SAE S2.02 Exploration algorithmique d'un problème

PageRank & applications

Dans cette SAÉ nous étudierons (une version simplifiée de) l'algorithme d'ordonnancement *PageRank* qui est utilisé pour la recherche d'information sur le Web, et ses applications à différents problèmes et situations.

Consignes

Pour la partie implémentation, vous utiliserez `Python` ou `Sagemath` avec les bibliothèques nécessaires. Pour le rapport, afin de présenter du texte, du code ainsi que le rendu du code on utilisera `jupyter-notebook`.

Le dossier rendant compte du projet est à rendre sur la page Ametice de la SAÉ au format HTML et au format ipynb. La date limite de rendu est le **9 juin, 12h00**. Un seul étudiant par équipe déposera le dossier. Outre le code et l'analyse des résultats, le dossier présentera de façon exhaustive la participation de chaque étudiant au projet (qui a fait quoi) ainsi qu'un pourcentage d'implication dans le projet (qui a fait combien). Chaque membre du groupe devra maîtriser l'intégralité de ce qui est rendu.

L'oral aura lieu le 11 juin et sera une séance de questions sur le rapport d'une durée de 15 minutes par groupe.

Le dossier devra contenir les réponses aux 6 parties décrites ci-dessous. Il est bien évidemment attendu que vous compreniez le fonctionnement de l'algorithme `PageRank` mais dans le document à rendre il n'est pas nécessaire de re-détailler le fonctionnement de cet algorithme présenté dans la section ci-dessous.

Bien évidemment la totalité du rendu aura été écrit par l'un des membres du groupes. Le recours à toute aide extérieure est interdit.

Principe de PageRank

La recherche d'informations pertinentes sur le Web est un des problèmes les plus cruciaux pour l'utilisation de ce dernier. Il y a des enjeux économiques colossaux. Le leader actuel de ce marché, Google, utilise pour déterminer la pertinence des références fournies, un certain nombre d'algorithmes dont certains sont des secrets industriels jalousement gardés, mais d'autres sont publics.
On va s'intéresser à (une partie de) l'algorithme `PageRank`. L'idée de base utilisée par les moteurs de recherche pour classer les pages du web par ordre de pertinence décroissante consiste à considérer que plus une page est la cible de liens venant d'autres pages, c'est-à-dire plus il y a de pages qui pointent vers elle, plus elle a de chances d'être fiable et intéressante, et réciproquement. Il s'agit donc de quantifier cette idée, c'est-à-dire d'attribuer un rang numérique ou score de pertinence à chaque page.

On numérote les pages du web de 1 à N (N est très grand, de l'ordre de 10^{12}) et on note $r_i > 0$ le score de la page i. **C'est ce score que l'on cherche à calculer**.
On considère le Web comme un graphe orienté (vu en détails dans le TP de R2.07 dédié à la SAE) dont les sommets sont les pages numérotées de 1 à N. Il y a une arête de i vers j s'il y a sur la page i un lien hypertexte vers la page j (on dit que i pointe vers j). On définit la matrice $C = \left(c_{i,j}\right)_{1 \leq i,j \leq N}$ de taille $N \times N$ par

$$c_{i,j} = \begin{cases} 1 & \text{si } j \text{ pointe vers } i \text{ et } i \neq j, \\ 0 & \text{sinon.} \end{cases}$$

C'est donc la matrice transposée de la matrice d'adjacence du graphe du web.
L'algorithme `PageRank` part du principe que toutes les pages n'ont pas le même poids :

- plus la page i a un score élevé plus les pages j vers lesquelles la page i pointe auront un score important ;
- plus il y a de liens différents sur la page j et moins on attribue d'importance à chaque lien.

Ainsi, en notant N_j le nombre de liens présents sur la page j, on aboutit au système suivant

$$r_i = \sum_{j=1}^{N} \frac{c_{i,j}}{N_j} \times r_j, \qquad \text{pour tout } i \in \{1, \ldots, N\}.$$

On pourra remarquer que $N_j = \sum_{k=1}^{N} c_{k,j}$.
Finalement en adoptant les notations matricielles suivantes

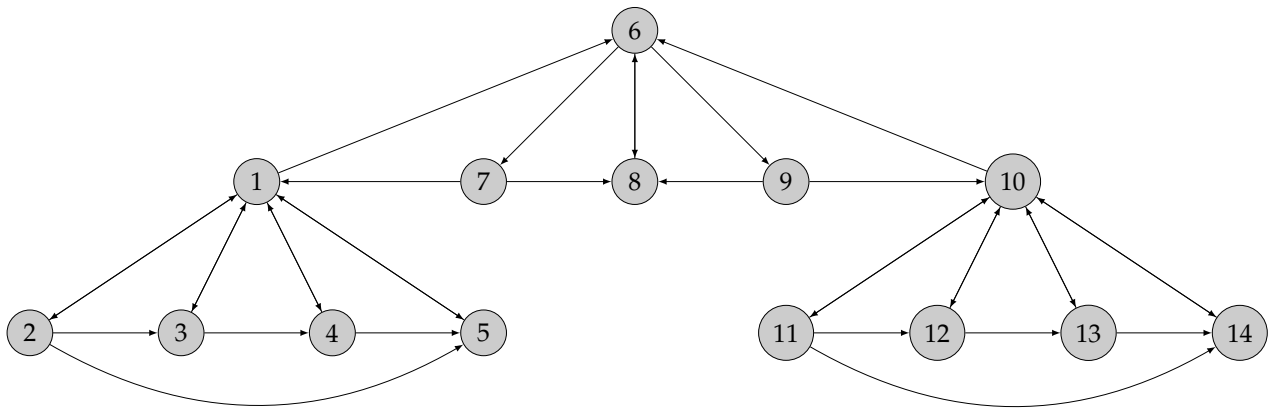
$$r = \begin{pmatrix} r_1 \\ \vdots \\ r_N \end{pmatrix} \in \mathbb{R}^N \qquad \text{et} \qquad Q = \left(q_{i,j}\right)_{1 \leq i,j \leq N} \quad \text{avec} \quad q_{i,j} = \begin{cases} \dfrac{c_{i,j}}{N_j} & \text{si } N_j \neq 0, \\ 0 & \text{sinon ,} \end{cases}$$

on aboutit à l'équation suivante : le vecteur r des scores est solution de $r = Qr$.

L'algorithme `PageRank` peut être utilisé pour des graphes qui ne sont pas associés au graphe du Web, dans ce cas ses résultats nécessitent une interprétation précise.

Partie 1 : PageRank - version itérative, premier exemple

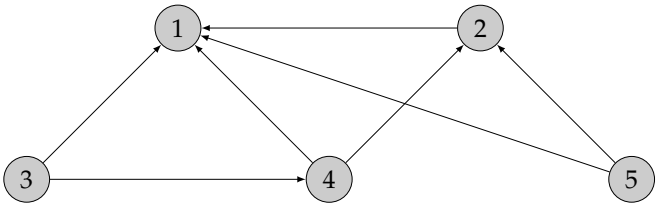
On considère le graphe du web simplifié suivant (avec $N = 14$ pages).



1. Justifier pourquoi l'algorithme de la puissance itérée (vu en détails dans le TD de R2.09 dédié à la SAE) permet de calculer le score de chacune des pages.
2. Implémenter cet algorithme pour calculer le score de chacune des pages du graphe précédent. On vérifiera (numériquement) que le vecteur de score obtenu est bien approximativement solution de $r = Qr$.
3. Analyser la pertinence du résultat obtenu.

Partie 2 : PageRank - version itérative, deuxième exemple

1. Appliquer l'algorithme de la Partie 1 au graphe suivant et commenter le résultat obtenu



Pour éviter cela, on appliquera désormais l'algorithme de la puissance itérée à la matrice de transition $P = \left(p_{i,j}\right)_{1 \leq i,j \leq N}$ définie par

$$p_{i,j} = \begin{cases} \alpha q_{i,j} + \dfrac{1-\alpha}{N}, & \text{si } N_j \neq 0, \\ \dfrac{1}{N} & \text{sinon,} \end{cases}$$

où $\alpha \in [0,1]$ est un paramètre à choisir appelé "facteur d'amortissement".

2. En utilisant cette matrice de transition (avec $\alpha = 0,85$), calculer les scores de chacune des pages du graphe précédent. Commenter. On vérifiera (numériquement) que le vecteur de score obtenu est bien approximativement solution de $r = Pr$.

Dans la suite c'est cet algorithme que l'on désignera par `PageRank` version itérative. Il est appelé itératif car il calcule successivement des approximations du vecteur de score r cherché.

Partie 3 : PageRank - version itérative, analyse

Pour le moment on pose $\alpha = 0,85$ et on considère le graphe de la Partie 1.

1. Analyser l'influence du critère d'arrêt dans l'algorithme de la puissance itérée.
2. Ajouter quelques hubs (pages qui ont beaucoup de liens sortant) et autorités (pages qui ont beaucoup de liens entrant). Commenter l'impact sur les scores.
3. Essayez d'accroître le score de certaines pages. Expliquez votre méthode et validez-la expérimentalement.
4. Faites varier le facteur d'amortissement α pour analyser son influence. On rappelle que $\alpha \in [0,1]$.

Partie 4 : PageRank - version itérative, analyse

Dans cette partie, appliquez l'algorithme de `PageRank` version itérative aux différentes matrices ci-dessous. On portera une attention particulière à l'analyse des résultats ainsi qu'à leur interprétation.
Les différentes matrices mentionnées ci-dessous sont détaillées sur la page Ametice

1. En utilisant le logiciel d'exploration de site web présent sur la page Ametice (et vu en TP de R2.07), construire trois matrices de votre choix et appliquez l'algorithme de `PageRank` à ces matrices. L'ordre de ces matrices sera compris entre 10 et 30. Ces matrices, le résultat du logiciel d'exploration ainsi que le site web choisi apparaîtront clairement dans le dossier.
2. Sur la page Ametice vous trouverez les matrices du réseau routier (issues de OpenStreet Map) de différentes villes. Chaque groupe étudiera la matrice "413 Avenue Gaston Berger" ainsi que la ville qui lui a été affectée (voir sur la page Ametice).

Partie 5 : PageRank - calcul direct des scores et comparaisons d'algorithmes

1. On rappelle que le vecteur de score est solution du système $r = Pr$. En déduire un algorithme de calcul direct (c'est-à-dire de calcul exact et sans approximations successives) du score r. Ecrire le pseudo-code correspondant à cet algorithme.
2. Implémenter cet algorithme.
3. Comparer les résultats obtenus par les deux algorithmes.
4. Comparer les performances des deux algorithmes. Si le temps d'exécution peut être un indicateur pertinent, ce n'est pas le seul.

Partie 6 : PageRank - matrice du langage

Cette partie bonus ne sera traitée que lorsque tout le reste aura été fait.
Reprendre l'analyse de la partie 4 avec la matrice du langage donnée sur la page Ametice. La difficulté est de gérer la taille des données puisque cette matrice est de taille 180062×180062.