UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

# Compte rendu du projet - Méthodes de ranking


Réalisé Par

BARBIER Yann
ESTEVAN Benjamin
GOUY Emeric
HELAIN Chloé
SOURSOU Adrien
ZENS Coralie


2022

# Table des matières

# Chapitre 1

# Présentation Du Sujet

## 1.1 Rappel de l'énoncé

Comme PageRank est exécuté chaque mois et que le graphe du Web évolue lentement, il est possible d'initialiser le vecteur x à la valeur obtenue lors du mois précédent en faisant attention aux sommets disparus.

L'initialisation est la suivante : soient $R$ les sommets du graphe $H^t$ qui ne sont pas dans le graphe $H^{t+1}$. $R$ contient les sommets disparus.

— Si $i$ est dans le graphe $H^{t+1}$, alors $x_{t+1}[i] = x_t[i]$

— Ensuite on normalise le vecteur car la somme des $x_{t+1}[i]$ est plus petite que 1.0

Notre objectif est d'étudier l'accélération de la convergence de l'algorithme des puissances lorsqu'il est initialisé comme proposé (par rapport à la version du cours où il est initialisé avec $e/n$). Pour cela, nous proposerons des algorithmes de modification des graphes du Web (retrait de sommets, gestion des arcs arrivant sur une page vide) dont nous disposons.

## 1.2 Travail à effectuer

Le travail à effectuer est l'étude de la performance de l'algorithme PageRank comparée à un algorithme PageRank modifié. Pour cela, nous coderons l'algorithme PageRank, nous implémenterons un algorithme dit "custom" et nous comparerons leurs performances à travers le nombre d'itération avant convergence, et selon les paramètres donnés suivants :

— Alpha

— Ratio de sommets détruits

— Ratio d'arcs détruits

Nous étudierons plus particulièrement les performances grâce à un calcul de l'accélération des exécutions par notre modification. Le calcul de l'accélération est décrit dans ce rapport en partie 2.2.2.

## 1.3 Plan du rapport

Afin d'organiser ce compte-rendu, nous avons choisi de le répartir en cinq points principaux sans compter la bibliographie ni les annexes :

### Présentation du sujet

Nous avons pu introduire le sujet qui nous a été proposé dans la partie ci-dessus, ainsi que cette présentation du plan de ce rapport.

### Rappels théoriques

Dans cette partie, nous rappellerons le fonctionnement de PageRank ainsi que les modifications du nouvel algorithme que nous utiliserons. Nous avancerons le point de vue théorique des modifications apportées.

### Travail effectué

Ici, nous expliquerons notre implémentation des algorithmes et notre manière de procéder pour faire ressortir les résultats produits par ces simulations et les différents axes étudiés.

### Analyse des résultats

Dans cette section, nous exploiterons les résultats générés par notre programme principal sur les différents graphes à notre disposition. Nous chercherons à voir si nos modifications auront eu un impact positif sur les performances de PageRank et sinon les origines de ceci.

### Conclusion

Nous effectuerons un bilan du projet dans cette partie tout en donnant le mot de fin de notre étude.

# Chapitre 2

# Rappels Théoriques

## 2.1 Fonctionnement de PageRank

L'algorithme de PageRank est utilisé pour obtenir une approximation du vecteur $\pi$ de pertinence des pages du Web de façon efficace et performante.

### 2.1.1 Présentation de PageRank

D'une part cet algorithme va ajouter une *marche aléatoire* sur le graphe pour pouvoir appliquer une méthode des puissances sur des graphes du Web possédant des pages dites vides qui ne possèdent pas de liens vers d'autres pages.

D'autre part, PageRank va modifier la formule de la matrice qu'il analyse pour assurer que les *conditions de convergence* de l'algorithme soit respectée grâce au paramètre $\alpha$ qui assure l'irréductibilité et l'apériodicité de la matrice sur laquelle s'applique l'algorithme.

### 2.1.2 La méthode des Puissances en général

La pertinence des pages Web se définit de façon circulaire telle qu'une page est pertinente si elle pointe et est pointée par des pages pertinentes.
Chaque page distribue de la pertinence et en reçoit selon un équilibre. La pertinence d'une page est divisée équitablement sur les pages vers lesquelles elle pointe. La pertinence d'une page est la somme des parts de pertinence qui pointe vers elle.

Pour déterminer le vecteur des pertinences $\pi$ sur la matrice $G$, on doit résoudre le système suivant :

$$\pi = \pi G \ et \ \pi e^t = 1 \text{ avec } e^t \text{ le vecteur colonne de 1.}$$

La distribution équitable des pertinences au sommet que l'on pointe se traduit par une matrice du graphe $P$ différente de la matrice d'adjacence $A$ tel que :

$$\text{Si } d^+(i) > 0,\ P[i;j] = \frac{A[i;j]}{d^+(i)}$$
$$\text{Sinon } P[i;j] = 0$$

L'additivité des pertinences qui pointent vers un sommet est traduite par la formule :

$$\pi(j) = \Sigma_i \pi(i) P[i;j]$$

Ces deux principes définissent la méthode des puissances qui est initialisée avec le vecteur $x = \frac{1}{N}$ et se réitère jusqu'à équilibre du vecteur $x$ tel que $x^{(t+1)} = x^{(t)}P$.

Cette initialisation donne initialement la même note à tous les sommets pour partir avec un stade neutre qui ne donne aucune préférence.

### 2.1.3 Marche aléatoire sur le graphe

Lors d'une exécution de la méthode des puissance, la pertinence des lignes vides ne peut être estimée à cause des multiplications par 0. D'un point de vue du parcours du graphe par des robots, ces pages posent un problème car elles n'ont aucune page à qui transmettre leur pertinence.

Avec une marche aléatoire, les robots iront depuis ces pages vides sur une page aléatoire avec une probabilité de $1/N$ pour des graphes de $N$ sommets.

Cela revient à ajouter des lignes de $1/N$ à la place des 0 dans la matrice $P$. On définit donc la matrice $M$ tel que : si $d^+(i) > 0$, $M[i;j] = P[i;j]$ sinon $M[i;j] = 1/N$.

On peut écrire $M$ également avec le vecteur $f$ tel que $\forall i, d^+(i) = 0$ alors $f(i) = 1$ sinon $f(i) = 0$. Alors on a $M = P + 1/N\ f^t\ e$.

Avec cette matrice, la méthode des puissances peut-être appliquée à un graphe contenant des pages vides.

### 2.1.4 La méthodes des puissances pour PageRank

En plus d'établir son exécution sur une matrice sans ligne vide, PageRank va s'assurer que la matrice sur laquelle elle travaille respecte les conditions de convergence de la méthode des puissances qui sont celles d'initialisation de l'algorithme et celles sur $M$ d'existence de $\pi$.

Comme $M$ est une matrice stochastique, pour résoudre $\pi = \pi M\ et\ \pi e^t = 1$, il faut respecter les conditions à l'existence d'une solution stationnaire pour une chaîne de Markov en temps discret.

Si $M$ est finie, irréductible et primitive alors il existe une solution de $\pi M = \pi$ et $\pi e^t = 1$ et cette solution ne dépend pas de l'état initial de la marche et $\lambda_2 < 1$ (la deuxième valeur propre). Cela signifie que la méthode des puissances converge vers cette unique solution quelque soit l'initialisation.

On peut déjà définir que $M$ est finie. Cependant le graphe n'est pas fortement connexe et donc $M$ n'est pas irréductible. Enfin une matrice est primitive si elle est finie, irréductible et apériodique.

PageRank va donc ajouter un paramètre $\alpha$ tel que depuis tout sommet le robot navigue aléatoirement sur le graphe avec une probabilité de $1-\alpha$ ou sinon il navigue normalement sur le graphe du Web.

On définit alors la matrice du graphe du Web $G$ pour PageRank tel que :

$$G \;=\; \alpha M + (1-\alpha)\frac{1}{N}\, e^t\, e \;=\; \alpha P + \alpha\, \frac{1}{N}\, f^t\, e + (1-\alpha)\frac{1}{N}\, e^t\, e$$

La matrice $G$ est définit tel que la probabilité de passer entre deux sommets du graphes est au moins de $\frac{1-\alpha}{N}$. Ainsi $G$ est une matrice irréductible.

De plus, la probabilité d'aller d'un sommet sur lui-même n'est pas nulle, $G$ est également apériodique et ainsi primitive.

Par conséquent, cette matrice respecte les conditions de convergence de l'algorithme des puissances. On a donc pour l'algorithme des puissances sur $G$ l'équation suivante :

$$xG = \alpha xP + \left(\frac{1-\alpha}{N} + \frac{\alpha}{N}(x\ f^t)\right)e$$

### 2.1.5 Condition d'arrêt

On définit la condition d'arrêt de notre algorithme quand il y a convergence des vecteurs d'une itération avec la précédente. On estime ce test de convergence vérifié quand :

$$\left|\left|x^{(t+1)} - x^{(t)}\right|\right|_1 < \epsilon$$

Ce qui revient sur les deux derniers vecteurs $x^{(t)}$ et $x^{(t+1)}$ à tester si : $\Sigma_i \left|x(i)^{(t)} - x(i)^{(t+1)}\right| < \epsilon$.

### 2.1.6 Résultats

En sortie de l'exécution de PageRank, on obtient un vecteur $\pi$ qui est une approximation des pertinences des pages du graphes du Web.

Le résultat qui nous intéresse est la performance de l'exécution de l'algorithme qui se traduit par le nombre d'itération nécessaire avant convergence des vecteurs $x^{(t)}$ et $x^{(t+1)}$.

## 2.2 Modifications de l'algorithme

### 2.2.1 Changement sur l'initialisation

La version modifiée de l'algorithme de PageRank que l'on étudie change l'initialisation en s'appuyant sur le précédent résultat de l'exécution de PageRank. On utilise le fait que l'on connaît déjà le vecteur $\pi$ de la précédente mise à jour et que celui-ci serait proche du prochain vecteur $\pi$.

Cependant on ignore l'importance qu'auront les modifications du graphe du Web depuis la dernière mise à jour sur le vecteur de pertinence. Dans cette analyse, nous nous concentrons uniquement sur le scénario où des pages ont été supprimées du graphe depuis la dernière mise à jour et il nous faut donc normaliser le vecteur de probabilité qui sera l'initialisation de l'algorithme PageRank.

Cette modification devrait être une amélioration de l'algorithme car il donne dès le début une valeur relative de la pertinence précédemment acquise par la page.
Cela peut ne pas être une amélioration si les sommets détruits ont de l'importance dans les pertinences de nombreux graphes. Moins il y a de différence entre les deux versions du graphe du Web, plus on suppose cette version de PageRank performante.

La suite de ce rapport vise à en juger concrètement à travers des simulations de mise à jour des graphes du Web par des destructions de sommets et un comparatif des exécutions des deux versions de PageRank : l'originale, étudiée en cours et initialisant avec $e/N$, et notre version initialisant avec le résultat de l'exécution de PageRank sur le même graphe avec plus de sommet.

### 2.2.2 Performance et Accélération

Le nombre d'itération obtenue avec l'algorithme customisé de PageRank peut être directement comparé à l'exécution de PageRank sur ce même graphe avec une initialisation avec $e/N$.

Cependant cette comparaison dépend directement du graphe et de sa structure.

Pour analyser la différence de performance des algorithmes entre des graphes très différents, nous avons recherché une valeur plus neutre et donnant la différence relative entre les deux exécutions qui ne dépendent pas du nombre réel d'itération.

Nous avons donc choisi de calculer l'accélération entre les deux exécutions, c'est à dire la différence du nombre d'itération normalisée par rapport à l'exécution du PageRank "classique".

#### Formule de l'accélération

Acc est l'accélération (sans unité) de l'exécution de PageRank par la customisation.
avec NbItPR est le nombre d'itérations avant convergence de PageRank "classique".
et NbItPRc est le nombre d'itérations avant convergence de PageRank customisé.

$$Acc = \frac{NbItPR - NbIterPRc}{NbItPR} = 1 - \frac{NbIterPRc}{NbItPR}$$

#### Lecture des valeur d'accélération

Ainsi une valeur d'accélération de 0.1 signifie que l'algorithme customisé réalise 10% d'itération en moins pour vérifier la condition d'arrêt de l'algorithme.

Au contraire une valeur d'accélération de -0.1 signifie que l'algorithme customisé réalise 10% d'itération en plus pour vérifier la condition d'arrêt de l'algorithme.

Donc avec la version modifiée, l'exécution de PageRank va accélérer de la valeur de l'accélération par rapport à l'exécution "classique".

Ce calcul donne une importance plus forte à retirer une itération dans le cas où il y a déjà peu d'itérations dans l'exécution classique qu'à retirer une itération dans le cas où il y a beaucoup d'itérations dans l'exécution classique.

# Chapitre 3

# Travail Effectué

## 3.1 Programme principal

La marche à suivre pour exécuter le programme et exploiter les résultats est indiquée dans le fichier README.md du projet. Le programme nécessite une liste d'arguments donnée ci-dessous :

— Le fichier contenant le graphe du Web d'origine

— Un fichier de sortie des résultats

— Un ensemble de valeurs à utiliser pour $\alpha$ spécifié de la manière suivante : $\alpha_{min}$, $\alpha_{max}$, $\alpha_{pas}$. Ainsi, on sélectionne toutes les valeurs entre $\alpha_{min}$ et $\alpha_{max}$ par pas de $\alpha_{pas}$,

— Un ensemble de valeurs à utiliser pour le ratio de sommets $r$ à enlever dans le graphe, spécifié comme pour $\alpha$,

— Un nombre de sous-graphes $n$ à générer pour chaque couple de paramètres $(r, \alpha)$.

Cette séparation entre $\alpha$ et $r$ nous permet de faire des études différenciées, se focalisant plus sur un des paramètres que sur l'autre. Par exemple, avec un pas pour $r$ plus petit que celui pour $\alpha$, l'accent sera mis sur la variation de l'accélération par rapport à $r$ avec un aperçu sommaire de l'influence de $\alpha$ et vice-versa. Cela nous permet de parcourir l'espace des paramètres $(r, \alpha)$ de manière plus intelligente que si l'on en faisait un simple quadrillage à pas égal.

## 3.2 Structures de données utilisées

Afin de stocker les informations des graphes, nous avons crée notre propre structure de matrice creuse, "s_matrix". Celle-ci contient :

— **vertices_count** : Le nombre de sommets du graphe

— **edges_count** : Le nombre d'arcs du graphe

— **edges** : Un tableau stockant l'ensemble des arcs du graphe, de manière linéaire (les arcs partant du premier sommet sont placées en premier, etc.)

— **row_start** : Un tableau qui indexe **edges** sur le numéro de sommet de départ (**row_start[i]** correspond à l'index du premier arc du (i+1)-ème sommet dans **edges**)

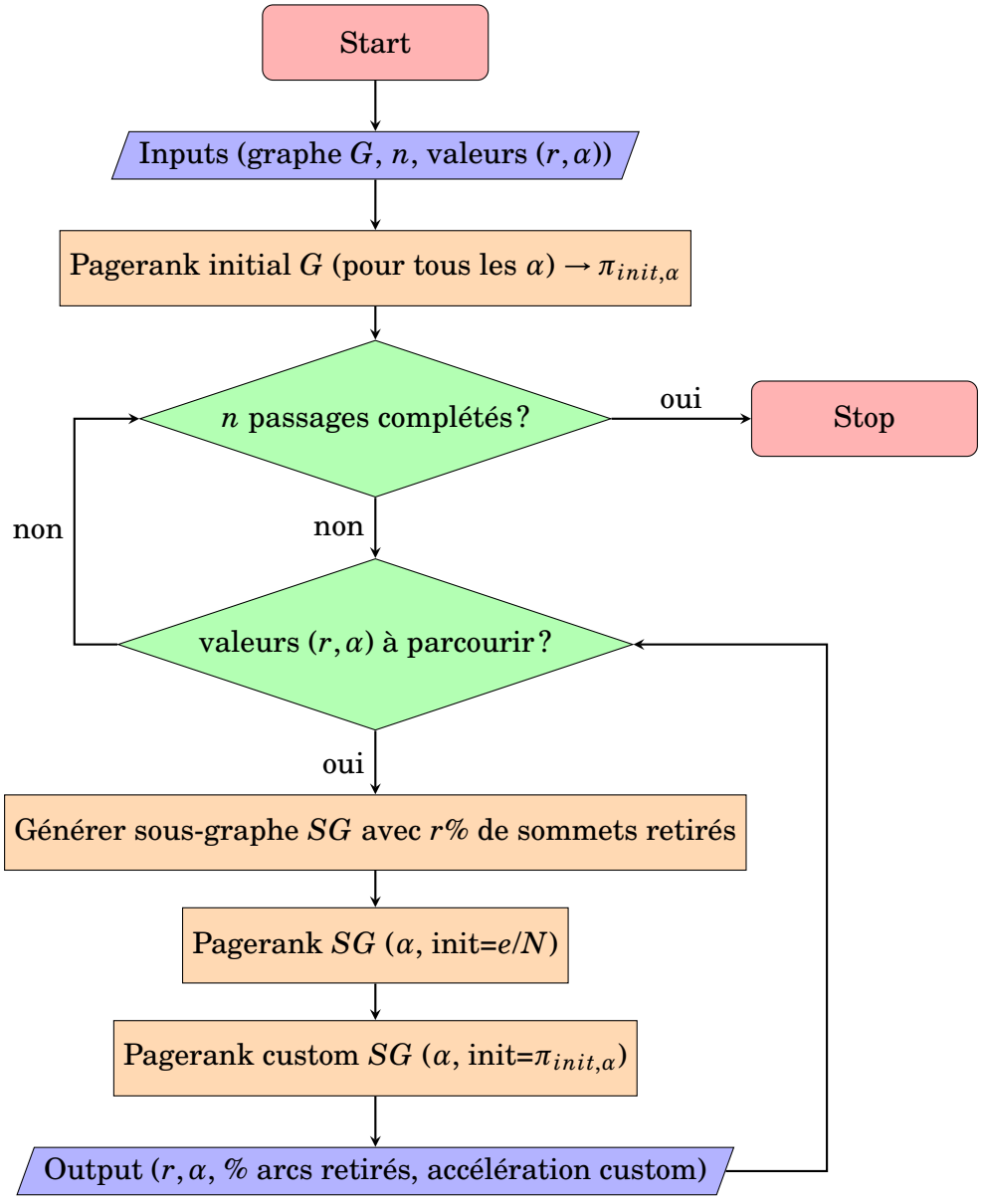


FIGURE 3.1 – *Flowchart* de notre programme principal

Aussi, nous avons créé une structure permettant de stocker un arc de cette même matrice avec comme informations :

— **y** : Le sommet d'arrivée de l'arc (le sommet de départ est déjà connu via le placement de l'arc dans le tableau **edges**)

— **w** : Le poids de l'arc

## 3.3 Génération des sous-graphes du web

Le premier objectif a été de générer de nouveaux graphes à partir des graphes originaux en supprimant un pourcentage de leurs sommets aléatoirement pour simuler l'évolution du graphe du Web. Nous proposons dans cette section de détailler notre algorithme de génération de sous-graphes qui est un élément primordial pour la réalisation de notre étude. Le paramètre qui a été pris en compte en priorité concerne la taille importante des graphes à traiter. En effet, le graphe le plus lourd (wb–edu) contient environ 10M sommets et 60M arcs. Il n'était donc pas envisageable d'implémenter une solution avec une complexité polynomiale. Nous introduisons donc une procédure de génération de sous-graphes du Web avec l'Algorithme 1 disponible en annexe.

Cet algorithme génère des sous-graphes en $O(S+A)$ avec $S$ le nombre de sommets du graphe d'origine et $A$ le nombre d'arcs. Nous avons implémenté un système de cache pour éviter de réallouer la mémoire des tableaux temporaires et du sous-graphe à chaque appel.

## 3.4 Génération des données

Avec le programme principal, nous générons, pour chaque graphe du web, deux fichiers `graphe_ratio.data` et `graphe_alpha.data` contenant des lignes avec les résultats suivants (comme indiqué en Figure 3.1) :

$$\alpha \text{ / } r \text{ / \% Arcs Détruits / Accélération custom}$$

Les paramètres utilisés pour la génération des fichiers sont les suivants :

— Le nombre de répétitions $n$ est fixé à 10.

— `graphe_ratio.data` : focalisation sur l'étude du ratio de sommets détruits, avec les paramètres :

  — $r \in \{0, 0.004, 0.008\}$ (très petites valeurs de ratio)

  — $r \in \{0.01, 0.04, 0.07, \cdots, 0.31\}$ (autres valeurs de ratio)

  — Dans ces deux scénarios, on utilise $\alpha \in \{0.80, 0.85, 0.90\}$

— `graphe_alpha.data` : focalisation sur l'étude de $\alpha$, généré avec les paramètres suivants : $\alpha \in \{0.70, 0.74, 0.78, \cdots, 0.94\}$, $r \in \{0.05, 0.10, 0.15\}$

Le choix de ces paramètres est motivé par 2 raisons principales :

— Représenter des situations réelles, avec des valeurs de $\alpha$ assez élevées ($\geq 0.70$) pour avoir un graphe modifié par Pagerank encore assez représentatif du graphe original et des valeurs de $r$ faibles ($\leq 0.05$), puisqu'un graphe de type web évolue en général peu.

— Tester les limites de l'algorithme custom, avec les valeurs de $r$ élevées, afin de voir jusqu'à quel point l'algorithme custom est plus rapide que la version normale.

## 3.5 Exploitation des résultats et génération des graphiques

Le script en bash lançant l'étude des 6 graphes génère deux fichiers data par graphe. L'un servant à l'étude de l'accélération en fonction du ratio de sommets et d'arcs détruits, et l'autre à l'étude de l'accélération en fonction de alpha. Ce script bash appelle lui même un script en R une fois les fichiers data générés, lançant l'étude complète des 6 graphes donnés, à savoir :

— wb–cs–stanford

— Stanford

— Stanford_BerkeleyV2

— in–2004v2

— wikipedia–20051105V2

— wb–edu

Le script R procède tout d'abord en chargeant le fichier dans lequel nous faisons varier le ratio de sommets détruits puis génère trois courbes, chacune étant le fruit de l'évolution de l'accélération en fonction du ratio de sommet (ou d'arcs) détruits avec une valeur de alpha fixe différente. Ces valeurs sont 0.80, 0.85 et 0.90.

De même, le graphique de l'étude de l'accélération en fonction de alpha est lancé avec le ratio de sommets détruits fixé à 0.05, 0.10 et 0.15. Notons que dans tous les graphes, un axe y = 0 est placé pour situer la partie du graphe où l'accélération peut éventuellement devenir nulle voir négative.

# Chapitre 4

# Analyse Des Résultats

## 4.1 Tendances globales sur l'ensemble des graphes

### 4.1.1 Influence du ratio de sommets détruits sur l'accélération

Le graphique figure 6.1 présente l'évolution de l'accélération moyenne par l'algorithme modifié par rapport à PageRank (en ordonnée) selon le ratio de sommets supprimés sur le graphe par rapport à celui du précédent état (en abscisse). À chaque courbe est assignée une valeur de alpha fixe.

Nous pouvons immédiatement constater en analysant les courbes de ce graphique qu'elles sont confondues. Cela indique que pour une même valeur du ratio de sommets détruits, les changements sur alpha vont avoir une influence négligeable sur la donnée analysée : l'accélération moyenne produite par l'algorithme modifié en fonction du ratio de sommets détruits.

La courbe que nous avons sous les yeux commence par une chute brutale. En effet, quand la proportion de sommets retirés est extrêmement faible, le vecteur de pertinence solution de PageRank dérivé du graphe original est déjà très proche de la solution à rechercher dans le sous-graphe. Dans ce contexte, les convergences ont lieu en moins de 10 itérations avec l'algorithme modifié.

À partir de 1% de sommets détruits, nous avons une réduction plus progressive de l'accélération moyenne. La quantité de sommets détruits augmentant, le vecteur de pertinence dérivé est de moins en moins adapté au sous-graphe considéré : l'information acquise par l'algorithme modifié le fait ralentir. Dans la zone des 20-25%, une grande partie des exécutions modifiées sont au final significativement plus lentes que les exécutions "classiques" associées. En effet, plus le ratio de sommets à traiter est important, plus la vitesse de convergence de l'algorithme modifié risque d'être élevée comparé au PageRank original. La courbe tend ensuite vers 0 lorsque le ratio de sommets à traiter diminue à chaque itération, et ce pour des valeurs de ratios de sommets à traiter dans l'intervalle que nous avons jugé pertinent. La courbe peut en effet se prolonger dans des valeurs d'accélération en moyenne négatives.

### 4.1.2 Influence du ratio d'arcs détruits sur l'accélération

Le graphique figure 6.2 présente l'évolution de l'accélération moyenne permise par l'algorithme modifié par rapport à PageRank (en ordonnée) selon le ratio d'arcs supprimés à chaque itération (en abscisse). À chaque courbe est assignée une valeur de alpha fixe.

Cette courbe est relativement similaire à celle traitée précédemment. En effet, elle arbore le même profil. Nous pouvons constater que la valeur alpha a toujours un effet négligeable mis en évidence par le fait que les courbes sont confondues.

Le ratio d'arcs détruits est directement lié au ratio de sommets détruits. Il est donc logique que de la même manière que pour la courbe précédente, l'accélération moyenne soit très forte pour des valeurs de ratio d'arcs détruits proches de 0, puis qu'elle s'amortisse avant de passer en dessous de 0 au-delà de 30% détruits.

### 4.1.3 Influence d'alpha sur l'accélération

Le graphique figure 6.3 présente l'évolution de l'accélération moyenne de l'exécution de l'algorithme PageRank modifié par rapport à l'original (en ordonnée) selon alpha (en abscisse). À chaque courbe est assignée une valeur du ratio de sommets supprimés fixe.

Le graphique ci-dessus montre trois courbes clairement distinctes, plates et qui ne se croisent jamais. Il est donc raisonnable de penser que si le ratio de sommets supprimés a une incidence directe sur l'accélération permise par l'algorithme modifié par rapport à PageRank, $\alpha$ a globalement très peu d'effet sur cette accélération moyenne. Les trois courbes semblent dotées d'une pente légèrement négative, laissant croire que plus $\alpha$ est élevé et moins l'accélération moyenne est prononcée.

Cependant la dispersion des points indique que l'accélération va beaucoup varié pour une même valeur d'alpha tombant dans certains cas dans des accélération négative et donc une exécution plus lente avec notre version de PageRank.

Nous pouvons constater à nouveau que le ratio de sommets supprimés a une influence importante sur l'accélération moyenne de l'exécution. Plus le ratio de sommets détruits est faible, plus l'accélération est élevée en moyenne : ce qui corrobore les observations faites pour les précédents graphiques.

## 4.2 Analyse comparative entre les graphes du Web étudié

Pour approfondir notre analyse, nous avons fait ressortir les résultats graphes par graphes et nous observons que certains vont avoir des tendances différentes de la tendance globale.

Les figures sur les ratio de sommet sont en celles 6.4 à 6.6. Les figures sur les ratio d'arcs sont en celles 6.7 à 6.9. Les figures observant l'influence d'alpha sont celles en 6.10 à 6.12.

En analysant les trois figures séparées sur les différents graphes du Web, nous pouvons remarquer que les graphes Stanford, Stanford_BerkeleyV2, in–2004V2 et wb–edu suivent la même allure qui est celle qui ressort sur l'étude globale.
Ces 4 graphes vont avoir une accélération en moyenne plus forte de l'exécution sur de très faibles ratios de sommets détruits et une pente qui ralentit très vite pour à peine atteindre 0 avec 30% de destruction. Et ils vont avoir une influence d'alpha négligeable sur l'accélération moyenne de l'algorithme PageRank par notre modification.

La deuxième chose que nous pouvons remarquer est que wb–cs–stanford a un comportement qui diffère des observations faite sur l'étude globale. L'accélération devient négative en moyenne passé un ratio de 7% de sommets ou 13% d'arcs détruits pour $\alpha = 0.85$, alors qu'elle semble tendre vers 0 en moyenne ou passer très légèrement en-dessous de 0 pour tous les autres graphes.
L'allure des courbes en fonction d'alpha aussi montre une allure remarquable. En effet, wb–cs–stanford a une accélération moyenne qui diminue lorsque la valeur de alpha augmente. On remarque que la pente de la courbe est plus prononcée que pour tous les autres graphes.

Nous pourrions tenter d'expliquer la plus forte influence du ratio de sommets ou arcs détruits et l'influence d'alpha sur celui-ci par sa plus petite taille que les autres graphes que nous étudions. Or comme nous étudions des ratio de sommet détruit et non un nombre numérique, la partie est proportionnelle à la taille du graphe et donc sa taille ne devrait plus rentrer en compte. Cela n'est pas la bonne explication et il y a autre chose qui rentre en compte.

Il est préférable de supposer que ce graphe a une particularité dans sa structure et l'organisation des arcs qu'il possède. Cette particularité fait que la destruction d'une portion des sommets va fortement influencer la pertinence des sommets restants. En conséquence de quoi initialiser avec les pertinences relatives du graphe originel éloignera plus le vecteur d'initialisation du vecteur de pertinence qu'une initialisation neutre et équitable avec $e/N$. De plus, la marche aléatoire sur le graphe avec une probabilité de $1 - \alpha$ va rendre la version de PageRank modifiée légèrement plus performante.

Enfin nous pouvons observer que le dernier graphe wikipedia–20051105V2 a de meilleures valeurs d'accélération moyenne que tous les autres graphes pour des conditions équivalentes. L'accélération reste toujours positive même avec 30% des sommets détruits ou 50% d'arcs détruits.

Ces meilleures performances dues à notre initialisation peuvent encore s'expliquer par la structure du graphe qui rend l'influence de la destruction de sommet sur la pertinence des sommets restant moins importante. Cela peut s'expliquer par la forte connexité des pages de wikipedia entre elles et par la limite à une seule citation de la même page par page.

Sur ce graphe, nous observons un comportement remarquable des courbes d'alpha.

Pour $\alpha = 0,9$, on observe qu'il offre la meilleur accélération moyenne à même ratio de sommet ou d'arc détruit jusqu'à 13% de sommet et 25% d'arcs après quoi cette valeur d'$\alpha$ devient moins performante que les autres comparées sur cette figure.
A contrario lorsque $\alpha = 0,8$, l'accélération est moins bonne en moyenne à moins de 13% de sommet et 25% d'arcs et devient au delà la meilleure d'$\alpha$. Cette observation n'a pas d'équivalent remarquable sur les autres graphes. On peut en déduire que la marche aléatoire n'améliore pas les performances de notre version de PageRank s'il y a peu de sommets détruits dans ce

graphe. Au contraire sur les autres graphes, plus la marche aléatoire est présente, plus notre algorithme est performant quelque soit le ratio de sommets ou d'arcs détruits.

L'analyse en fonction d'alpha pour wikipedia–20051105V2 présente également quelques excentricités. Pour de très faible ration de sommets détruits, la courbe monte légèrement lorsque alpha augmente. De plus, les valeurs d'accélérations générées par l'étude de ce graphe indiquent qu'elles sont majoritairement supérieures à la moyenne globale. Elles se situent entre 0.1 et 0.35 là où aucune des courbes des autres graphes ne dépasse une accélération de 0.2.

Les différents graphes du Web vont montrer des influences proches des portions du graphes détruites et d'alpha généralement identiques sur l'accélération moyenne de l'exécution de PageRank avec notre algorithme d'initialisation. Cependant des cas à part ressortent et sont le reflet de particularités structurelles de ces graphes.

# Conclusion

Les modifications que nous avons apportées à l'algorithme PageRank permettent le plus souvent de réduire le nombre d'itérations avant convergence.

Cette accélération se situe en général entre 0 et 20 %, mais elle peut vraisemblablement aller au-delà et réduire de 80% la durée de l'exécution lorsque peu de sommets sont détruits à chaque itération (avec un ratio détruit très proche de 0%). Sur des ratios plus importants de sommets détruits, la version modifiée de PageRank entraîne un plus grand nombre d'itérations pour atteindre le nouveau vecteur de pertinence car il y a trop de différences entre le graphe de référence et la nouvelle mise à jour. Ces cas n'étant pas observés dans le Web du fait de sa taille, nous pouvons conclure de ce point de vue que l'initialisation modifiée de PageRank est préférable à l'initialisation classique.

Alpha au contraire ne semble jouer qu'un rôle mineur. Évidemment, une plus grande valeur d'alpha augmente le nombre d'itérations avant convergence en réduisant l'importance de la marche aléatoire. Cependant en comparant les deux algorithmes, sa valeur n'a globalement pas d'influence sur l'accélération moyenne de PageRank par notre algorithme.

Il existe également des différences de comportement entre les graphes, bien qu'elles soient la plupart du temps négligeables ou minimes sur les intervalles de valeurs qui nous intéressent.

En pratique, un vecteur x tel qu'implémenté ici est bien utilisé par Google pour accélérer le calcul de sa matrice des pertinences. D'après l'université polytechnique de New York, PageRank est lancé environ une fois par mois en utilisant cette méthode [4].

# Chapitre 5

# Bibliographie


[1] Documentation du langage C :
https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

[2] Documentation du langage R :
https://www.r-project.org/other-docs.html

[3] Fonctionnement de PageRank :
https://www.webrankinfo.com/dossiers/pagerank/formule

[4] Détails supplémentaires sur le fonctionnement de PageRank de l'université polytechnique de New York (document de cours) : K.Ming Leung, 2004
https://cse.engineering.nyu.edu/ mleung/CS4744/f04/PageRank/PageRank.pdf

[5] PAGE, L. (1998). *Method for node ranking in a linked database* (U.S. Provisional Patent Application Ser. No. 09/004,827). U.S. Patent and Trademark Office. `https://patents.google.com/patent/US6285999B1/en`

# Chapitre 6

# Annexe

## Code

L'algorithme 1 décrit en pseudo-code la génération des sous-graphes dont un ratio de sommet est détruit.

## Graphiques

Les graphiques étudiant l'accélération de l'exécution de PageRank par notre modification globalement, tous graphes du Web confondus sur les figures 6.1 à 6.3.

Les graphiques étudiant l'accélération de l'exécution de PageRank par notre modification globalement, un graphe du Web à la fois sur les figures 6.4 à 6.12.

**Algorithme 1 :** GenerateSubgraph(G, n)

**Entrées :** - G : Matrice creuse de S sommets et A arcs : Le graphe à partir duquel on veut générer un sous-graphe.
- n : entier : Le nombre de sommets à supprimer avec $n \leq S$

**Résultat :** Matrice creuse de S - n sommets et A arcs maximum : Le sous-graphe généré.

```
1  removedSet ← {faux}
2  verticesSet ← {1..S}
3  edgesCountSet ← {0}
4  Mélange(verticesSet)
5  pour i ← 1 à n faire
6      removedSet_{verticesSet_i} ← {true}
7  e ← 0
8  pour i ← 1 à S − n faire
9      si removedSet_i = faux alors
10         pour chaque arc ∈ G_i faire si removedSet_{arc.dest} = faux alors
11             edgesCountSet_i ← edgesCountSet_i + 1
12         e ← e + 1 ;
13 k ← 1
14 pour i ← 1 à S faire
15     si removedSet_i = faux alors
16         verticesSet_i ← k
17         k ← k + 1
18 G2 ← CreerMatriceCreuse(S - n,e)
19 pour i ← 1 à S faire
20     si removedSet_i = faux alors
21         pour chaque arc ∈ G_i faire si removedSet_{arc.dest} = faux alors
22             AjouterArc(G2, verticesSet_i, verticesSet_{arc.dest},
                1.0/edgesCountSet_{arc.dest})
23         ;
```

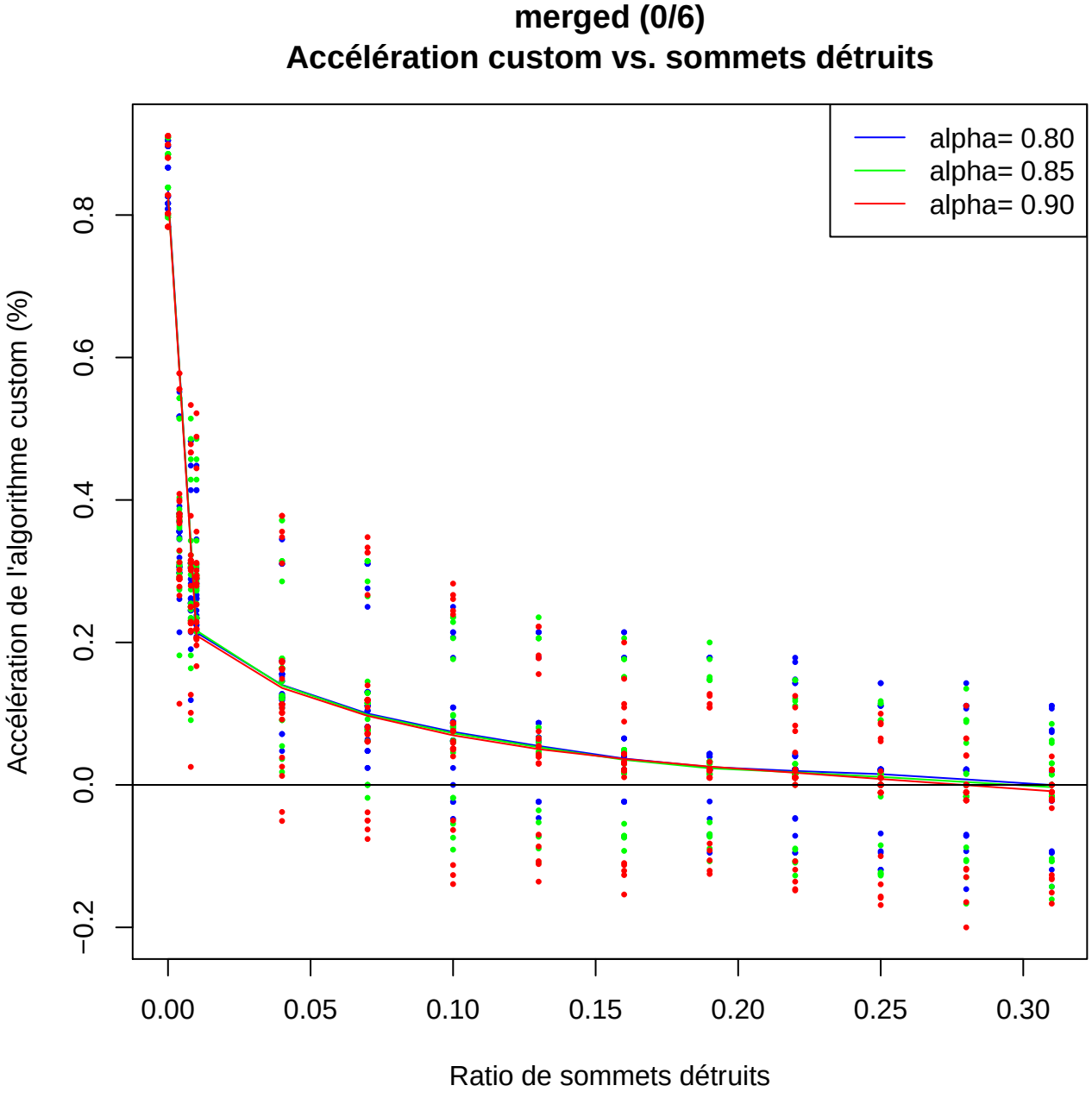


FIGURE 6.1 – Évolution de l'accélération par rapport au nombre de sommets retirés, avec tous les graphes regroupés

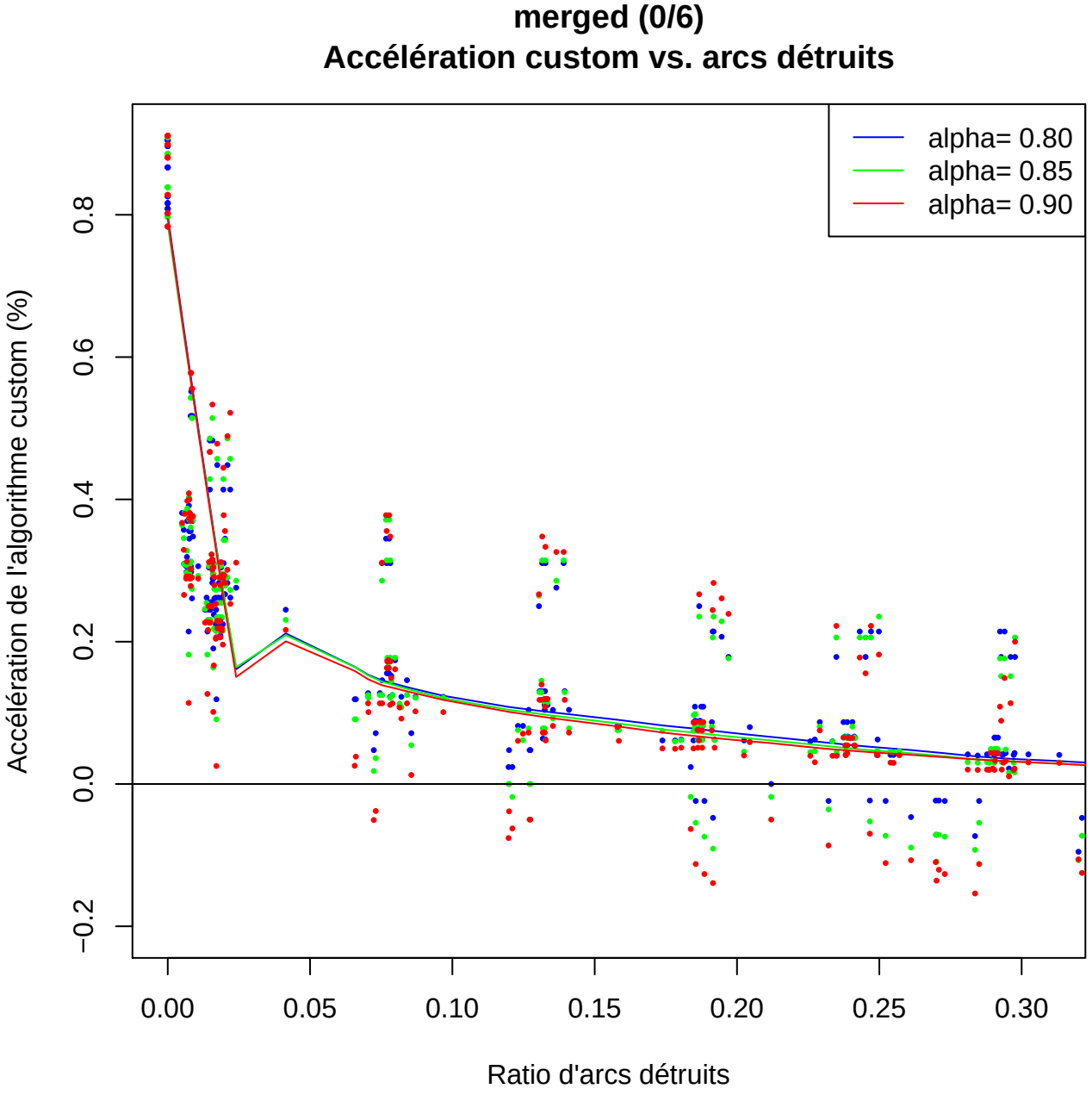


FIGURE 6.2 – Évolution de l'accélération par rapport au nombre d'arcs retirés, avec tous les graphes regroupés

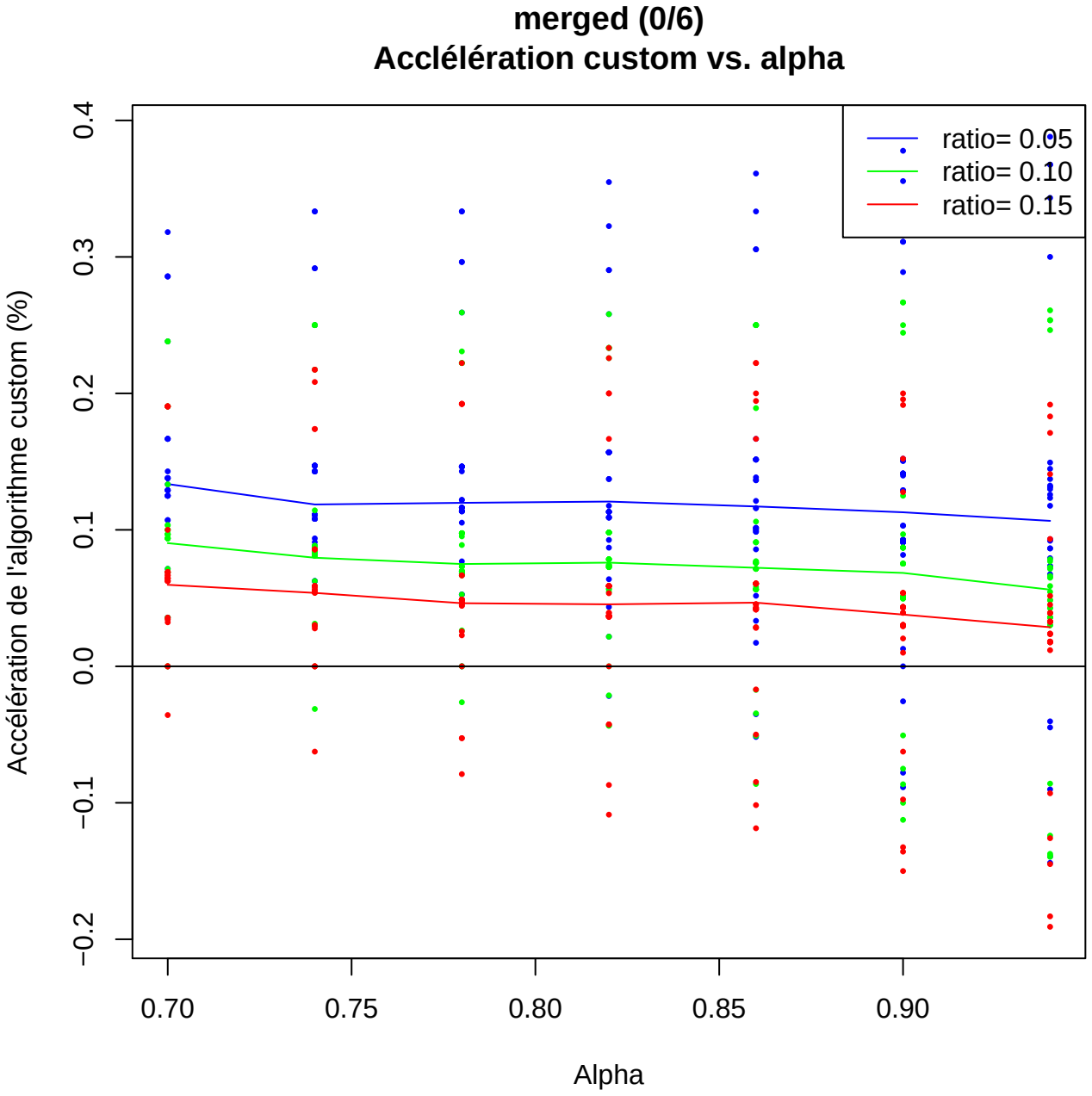


FIGURE 6.3 – Évolution de l'accélération par rapport à $\alpha$, avec tous les graphes regroupés

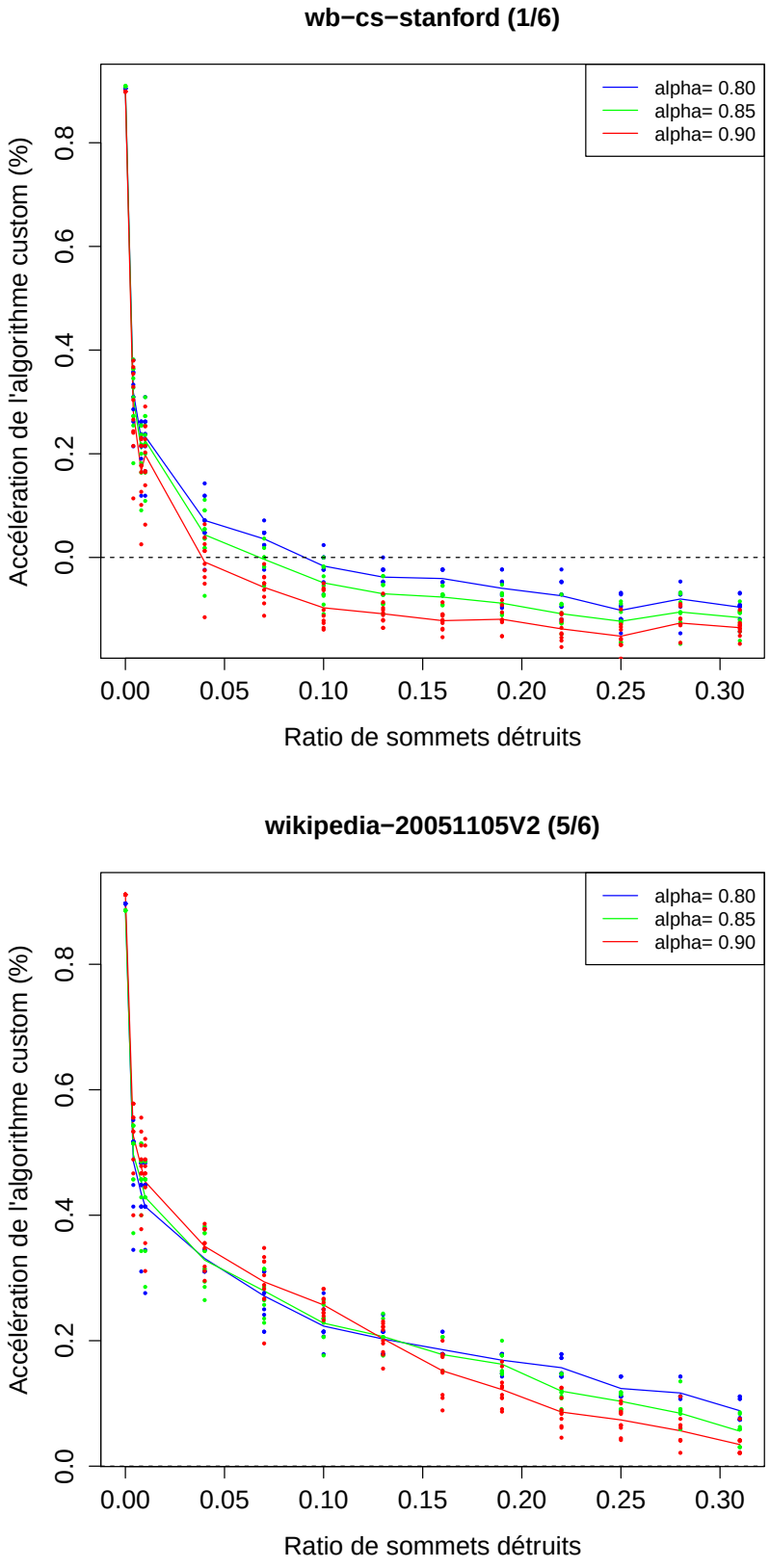


FIGURE 6.4 – Évolution de l'accélération par rapport au nombre de sommets retirés, avec les graphes séparés #1

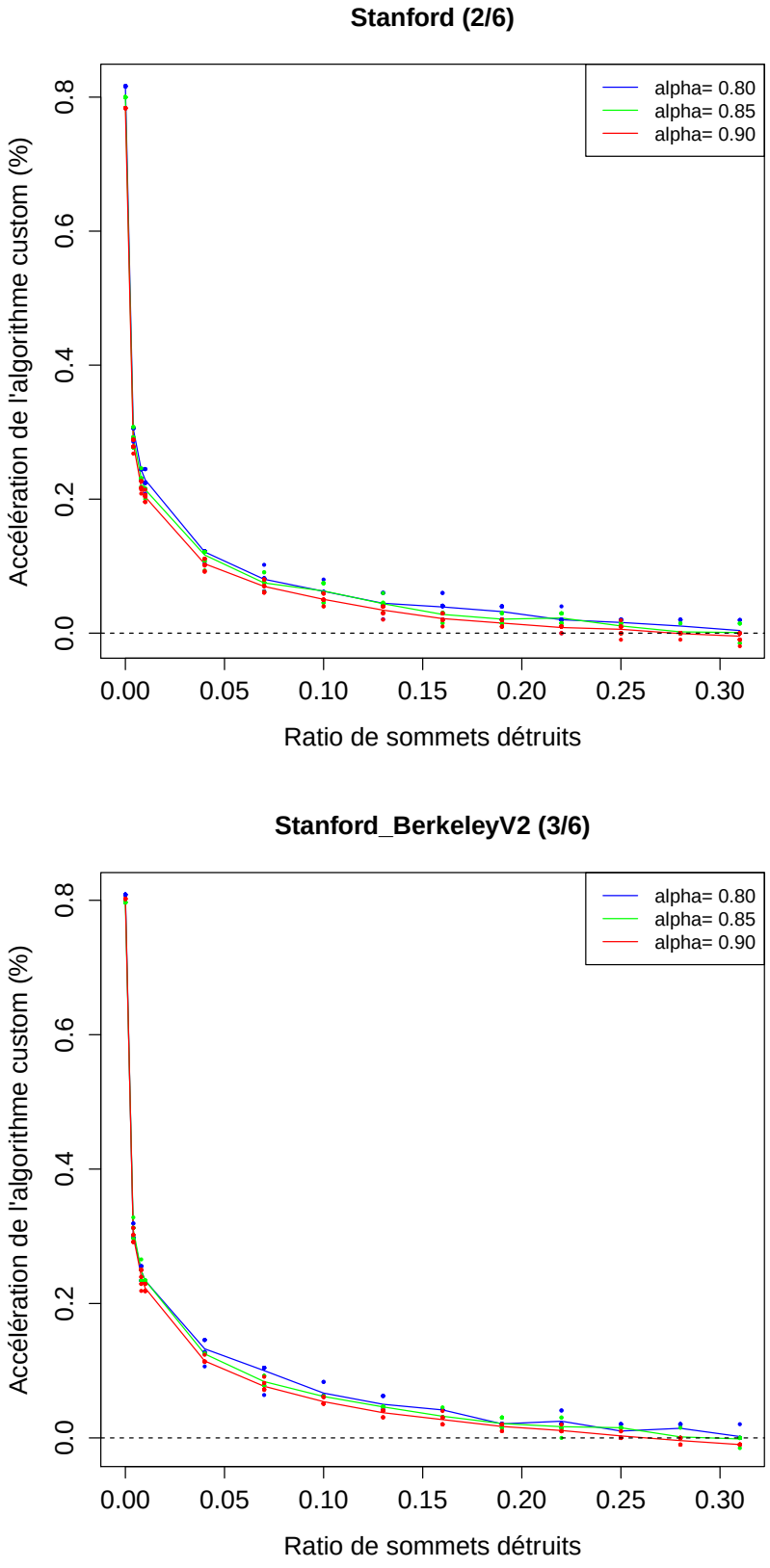


FIGURE 6.5 – Évolution de l'accélération par rapport au nombre de sommets retirés, avec les graphes séparés #2

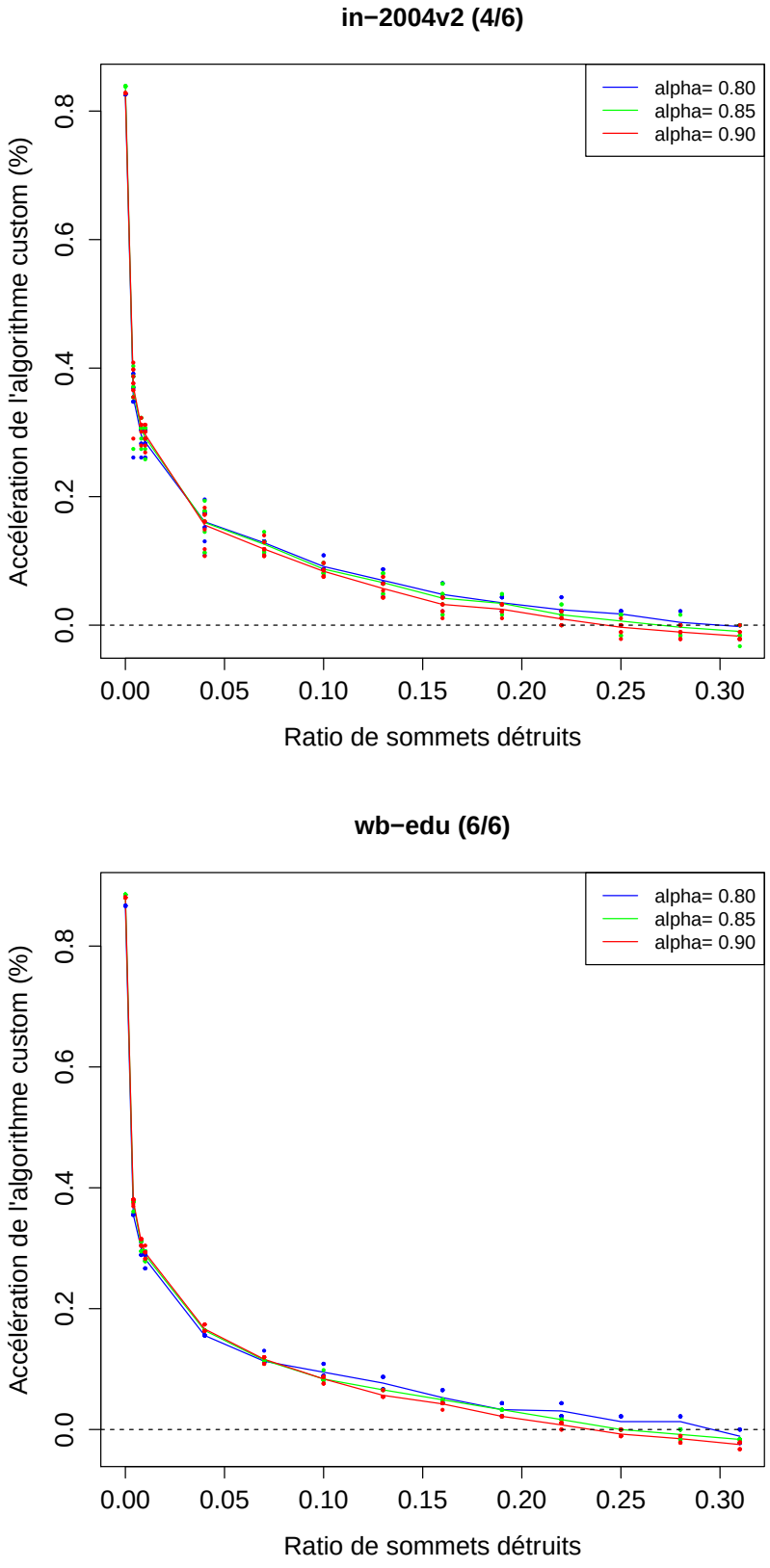


FIGURE 6.6 – Évolution de l'accélération par rapport au nombre de sommets retirés, avec les graphes séparés #3

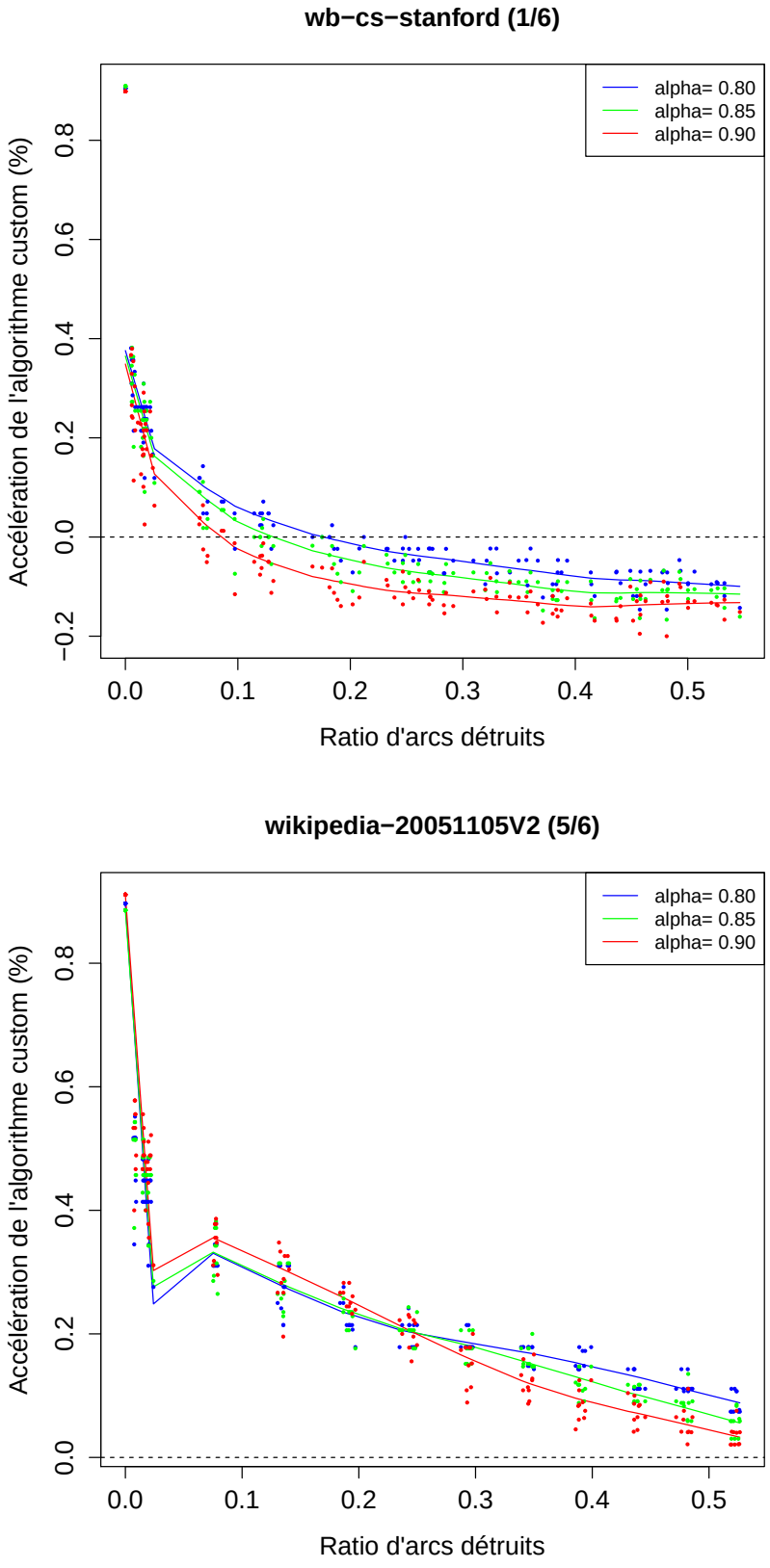


FIGURE 6.7 – Évolution de l'accélération par rapport au nombre d'arcs retirés, avec les graphes séparés #1

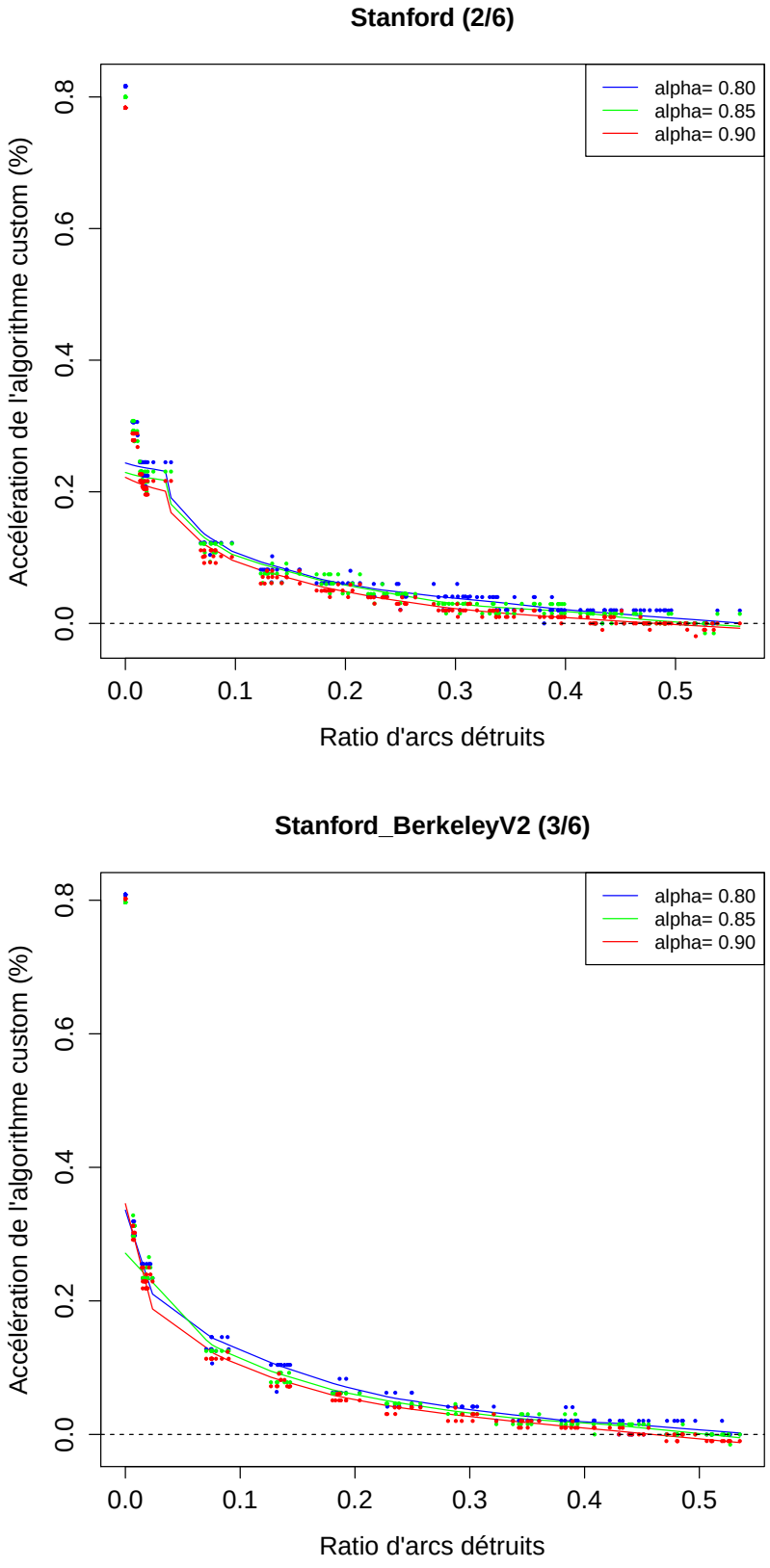


FIGURE 6.8 – Évolution de l'accélération par rapport au nombre d'arcs retirés, avec les graphes séparés #2

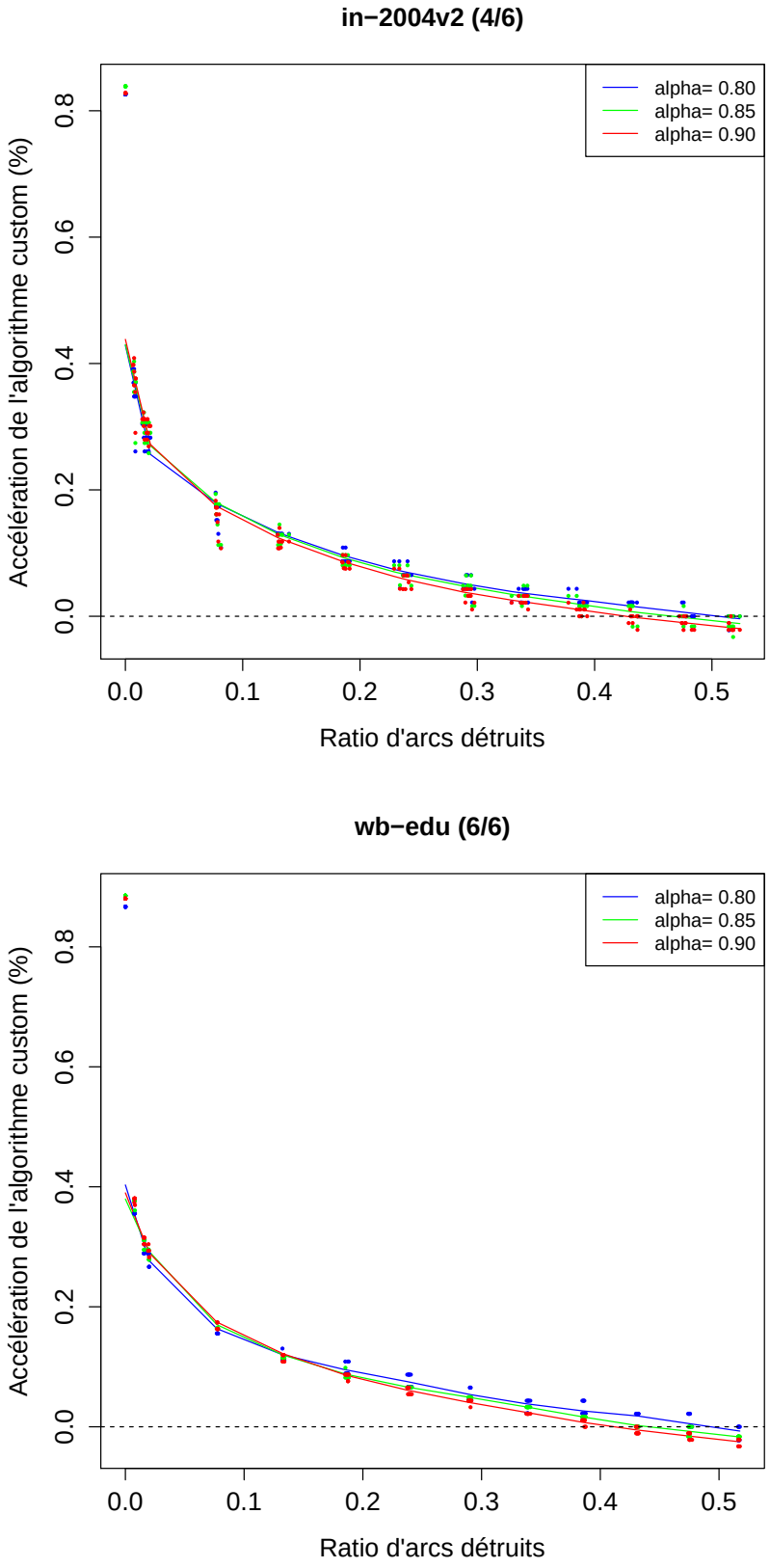


FIGURE 6.9 – Évolution de l'accélération par rapport au nombre d'arcs retirés, avec les graphes séparés #3

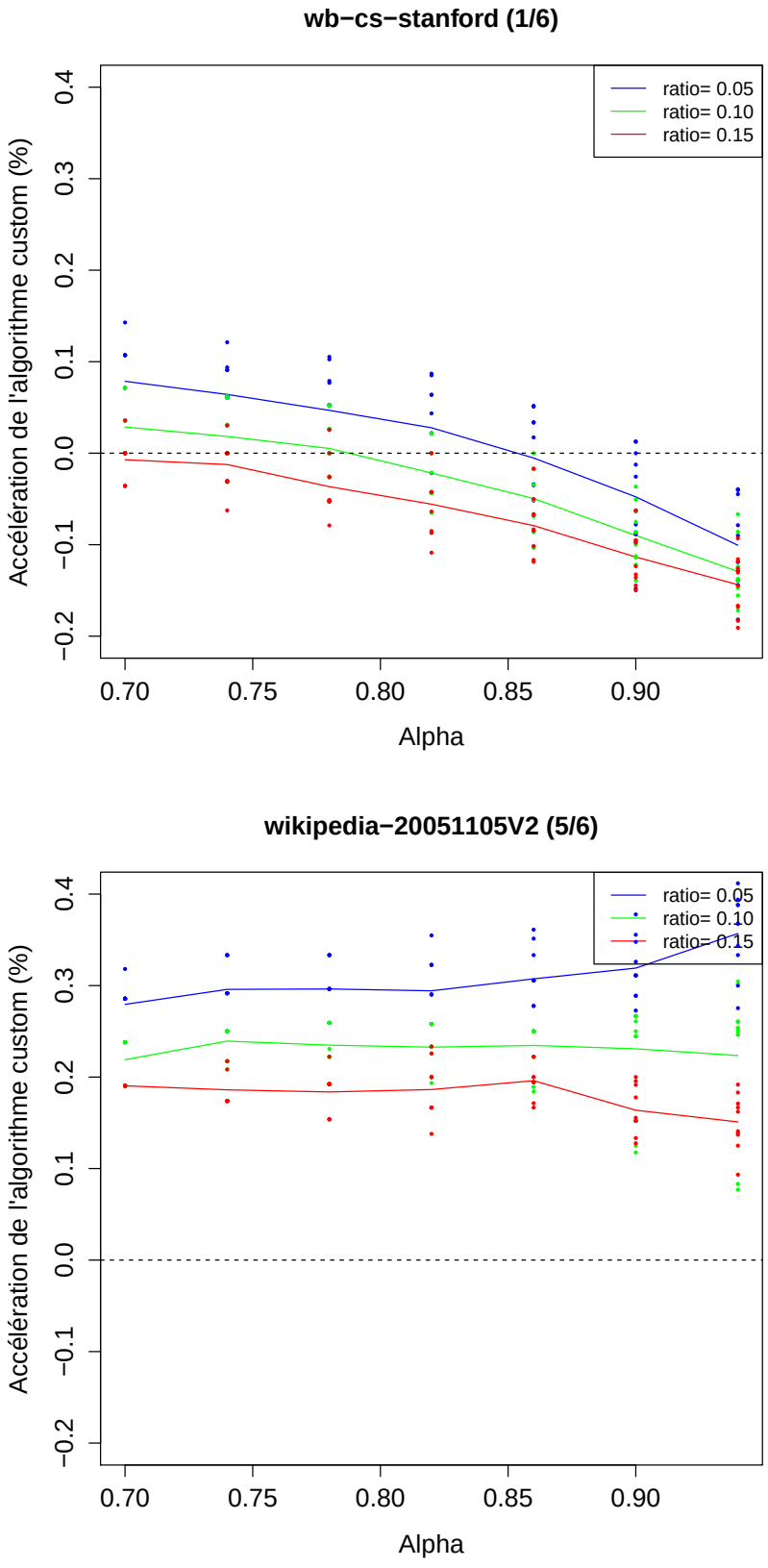


FIGURE 6.10 – Évolution de l'accélération par rapport à $\alpha$, avec les graphes séparés #1

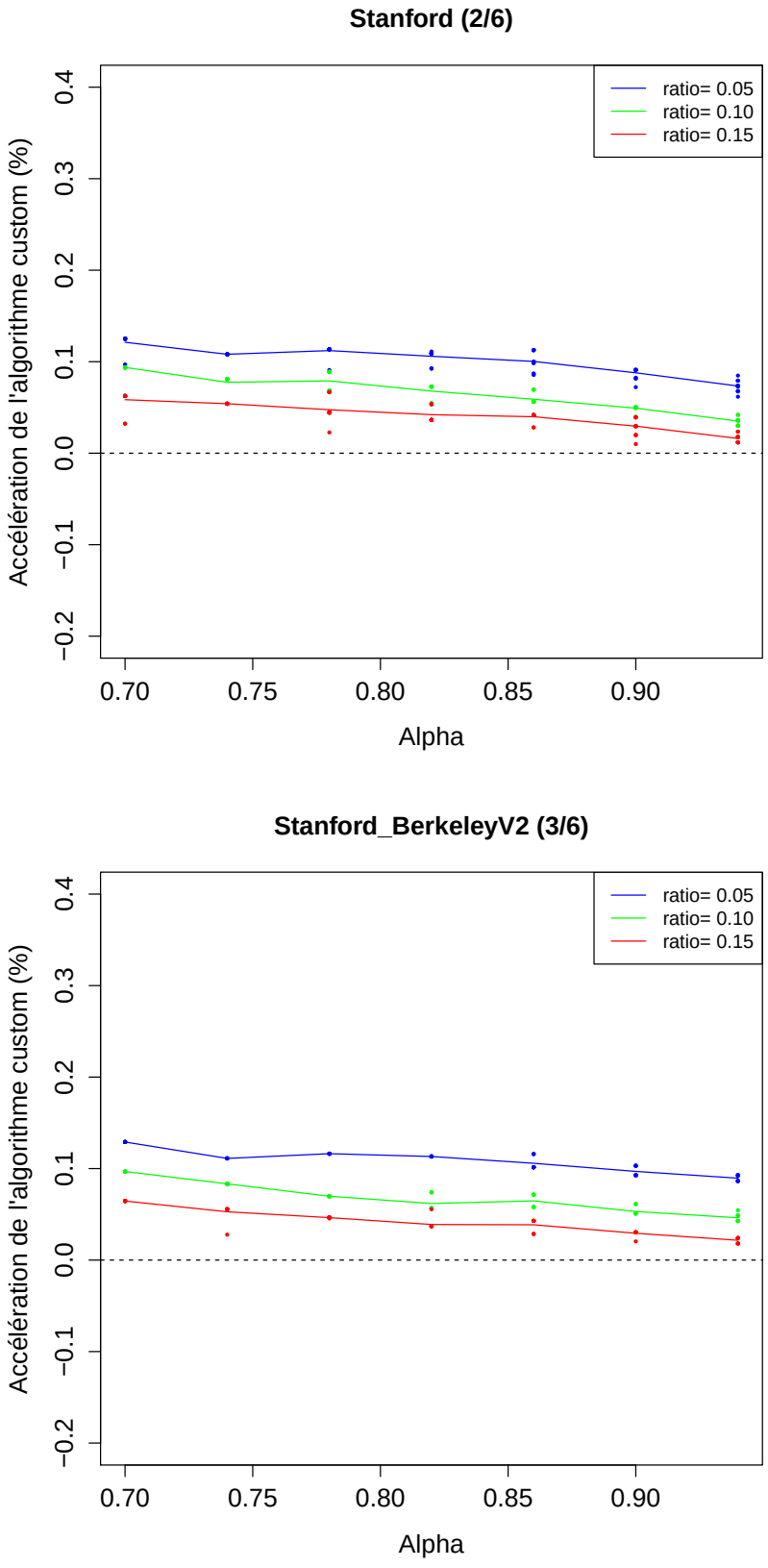


FIGURE 6.11 – Évolution de l'accélération par rapport à $\alpha$, avec les graphes séparés #2

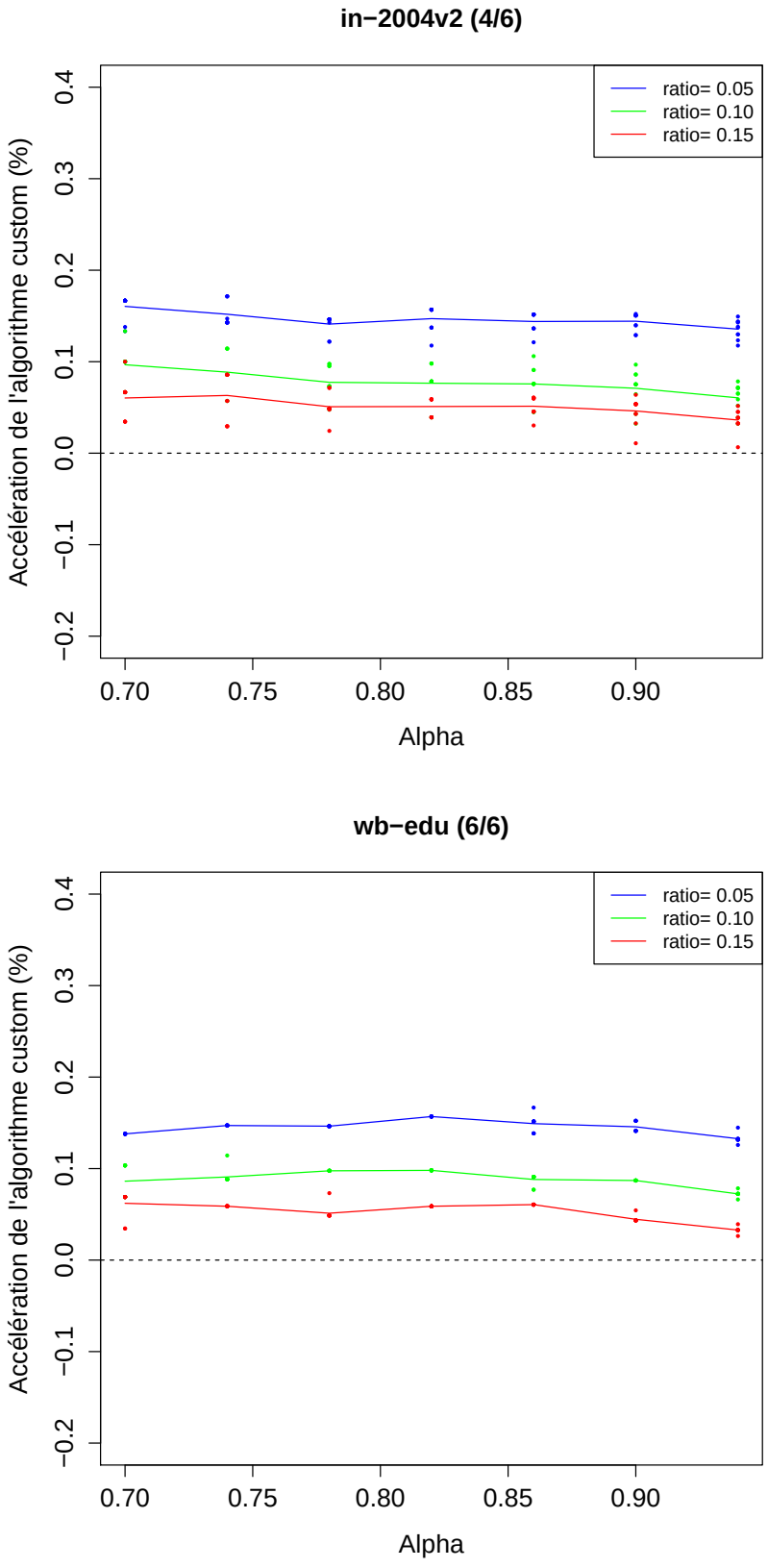


FIGURE 6.12 – Évolution de l'accélération par rapport à $\alpha$, avec les graphes séparés #3