la croissance du temps d'exécution lorsque la taille des données augmente. Par exemple, $\mathcal{O}(n)$ signifie que le temps d'exécution augmente proportionnellement à la taille des données (on dit complexité linéaire), tandis que $\mathcal{O}(n^2)$ signifie qu'il augmente beaucoup plus rapidement (de façon quadratique).

Pour un problème de cette envergure, la complexité de l'algorithme est extrêmement importante à cause des immense tailles des entrées. Dans ce cas, l'entrée n est la dimension de la matrice des probabilités de transition, c'est à dire le nombre de pages dans le Web. Pour trouver la complexité de PageRank et pouvoir l'analyser, on a implémenté l'algorithme en Python:

```
1 import numpy as np
2
3 # Exemple du graphe correspondant a celui de la Figure 1
4 graphFigure1 = {"1": [2,3], "2": [3,4,5], "3": [1,2], "4": [3], "5": []}
5
6 """
7 Generation de la matrice des probabilites de transition d'un graphe donne.
8 La matrice retournee est stochastique.
9
10 Parametres:
11 - graph: Graphe representant le Web.
12 """
13 def initTransitionsMatrix(graph):
14     dim = len(graph) # Nombre de noeuds (pages web)
15     matrix = np.zeros((dim, dim), dtype=np.float128) # Matrice des probabilites de
16     transition
17     for i in range(len(graph)):
18         links = graph[str(i+1)] # Liens sortant de la page
19         line = matrix[i] # Ligne correspondante a la page dans la matrice
20         # Ajout d'un lien vers toute les pages si
21         # la page actuelle n'a pas de liens sortants
22         if len(links) == 0:
23             line += 1/dim
24         else:
25             # Insertion des transitions dans la matrice
26             for link in links:
27                 line[link-1] = 1 / len(links) # Normalisation des probabilites
28     return matrix
29
30 """
31 Methode de la puissance pour calculer le vecteur des probabilites d'équilibre
32 (stationnaires) d'une matrice des probabilites de transition.
33
34 Parametres:
35 - transitionsMatrix: Matrice des probabilites de transition.
36 - iterations: Nombre d'iterations maximal.
37 - tolerance: Seuil de tolerance pour assumer la convergence.
38 """
39 def powerMethod(transitionsMatrix, iterations=50, tolerance=1e-6):
40     P = transitionsMatrix
41     dim = len(P)
42     pi = np.ones(dim) * 1/dim # Initialisation uniforme du vecteur stationnaire
43     for _ in range(iterations):
44         pi_next = pi @ P # Nouveau vecteur calcule (vecteur propre)
45         # Verification de variation significative (convergence) du vecteur propre.
46         if np.linalg.norm(pi_next - pi) < tolerance:
47             break
48         pi = pi_next # Actualisation du vecteur stationnaire
49     return pi
50
51
52
```

```

53 """
54     Algorithme PageRank qui donne le score de relevance de chaque page d'un graphe.
55
56     Parametres:
57     - graph: Graphe representant le Web.
58     - dampingFactor: Facteur d'amortissement (probabilite de teleportement).
59     """
60
61 def pageRank(graph, dampingFactor=0.85):
62     transitionsMatrix = initTransitionsMatrix(graph)
63     dim = len(transitionsMatrix)
64     E = np.ones((dim, dim)) * 1/dim # Matrice de perturbation
65     # Rendre irreductible la matrice des probabilites de transition
66     P = dampingFactor * transitionsMatrix + (1-dampingFactor) * E
67     # Calcul du vecteur des probabilites d'equilibre (stationnaires)
68     pi = powerMethod(P, iterations=50, tolerance=1e-6)
69     return pi
70

```

Listing 2: Implémentation de l'algorithme PageRank de base

Analysons maintenant la complexité temporelle et quelques propriétés de l'algorithme à l'aide de graphiques générés en utilisant cette implémentation. (Voir Section 7 pour accéder au code source utilisé pour générer les différents graphiques)

1. Complexité temporelle de PageRank:

En analysant le code, on peut voir que PageRank se réduit effectivement à des itérations d'une multiplication vecteur-matrice. La multiplication d'un vecteur de taille n par une matrice $n \times n$ consiste à calculer le produit scalaire entre le vecteur et chaque ligne de la matrice. Chaque produit scalaire nécessite n multiplications, et il y a n lignes dans la matrice. Ainsi, le total des opérations est $n \cdot n$, ce qui donne une complexité de $\mathcal{O}(n^2)$. Cette opération, étant la principale de l'algorithme, est répétée pendant un certain nombre d'itérations par la méthode de la puissance. En général, la complexité de la méthode de la puissance serait $\mathcal{O}(k \cdot n^2)$, où k est le nombre d'itérations. Cependant, puisque la méthode de la puissance est assurée de converger (Théorème 1), alors k sera toujours un entier généralement inférieur à 50, donc il peut être exclut de la notation car n^2 sera toujours le terme dominant. Avec ça, on pourrait s'attendre que le graphique du temps d'exécution de notre implémentation de PageRank, et celle de la méthode de la puissance, ait la forme d'une courbe quadratique.

Générons d'abord le graphe représentant le temps d'exécution de la méthode de la puissance en fonction du nombre de noeuds dans le graphe. Pour la génération de ce graphique, chaque noeud sera connecté à un tiers des noeuds du graphe:

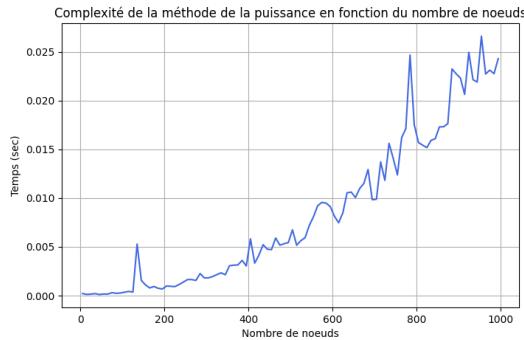


Figure 4: Graphique de la complexité de la méthode de la puissance en fonction du nombre de noeuds

On constate que, effectivement, la courbe a une forme quadratique, ce qui confirme que la méthode de la puissance a une complexité de $\mathcal{O}(n^2)$.

Faisant de même, mais maintenant avec l'algorithme PageRank au complet. Pour suivre la notion commentée dans la Section 4.1, dans ce cas chaque noeud sera connecté uniquement à 10 autres noeuds peu importe le nombre total de noeuds dans le graphe:

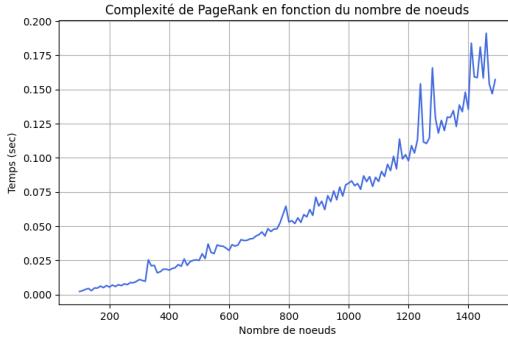


Figure 5: Graphique de la complexité de PageRank en fonction du nombre de noeuds

On peut voir de même que la courbe a une forme polynomiale, ce qui pourrait confirmer notre théorie sur la complexité, cependant on a remarqué quelque chose très intéressante.

Voici un graphe qui montre le temps d'exécution de PageRank pour un graphe de 500 noeuds en fonction du nombre d'arêtes total dans le graphe:

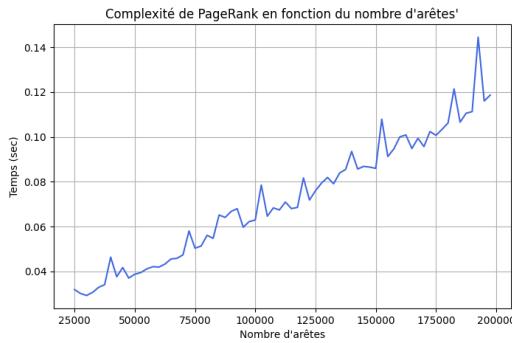


Figure 6: Graphique de la complexité de PageRank en fonction du nombre d'arêtes

On remarque que c'est une fonction linéaire, alors le nombre d'arêtes influence aussi de façon linéaire la complexité de PageRank.

On peut donc s'attendre que la complexité de PageRank soit $\mathcal{O}(k \cdot n^2 + E)$, qui peut être aussi simplifiée à $\mathcal{O}(n^2 + E)$, où n représente le nombre de noeuds dans le graphe (nombre de pages dans le Web) et E représente le nombre d'arêtes dans le graphe (nombre total d'hyperliens dans le Web). Cela pour un problème de la taille de PageRank est énorme. Effectivement, le temps nécessaire à Google pour calculer le vecteur PageRank a été rapporté comme étant de l'ordre de plusieurs jours. [1]

2. Influence du seuil de tolérance dans la précision de la méthode de la puissance:

Dans les sections précédentes on a parlé de la méthode de la puissance et de sa fiabilité. On aimerait savoir si en appliquant un seuil de tolérance de 1×10^{-6} , on est en train de perdre des variations qui pourraient être importantes contre la vraie convergence du vecteur propre (vecteur des probabilités stationnaires). Pour faire cela, on va utiliser l'exemple de [2] où ils nous donnent le vrai (théorique) vecteur des probabilités stationnaires. On va mesurer l'évolution de la vrai erreur entre le vrai vecteur des probabilités stationnaire et celui approximé par la méthode de la puissance à chaque itération. De l'autre côté, on va mesurer aussi sur le même graphe, la variation (différence) entre le vecteur approximé par la méthode de la puissance à l'itération actuelle et celui calculé à l'itération précédente. On a éliminé le seuil de tolérance pour pouvoir regarder les variations pendant les 50 premières itérations.

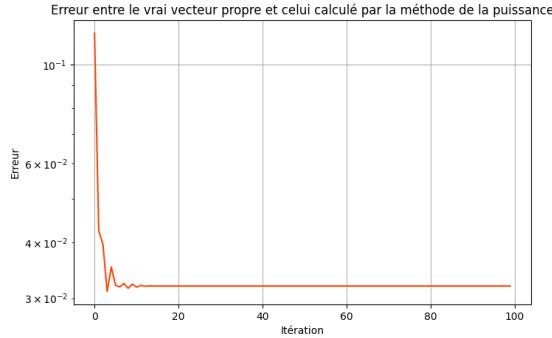


Figure 7: Graphique des variations entre le vecteur propre actuel et celui de l'itération précédente

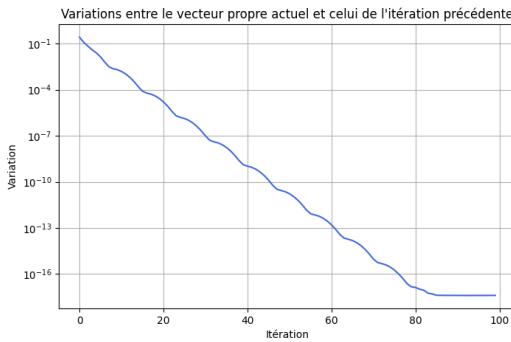


Figure 8: Graphique de la complexité de la méthode de la puissance en fonction du nombre de noeuds

On remarque que même si les variations entre les valeurs du vecteur propre continuent à descendre (le vecteur propre n'a pas encore convergé complètement), l'erreur entre le vrai vecteur propre et celui approximé arrive à son minimum (converge) bien avant et ne continue pas à diminuer. Cela est parce que les variations au delà des itérations de la convergence de la vraie erreur sont très petites, donc la différence entre le vrai vecteur propre et celui approximé par la méthode de la puissance est presque nulle. Pour cette raison, on peut voir que un seuil de tolérance de 1×10^{-6} pour accepter la convergence est plus que suffisant et n'affecte pas la précision de la méthode de la puissance. En effet, l'erreur entre le vrai vecteur propre et celui calculé par la méthode de la puissance converge bien avant que la variation entre deux itérations de la méthode de la puissance soit inférieure à 1×10^{-6} .

On aimerait juste expliquer qu'on a choisi de vérifier la convergence avec la différence entre les vecteur propres à chaque itération et non avec les valeurs propres car généralement la convergence du vecteur propre est plus rapide que celle de la valeur propre et, en plus, c'est le résultat qui nous intéresse. Lorsqu'on fait la convergence avec la valeur propre, le résultat final est le même, mais la méthode de la puissance risque de faire plus d'itérations, ce qui n'est pas désirable.

Finalement, on trouve intéressant ce graphique généré qui montre la convergence individuelle de chaque valeur du vecteur des probabilités stationnaires d'un graphe de 5 noeuds en fonction du nombre d'itérations de la méthode de la puissance (avec le seuil de tolérance éliminé, on fait 25 itérations):

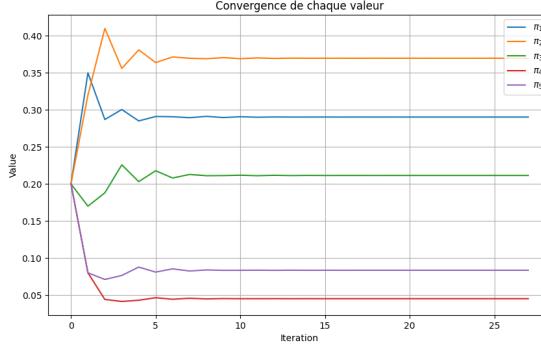


Figure 9: Graphique de la complexité de la méthode de la puissance en fonction du nombre de noeuds

3. Taux de convergence de la méthode de la puissance:

On a déjà parlé du taux de convergence de la méthode de la puissance et comment il est influencé par la valeur du facteur d'amortissement. Pour visualisé cet effet, on a générée un graphique qui montre l'itération à laquelle la méthode de la puissance s'arrête (avec un seuil de tolérance de 1×10^{-6}) pour un même graphe en fonction de la valeur du facteur d'amortissement:

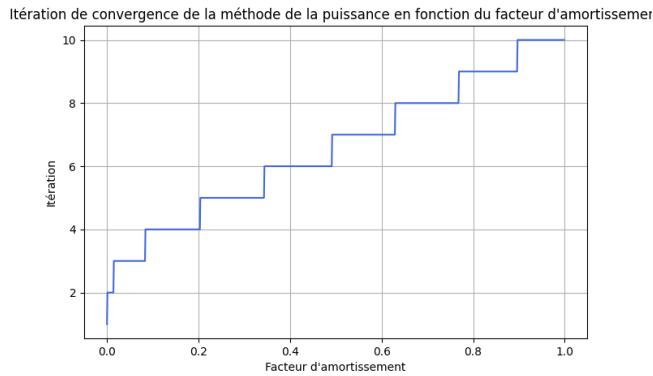


Figure 10: Graphique de l'itération de convergence de la méthode de la puissance en fonction du facteur d'amortissement

Cela supporte la théorie de laquelle on parlait dans la Section 4.1. Effectivement, le nombre d'itérations que la méthode de la puissance va effectuer avant d'accepter la convergence augmente linéairement en fonction de la valeur du facteur d'amortissement.

On aimerait savoir si le taux de convergence est influencé par un autre facteur à part le facteur d'amortissement. Pour cela, on a donc générée un graphique qui montre l'itération à laquelle la méthode de la puissance s'arrête (avec un seuil de tolérance de 1×10^{-6} et un facteur d'amortissement $\alpha = 0.85$) en fonction du nombre de noeuds dans un graphe:

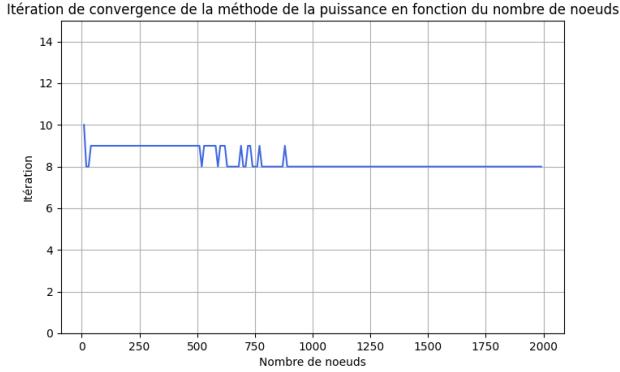


Figure 11: Graphique de l'itération de convergence de la méthode de la puissance en fonction du nombre de noeuds

On retrouve une fonction constante, ce qui veut dire que le taux de convergence de la méthode de la puissance est indépendant du nombre de pages dans le Web (nombre de noeuds dans le graphe). On peut donc conclure que le taux de convergence de la méthode de la puissance dépend uniquement de la valeur du facteur d'amortissement.

5.2 Optimisation: Multiplication de matrices creuses

On a vu dans la Section 4.1 que la matrice des probabilités de transition du Web est une matrice creuse. On peut donc implémenter une méthode de multiplication matricielle spéciale pour les matrices creuses qui consiste à ne multiplier que les éléments non nuls des matrices, ce qui permet de réduire les calculs et de gagner du temps (améliorer la complexité). Elle utilise des structures de données spéciales pour stocker uniquement les éléments non nuls, améliorant ainsi l'efficacité de l'opération. Voici son implémentation:

```

1 import numpy as np
2
3 """
4 Creation d'une matrice de transition creuse (sparse) qui garde uniquement
5 les probabilites non nuls du graphe (elements non nuls de la matrice de
6 transition classique).
7 """
8 def init_sparse_transition_matrix(graph):
9     dim = len(graph)
10    # Noeud: [transitions] --> transition = (destination, probabilite)
11    matrix = {i: [] for i in range(1, dim+1)}
12    for i, (node, links) in enumerate(graph.items()):
13        if len(links) == 0:
14            # Etat absorbant -> lien vers toutes les pages
15            for j in range(dim):
16                transition = (j, 1/dim)
17                matrix[i+1].append(transition)
18        else:
19            proba = 1/len(links)
20            for link in links:
21                matrix[i+1].append((int(link), proba))
22    return matrix
23
24
25 """
26 Methode de la puissance avec matrice creuse.
27 Multiplication matricielle realisee en O(N).
28 """
29
30 def power_method_sparse(matrix, dampingFactor, iterations=50, tolerance=1e-6):

```

```

31     dim = len(matrix)
32     pi = np.ones(dim) / dim # Vecteur initial uniforme
33     for _ in range(iterations):
34         next_pi = np.zeros(dim)
35         # Multiplication: Iterer uniquement sur les valeurs non nulles
36         # stockees dans la matrice creuse.
37         for i, edges in matrix.items():
38             for j, prob in edges:
39                 # Ajouter la contribution de la page actuelle
40                 next_pi[j-1] += dampingFactor * pi[i-1] * prob
41             # Appliquer le facteur de teleportation a chaque element de pi
42             # (Avec une probabilite 1-dampingFactor, le surfeur saute vers une page
43             aleatoire)
44             teleportation = (1 - dampingFactor) / dim
45             next_pi += teleportation
46             # Vérifier la convergence du vecteur propre
47             if np.linalg.norm(next_pi - pi, 1) < tolerance:
48                 break
49             pi = next_pi
50     return pi
51
52 """
53 Algorithme PageRank optimise les matrices creuses.
54 Lequel est le cas du Web dans la réalité.
55 """
56
57 def pageRank_sparse(graph, dampingFactor=0.85):
58     # Creer la matrice sparse
59     sparse_matrix = init_sparse_transition_matrix(graph)
60     # Calculer le vecteur PageRank (probabilites stationnaires)
61     pi = power_method_sparse(sparse_matrix, dampingFactor)
62     return pi
63

```

Listing 3: Implémentation de l'algorithme PageRank optimisé pour les matrices creuses

Générons alors le graphique du temps d'exécutions de cette implémentation de PageRank optimisé pour les matrices creuses:

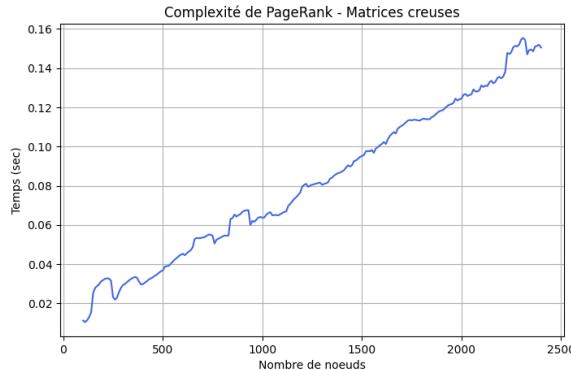


Figure 12: Graphique de la complexité de PageRank optimisé pour les matrices creuses

On constate que le graphique est linéaire, donc cette méthode multiplication pour les matrices creuses est réalisée en temps linéaire $\mathcal{O}(n)$. Donc, cette méthode optimise l'implémentation de PageRank pour la faire de complexité $\mathcal{O}(n + E)$.

6 Conclusion

6.1 Impact

L'algorithme PageRank a été un acteur principal dans l'élévation de Google comme le moteur de recherche le plus utilisé à travers le monde. En effet, c'est principalement cet algorithme qui a permis à Google de se démarquer parmi les premiers moteurs de recherches. Google ayant été déployé officiellement en 1998 [19], il peut sembler surprenant qu'il ait été plus populaire que Yahoo!, qui a été déployé 4 ans plus tôt, en 1994. C'est effectivement l'algorithme PageRank qui a permis à Google de dépasser Yahoo! en termes d'utilisateur, seulement parce que cet algorithme était meilleur que celui utilisé par Yahoo! pour classer les pages.

PageRank n'a pas seulement eu un impact positif sur Google, mais aussi sur l'Internet tout entier. En effet, pendant le début des années 2000, la proportion de la population utilisant Internet a grandement augmenté, majoritairement grâce à Google. La preuve: le terme le plus utilisé qui signifie "faire une recherche sur Internet" est le verbe "Googler".

L'algorithme PageRank est aussi plus qu'un simple algorithme de classement de pages Web, puisqu'il peut être généralisé à un algorithme d'analyse de lien qui assigne un poids numérique à chaque élément d'un ensemble de documents hyperliés, dans le but de mesurer l'importance relative de ces documents au sein de l'ensemble. L'algorithme peut donc être appliqué sur n'importe quel collection d'entités qui se réfèrent entre elles [3 (Wikipedia)]. Avec cette définition, on peut conclure que PageRank peut avoir beaucoup d'autres applications en dehors du Web.

6.2 Applications

PageRank étant un algorithme relativement simple, il est facile de le modifier aux besoins de l'application pour laquelle on veut l'utiliser. En effet, la versatilité de PageRank fait en sorte qu'il puisse être adapté pour être appliquer dans plusieurs domaines. Il peut même être complexifié dans le cas où le problème auquel on veut appliquer l'algorithme est plus complexe que le classement de pages Web.

Les mathématiques de PageRank étant générales, il est possible de les appliquer à plusieurs domaines: la bibliométrie, l'analyse de réseau sociaux et informationnelle, la prédiction ou recommandation de liens, etc. Les domaines de la science qui bénéficient de cet algorithme sont multiples: la biologie, la chimie, la neuroscience, la physique, les travaux routiers, etc.

Par exemple, PageRank est utilisé autrement sur Internet que pour classifier des pages Web. Le réseau social Twitter (maintenant X) utilise PageRank pour faire des recommandations de comptes qui pourraient intéressé les utilisateurs. Dans cette application de l'algorithme le graphe est décrit comme-ci: les noeuds sont les utilisateurs et les arêtes sont les mentions ("*tags*") entre les utilisateurs. Par exemple, un utilisateur *A* qui mentionne l'utilisateur *B* dans une publication constitue une arête du noeud *A* au noeud *B*.

PageRank a aussi diverses applications dans la recherche scientifique. En neuroscience, les chercheurs ont trouvé une corrélation entre le PageRank d'un neurone dans un réseau de neurones et sa fréquence de décharge relative ("*firing rate*"). Cela a de grandes implications puisque cela permet de mieux comprendre comment fonctionne le cerveau humain, un problème fondamental dans plusieurs disciplines, notamment l'intelligence artificielle qui est très d'actualité en ce moment.

En biologie, les auteurs Allesina et Pascual ont démontré dans un papier en 2009 que PageRank pouvait être utilisé pour déterminé l'importance de certaines espèces dans la chaîne alimentaire complexe d'un écosystème [5]. Un graphe alimentaire d'un écosystème peut être construit en prenant les espèces comme noeuds et les arêtes comme le transfert d'énergie d'une espèce à l'autre, la plupart du temps en se faisant manger. Un exemple pourrait être une espèce de renard et une espèce de lièvre comme noeuds. L'arête serait dirigée du lièvre vers le renard (Lièvre → Renard), puisque c'est le lièvre

qui donne de l'énergie au renard en se faisant manger. Une différence important avec le PageRank utilisé pour les pages Web est que ici l'importance est inversée, c'est-à-dire que une espèce est importante si elle pointe vers, ou en d'autres mots est mangée par, une autre espèce importante. Cette distinction illustre la capacité de pouvoir modifier PageRank selon l'application qu'on veut en faire.

Dans les sports, PageRank est utilisé pour classifier la performance des différents acteurs. Par exemple, la Ligue Nationale de Football Américain (*NFL*) utilise l'algorithme pour classifier les équipes de la ligue. L'algorithme est aussi utilisé pour classifier la performance des joueurs de soccer individuels à travers différentes ligues. Finalement, la *Diamond League*, une ligue de compétitions d'athlétisme, utilise PageRank pour classifier leurs athlètes.

Les administrations de différentes villes dans le monde utilisent aussi PageRank pour classifier des espaces ou des rues, dans le but de prédire combien d'acteurs (piétons, cyclistes ou véhicules) empruntent la rue ou l'espace concernée [3].

On peut donc en conclure que l'algorithme de Google a de nombreuses applications et qu'il ne cesse d'en avoir des nouvelles, peu importe le domaine.

6.3 Futur

Google ayant apportés de nombreuses modifications à leur moteur de recherche dans le but d'obtenir de meilleurs résultats, PageRank ne joue plus un rôle aussi important qu'au début de son utilisation. De nos jours, le domaine du *SEO (Search Engine Optimisation)* se concentre sur d'autres aspects pour tenter d'obtenir des meilleurs résultats. Cependant, l'existence des liens entrants de pages populaires sur une page continue d'augmenter le rang d'une page lorsqu'on effectue une recherche sur Google.

Pour optimiser les résultats d'une requête sur les moteurs de recherches, voici une liste de technologies qui seront probablement utilisées dans le futur:

- Les avancées rapides de l'intelligence artificielle seront très probablement intégrés à la recherche Web. En effet, on peut déjà observé l'onglet "*AI Summary*" lorsqu'on fait une recherche sur Google, preuve de l'intégration de l'IA. De plus, les modèles de langage large (*LLMs*) seront très certainement utilisés pour faire le pont entre les recherches sur le Web. Une simple question d'un utilisateur sera probablement répondue par un *LLM* plutôt que par une page Wikipédia.
- La personnalisation de l'expérience sur Internet jouera aussi probablement un rôle important dans le futur des recherches sur le Web. Les grandes compagnies technologiques comme Meta collectent constamment de l'information sur leurs utilisateurs, au point où les publicités qui nous sont proposés sont personnalisés. Les recherches sur le Web subiront probablement le même sort, c'est-à-dire que le moteur de recherche connaîtra les préférences de l'utilisateur afin de lui faire des recommandations pertinentes.
- L'opinion des utilisateurs: avec l'addition de fonctionnalités basés sur ce que pensent les utilisateurs comme les "*Community Notes*" de X ou d'Instagram, il est probable que la pertinence perçue d'un résultat d'une recherche d'un utilisateur sera considéré pour un autre utilisateur qui fait une recherche similaire.

Sachant cependant que Google garde précieusement les secrets de leur algorithme de classement des pages, il sera difficile de savoir qu'est-ce qui influence réellement les résultats obtenus à la suite d'une requête.

6.4 Réflexions

L'histoire de PageRank est un bel exemple de comment un problème complexe peut parfois avoir une solution mathématique simple et élégante. Il est motivant de se dire qu'il est possible, en combinant

un problème réel et des concepts mathématiques découvert il y a longtemps, de trouver des solutions aux problèmes de la vie de tous les jours.

Après avoir fait ce travail, nous regarderons désormais de près la progression des moteurs de recherche pour voir si un nouvel algorithme émergera et détrônera PageRank.

7 Bibliographie

Tous les graphiques et illustrations ont été réalisés par nous mêmes.

- *Tous graphiques ont été générés en Python à l'aide de la librairie **Matplotlib**. Le source code de leur implémentation est disponible dans le dépôt Github suivant: <https://github.com/medinamartin3/PageRank>.*
- *Toutes les illustrations du graphe orienté ont été réalisé avec l'outil en ligne suivant: <https://graphonline.top/>.*

[1] Langville, A. N., & Meyer, C. D. (2005). **A survey of eigenvector methods for web information retrieval**. SIAM Review, 47(1), p.135–p.161. Retrieved from: [Link](#).

Commentaires: Très clair et très bien écrits, mais les concepts ne sont pas abordés en profondeur. De plus, la source date de 2005 et donc quelques informations ne sont plus d'actualités.

[2] Rousseau, C., & Saint-Aubin, Y. (2008). **Mathématiques et technologie**. Springer. ISBN: 978-0-387-69212-8, p.273–p.296.

Commentaires: Très superficiel, parle de beaucoup de choses mais n'explique pas pourquoi elles sont importantes. Assez incomplet, c'est une bonne introduction mais il manque beaucoup d'informations permettant de bien comprendre la totalité du thème.

[3] Wikipedia contributors. (n.d.). **PageRank**. *Wikipedia*. Retrieved from: [Link](#).

Commentaires: Les réflexions des auteurs sont très intéressantes et les sujets abordés sont d'actualité, mais les mathématiques utilisées et leur ordre d'apparition ne permettent pas de bien expliquer comment fonctionne.

[4] Amrani, A. (2020, December 19). **PageRank algorithm, fully explained**. *Towards Data Science*. Retrieved from: [Link](#).

Commentaires: Incomplet et très superficiel. Explique les concepts les plus importants mais sans profondeur, manque de sérieux.

[5] Moor, B. (2018, August). **Mathematics behind Google's PageRank algorithm**. (Master's thesis, Texas Woman's University). Retrieved from: [Link](#).

Commentaires: Très bien, la section dédiée à l'explication des mathématiques nous a grandement aidé à comprendre. Le fil conducteur de l'explication de l'algorithme est très bien réalisé.

[6] Ipsen, I., & Wills, R. **The mathematics behind Google's PageRank**. *North Carolina State University*. Retrieved from: [Link](#).

Commentaires: Pas très utile pour comprendre PageRank, mais extrêmement utile pour comprendre en profondeur la méthode de la puissance (*Power method*) et ses propriétés.

- [7] Brin, S., & Page, L. (1998). **The PageRank citation ranking: Bringing order to the web.** Retrieved from: [Link](#).

Commentaires: Utile pour comprendre les aspects non mathématiques de l'algorithme, mais très vieux et donc pas très utile pour les aspects mathématiques.

- [8] Verzhbitskaia, Z. (n.d.). **Le passé, le présent et l'avenir du PageRank de Google.** SEO PowerSuite Ltd. Retrieved from: [Link](#).

Commentaires: Très utile pour comprendre comment Google combat la manipulation de PageRank et la vente de liens dans le but d'obtenir un meilleur classement.

- [9] Tanguy. (n.d.). **Qu'est-ce que le PageRank de Google ? Ads-Up.** Retrieved from: [Link](#).

Commentaires: Peu d'informations utiles, très en surface.

- [10] Bonald, T. (2019, January). **PageRank.** Telecom ParisTech. Retrieved from: [Link](#).

Commentaires: Très complexe à lire, les mathématiques sont compliqués. Utile pour comprendre les améliorations possibles pouvant être apportées à PageRank.

- [11] Université d'Aix-Marseille. (n.d.). **Projet PageRank.** Licence de maths, Analyse numérique, semestre 5. Retrieved from: [Link](#).

Commentaires: Bonne introduction bien expliquée mais généralement incomplet puisqu'il s'agit d'un document académique que des étudiants devaient probablement complété.

- [12] Shum, K. (2013, March 4). **Notes on PageRank Algorithm** (Lecture 13). Hasso Plattner Institute. Retrieved from: [Link](#).

Commentaires: Bien structuré mais sans information unique que les sources précédentes n'ont pas déjà couverte.

- [13] Efimov, V. (2023, August 9). **Visualised Explanation of PageRank.** Towards Data Science. Retrieved from: [Link](#).

Commentaires: Très bien illustré, excellente introduction de l'algorithme. Cependant, aucune nouvelle information non couverte par les sources précédentes.

- [14] Chonny. (2021, January 8). **PageRank: Link Analysis Explanation and Python Implementation from Scratch.** Towards Data Science. Retrieved from: [Link](#).

Commentaires: Très utile pour valider que notre implémentation de l'algorithme était correcte.

- [15] L'Ecuyer, P. (n.d.). **Chaînes de Markov en temps discret.** Retrieved from: [Link](#).

Commentaires: Très complet et très utile mais vraiment complexe, permet de bien comprendre les Chaînes de Markov. Va très en profondeur dans les mathématiques, ce qui n'est pas nécessairement requis pour PageRank.

- [16] Arians, J. (n.d.). **The Google PageRank Algorithm**. Retrieved from: [Link](#).

Commentaires: Utilisé pour vérifier la complexité computationnelle de la multiplication matricielle et la multiplication de matrices creuses.

- [17] Churchill, A. (2021, May 7). **A Friendly Introduction to the PageRank Algorithm**. Medium. Retrieved from: [Link](#).

Commentaires: Bonne introduction facile à lire mais peu intéressant car couvre les sujets seulement en surface.

- [18] NJ. (2025, March). How many websites are there in the world? Sitefy. Retrieved from: [Link](#).

Commentaires: Utilisé pour savoir combien de page Web se trouve actuellement sur Internet et d'autres statistiques

- [19] Rose, C. (2023, January 9). The complete history of search engines. SEO Mechanic. Retrieved from: [Link](#).

Commentaires: Utilisé pour savoir l'histoire des moteurs de recherches et connaître les compétiteurs de Google et leurs algorithmes.