**RAPPORT DE TRAVAUX SUR LA MÉTHODE DE PAGE RANKING**

Diagram, shape

Description automatically generated

Logo, company name

Description automatically generated

Tomy Soumphonphakdy - Yann Cauchepin

IS5 - Polytech Lille - 07/10/2020

# Sommaire

[**Sommaire**](#_heading=h.6hdd7adwteyj) **1**

[**Introduction**](#_heading=h.nb2y6zlgh18k) **2**

[**Pagerank Classique**](#_heading=h.tu819c8l3waw) **3**

[Résultats](#_heading=h.tfqtdt3vt3ip) 3

[Analyse pour des ꞵ différents](#_heading=h.1z9aqoks78zk) 4

[Analyse de la valeur propre dominante](#_heading=h.80okg3gxwiqo) 7

[**Pagerank Personnalisé**](#_heading=h.8a83bto9nm0y) **8**

[Pagerank personnalisé sur 3 articles du top 20](#_heading=h.8ctn9qv92bdv) 8

[Pagerank personnalisé sur le dernier article](#_heading=h.90e9jzf1nvbm) 9

[Analyse de la valeur propre du noeud personnalisé](#_heading=h.3smaewwcvved) 10

[**Map-Reduce**](#_heading=h.j0e7ps7m5im3) **11**

[**Conclusion**](#_heading=h.vf9id625f4w4) **12**

# 

# Introduction

Nous avions pour objectif de coder un programme en C et un en R permettant d’utiliser la méthode de Pagerank et sa version personnalisée.

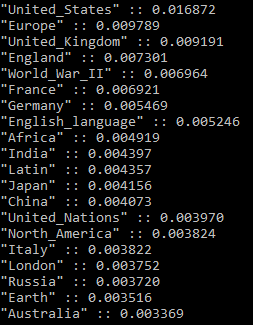
La méthode de Pagerank permet de classer des sites selon leur pertinence. Pour se faire, en fonction de l’existence de liens entre les sites, nous calculons une probabilité de se retrouver sur une page si l’on clique un grand nombre de fois aléatoirement sur des liens web. Pour ce faire, elle associe deux idées. L’une est de suivre, à l’instar d’une chaîne de Markov, les arcs du graphe pour chercher la mesure stationnaire. La seconde étant la probabilité de se retrouver “téléporté” sur un noeud afin d’éviter des effets de puits sur des noeuds isolés, ne renvoyant pas vers aucun autres noeuds, ou ne permettant pas de revenir sur certains sites.

Nous appliquons ce programme sur des fichiers au format tsv contenant plusieurs historiques de visites de pages Wikipedia. Ce jeu de données correspond à un jeu dont le but est de partir d’une page donnée et d’arriver à une autre le plus vite possible. Nos résultats dans ce rapport sont ceux renvoyés avec *paths\_finished.tsv*. Cela constitue un graphe de 4169 noeuds. Lorsque ꞵ n’est pas précisé, il vaut 0.85. De même pour la marge d’erreur qui vaut par défaut, dans ce document, 10-7.

# Pagerank Classique

## Résultats

Les vingt premiers articles sortis dans l’ordre décroissant par notre programme sont :



Cela signifie que *“United\_States”* est considérée comme étant la page la plus populaire de la base de donnée extraire du fichier *path\_finished.tsv*.

Ce résultat a demandé au programme 24 itérations afin d’obtenir la précision adéquate.

Nous observons que beaucoup d’articles renvoient vers une location comme un pays. Les pays apparaissant sont généralement les pays les plus riches/puissants. Leur implication dans la seconde guerre mondiale peut être une cause de la popularité de *“World\_War\_II”*.

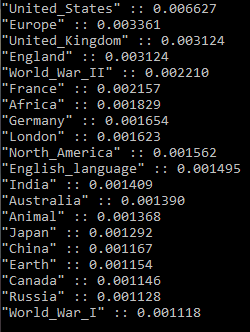
Nous en déduisons que ces articles sont aussi les plus simples d’accès, et donc avoir pour article d’arrivée l’un de ces noeuds dans le jeu rend la tâche plus aisée.

## Analyse pour des ꞵ différents



:

0.000240 correspond à 1/4169 soit le nombre de noeuds. Cela est dû au fait qu’importe l’itération, chaque page a pour valeur : qui est une constante. La normalisation ramène les valeurs à ce qui est bien en accord avec l’esprit de “téléportation” aléatoire sur chaque noeud.



:

Les valeurs sont moins disparates et moins élevées. La page au rang le plus bas est *"Yagan" :: 0.000192*, une valeur d’ordre similaire à 1/N. Cela signifie que beaucoup de pages ont des valeurs légèrement inférieures à 1/N tandis que les pages aux d’ordre supérieur sont peu nombreuses (24/4169 et au rang ~80 on a des valeurs proches de 0.000500). L’effet de téléportation causé par est peut-être encore trop présent.

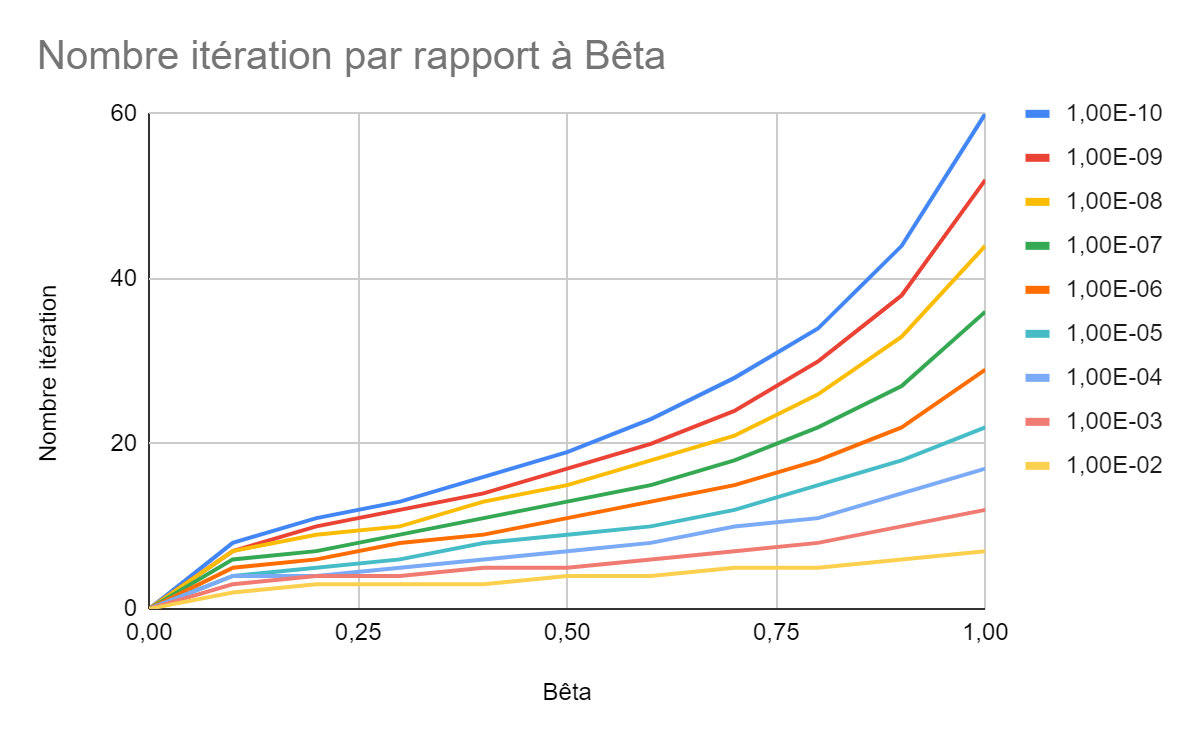


:

Ce cas est sujet aux effets de puits des noeuds/ensembles de noeuds ne permettant pas de revenir vers certains noeuds. En effet, nous dénombrons ~500 noeuds aux valeurs nulles et les premières valeurs passant la barre des 0.000100 sont au rang ~1800. Des articles classées plus bas pour un ꞵ de 0.85 tels que “*Christianity”*, *“Islam”* et *“European\_Union”* en profitent pour gagner en rang. Ils font donc probablement partie des noeuds/ensembles de noeuds où l’effet de puit est plus puissant.

Dans le cadre du jeu, il est possible de se dire que ces derniers sont donc moins intéressant à visiter car ils permettent moins d’accès vers d’autre noeuds. Et intuitivement, les pages cibles les plus simples sont celles au rang le plus élevé tandis que celles du bas de tableau ont une difficulté considérablement plus haute.

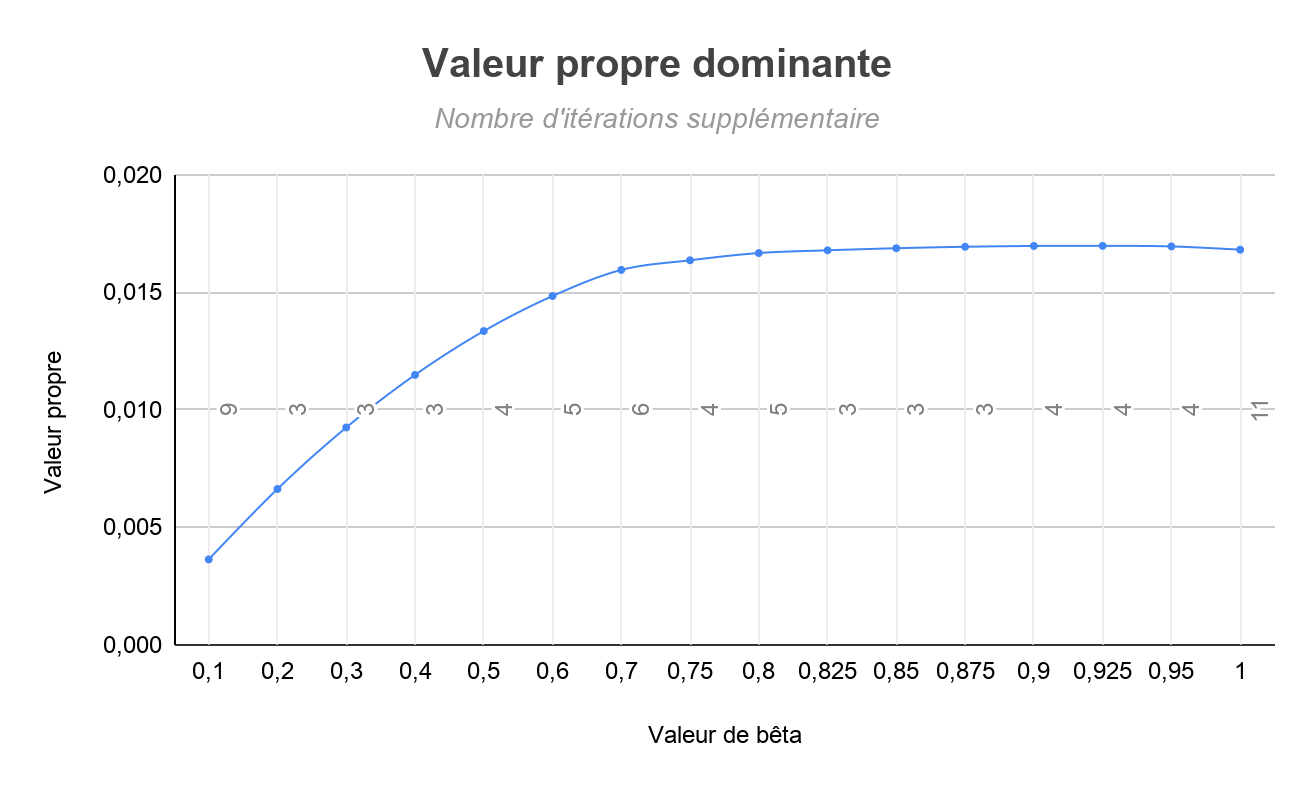
Hormis l’influence qu'a ꞵ sur les résultats, une augmentation de celui-ci nécessite au programme plus d’itérations pour une même précision.



Ce graphique confirme que l’augmentation de ꞵ requiert une augmentation d’itération mais on voit aussi que la diminution de la précision (x10-1 entre chaque courbe) influe aussi. Il est bon de remarquer que plus ce facteur est petit, plus la courbe s’éloigne d’une droite pour se rapprocher d’une courbe quadratique. Les écarts entre deux puissances de 10 semblent généralement égaux pour un ꞵ donné, ils sont logarithmiques.

## 

## Analyse de la valeur propre dominante



Ce schéma a été effectué avec un seuil de précision de 10-12.

On peut constater que la valeur propre dominante se stabilise pour une valeur de bêta supérieure à 0,85. De plus, pour un même écart du paramètre bêta, le nombre d’itérations effectué par l'algorithme augmente considérablement pour des valeurs de bêta élevées.

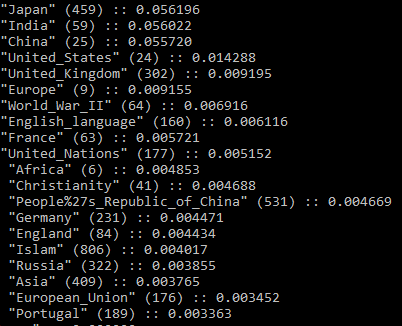
Il est donc recommandé de fixer une valeur de bêta proche de 0,85.

Fixer une valeur supérieure augmenterait la sensibilité du page ranking aux ensembles web qui ne sont pas connectés au reste du web. En simulant un parcours sur l’ensemble des sites web, les valeurs propres seraient trop bruités par ces ensembles.

Fixer une valeur inférieure diminuerait la précision des valeurs propres réelles de la matrice de transition du graphe web. Le page ranking ne serait donc pas efficace pour classer les sites web les plus pertinents.

# Pagerank Personnalisé

## Pagerank personnalisé sur 3 articles du top 20



Voici la sortie d’un Pagerank personnalisé sur les articles *”India”*, *“Japan”* et *“China”* qui étaient sans personnalisation aux rangs 10, 12 et 13. Leurs valeurs sont bien plus élevées que les autres désormais.

Des articles tels que *“People%27s\_Republic\_of\_China”*, *“Asia”* ou encore *“Islam”* ont gagné en rang mais sans pour autant que cela soit significatif et la plupart des pages bien classées sans personnalisation le restent également ici.

Nous remarquons aussi que la somme des trois valeurs dominantes correspondent approximativement à la somme des valeurs dominantes sans personnalisation et de puisque la téléportation des surfers se fait uniquement sur ces noeuds là.

## 

## Pagerank personnalisé sur le dernier article

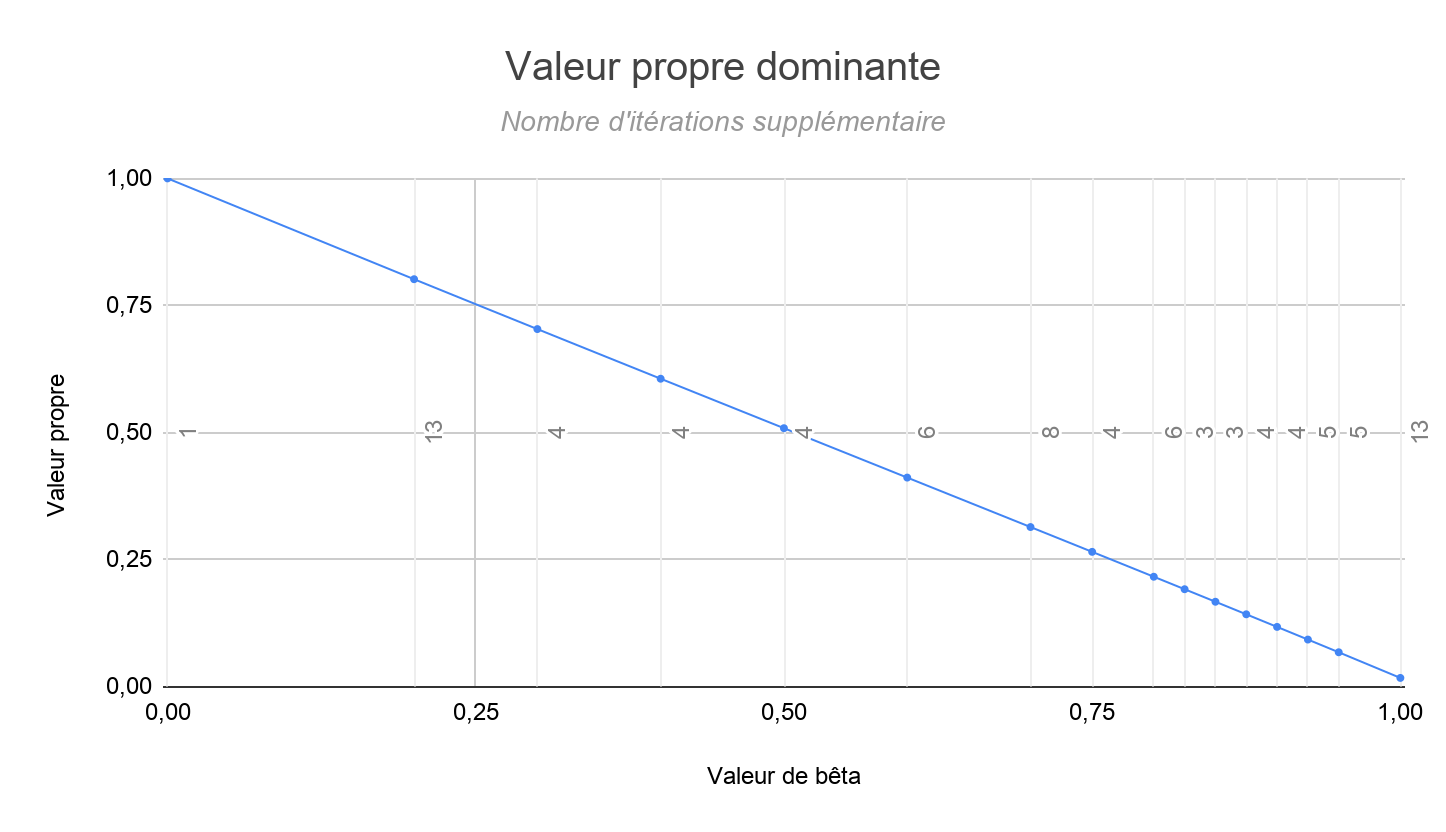


La page la moins populaire selon Pagerank classique serait *“Yagan”*, une langue parlée au Chili quasiment éteinte.

Le fait de forcer la navigation vers cet article (qui a une valeur encore une fois proche de ) fait apparaître dans le top des articles des noms auparavant moins bien classés comme *“Ancient\_Egypt”* [Rang 105] voire même *“Folklore”* [Rang 654].

## 

## Analyse de la valeur propre du noeud personnalisé



Ce schéma a été effectué en effectuant un pagerank personnalisé sur le site *“United\_States”*, le plus populaire du graphe web étudié, avec un seuil de précision de 10-12.

La valeur propre dominante chute de manière linéaire en fonction de la valeur de bêta. On ne remarque pas de plateau sur le graphe.

# 

# Map-Reduce

La technique Map-Reduce serait intéressante ici pour améliorer la vitesse d'exécution de l’algorithme.

À chaque itération du programme, l’algorithme effectue le produit d’une matrice par un vecteur. Considérant qu’il s’agit de grandes structures, nous pourrions répartir ce calcul en Map-Reduce sur différents CPU pour diminuer le temps de calcul.

Par exemple, on répartit un ensemble de lignes de la matrice sur chaque CPU (Map). En récupérant également le vecteur du produit, chaque CPU obtiendrait un vecteur résultat. L’ensemble des vecteurs résultats serait ensuite récupéré pour en faire la somme (Reduce). Toutefois, dans cet exemple-ci, cette répartition demanderait un nombre non négligeable de sommes flottantes, ce qui est peu intéressant en terme de précision.

On pourrait alors utiliser cette technique différemment comme en se concentrant sur une répartition en colonnes. En récupérant cette fois un sous-ensemble du vecteur du produit, chaque CPU calculerait alors un sous-ensemble du vecteur résultat final.

Cette technique est hautement recommandée ici compte tenu de la taille importante de la matrice envisagée.

# Conclusion

Notre implémentation de Pagerank n’est pas parfaite. Une construction de la matrice de transition en C sans passer par R aurait permis de gagner en temps d’exécution. Nous aurions aussi pu stocker nos données différemment, notamment la matrice de transition qui est une matrice creuse. Une structure de liste chaînée ne contenant que les valeurs non-nulles est une première idée envisageable.

Ce TP nous aura permis de réaliser les effets de β sur les résultats obtenus, de voir l’évolution du nombre d’itérations en fonction de la précision souhaitée ou de β et de visualiser les effets de la personnalisation de noeuds.