EPITA

SCIA 2021

Le Projet de cette année

(Sparse) Very Large Graphs

Enseignants
Robert Erra
& Alexandre Letois
& Mark Angoustures

Semestre S8

Mai-Juin-Juillet 2021

Commentaires :
(1) Rappel: pour la date limite : 15 juillet 2021 23h59.
(2) Ce document sera mis à jour à chaque question pertinente qui sera posée et après la 4ème et dernière séance.

Table des matières

1 Règles et rendu **3**
- 1.1 Règles . . . 3
- 1.2 Que devez-vous rendre ? . . . 3
- 1.3 Comment et quand ? . . . 3
- 1.4 Contenu du rapport ? . . . 3
- 1.5 Sorties/Outputs . . . 4

2 Projets proposés **4**

3 Remarques **4**
- 3.1 Graphes orientés ou non orientés, pondérés ou non ? . . . 4
- 3.2 La composante connexe géante . . . 4
- 3.3 Quels graphes . . . 4
- 3.4 Quels stockage de graphes ? . . . 5
- 3.5 Format des fichiers graphes . . . 5
- 3.6 Quels sont les objectifs ? . . . 6
- 3.7 Stratégies et tactiques . . . 6
- 3.8 Stratégies . . . 6
- 3.9 Tactiques . . . 6
  - 3.9.1 Projet BE . . . 6
  - 3.9.2 Projet VLS . . . 7

4 Projet BE : l'algorithme de Takes-Kosters pour le calcul de toutes les excentricités **7**
- 4.1 Introduction . . . 7
- 4.2 Fonctionnement . . . 8
  - 4.2.1 Formules préliminaires . . . 8
  - 4.2.2 L'Algorithme *DiameterDB* . . . 8
- 4.3 Code de Takes . . . 9
  - 4.3.1 Description . . . 9
  - 4.3.2 Fonctionnement . . . 9
  - 4.3.3 Format . . . 9
- 4.4 Illustration . . . 10
- 4.5 L'algorithme *BOUNDINGECCENTRICITIES* de Takes et Kosters . . . 10
- 4.6 Projet . . . 10

5 Projet VLS : l'algorithme de calcul de *Very Light Spanner* **11**
- 5.1 Introduction . . . 11
- 5.2 t-spanner . . . 12
- 5.3 La maille d'un graphe et la conjecture d'Erdös . . . 12
- 5.4 Objectif . . . 12
- 5.5 Algorithme . . . 13
- 5.6 Variantes . . . 15
- 5.7 Calcul des communautés d'un graphe : Louvain ou Leiden ? . . . 15
- 5.8 Calcul de communautés . . . 15

6 Liens utiles **16**
6.1 Articles . . . 16
6.2 Code et documentation . . . 16
6.3 Data . . . 16

1 Règles et rendu

Nous détaillerons très précisément cette section après la fin de la dernière séance :

1. ...

En attendant, quelques rappels :

1.1 Règles

1. Vous avez choisi un projet
2. Vous l'avez traité à deux
3. Ce document décrit (i) les projets + (ii) ce que vous devez envoyer ...
4. +(iii) et comment vous devrez le "renvoyer".
5. Date limite :
6. On fixe la date limite au **Jeudi 15 juillet 2021** 23h59.

1.2 Que devez-vous rendre ?

— Un fichier unique (zippé, taré etc.) : rapport + code(s) + script(s) etc.
— Remarque : n'envoyez aucun graphe svp, éventuellement le lien vers le graphe ou les graphes suffira.
— Le rapport (sur la 1ère page) : Nom(s) de famille + Prénom(s) + adresse(s) email + titre + résumé
— Format du rapport ? doc, docx, odt, pdf, latex etc.
— Métrique du rapport ?

1. Minimum : 10000 (dix mille) caractères espaces non compris [hors code]
2. Maximum : 20000 (vingt mille) [hors code]
3. Annexes/images : si vous en avez besoin : no limit

— Toute citation / images téléchargées : donnez les références

1.3 Comment et quand ?

1. Date limite : **Jeudi 15 juillet 2021** 23h59.
2. Un et un seul "dépot" : l'équipe Teams
3. Un seul fichier : son nom doit aussi svp comporter vos noms
4. **Retard : -2 points par jour de retard.**

1.4 Contenu du rapport ?

Votre rapport doit présenter ce que vous avez fait, et les résultats obtenus. Avec un résumé et une conclusion claire.

Vous serez noté sur :

1. ++++ : la qualité de votre code
2. ++++ : la clarté de votre rapport
3. ++++ : la lisibilité de votre rapport
4. +++ : sur la pertinence et l'intérêt des choix que vous aurez faits
5. ++ : vos résultats.

1.5 Sorties/Outputs

Toute information pertinente à la compréhension des résultats. **On précisera aussi cela après la dernière séance pour chacun des projets.**

2 Projets proposés

1. Vous devez choisir **un et un seul projet** parmi les deux projets proposés ici :
   — Projet BE : calcul de toutes les excentricités ; voir la section (4)
   — Projet VLS : calcul rapide d'un spanner "léger" [1] ; voir la section (5).
2. Ce projet remplace l'examen.
3. Il y a des contraintes :
   — Votre code doit être capable de traiter des graphes de millions, voire de dizaines ou de centaines de millions, en un temps *raisonnable*.
   — Vous pouvez vous inspirez de codes existants (voir la section (6)) mais vous devez utiliser la bibliothèque *Igraph* (en C) ; voir la section (6)
4. Enfin : c'est un travail de groupe : faites des binômes svp. (Si un groupe de 3 étudiant.e.s voudrait travailler ensemble : ok mais vous faites les 2 projets !).

3 Remarques

3.1 Graphes orientés ou non orientés, pondérés ou non ?

On s'intéresse en priorité aux graphes **non orientés** et **non pondérés**. La raison est que :

1. Si on a un graphe pondéré, et non orienté, le BFS ne fontionne pas, il faut passer à Dijsktra.
2. Si on a un graphe non orienté, le BFS reste encore possible, mais l'inégalité triangulaire n'étant plus vraie, il faut presque tout changer dans l'approche.

Donc, attention si vous prenez des exemples sur SNAP ou autre de bien vérifier leur "statut".

3.2 La composante connexe géante

De nombreux graphes réels ne sont pas connexes mais ont souvent une composante connexe principale (la plus grande) très large. On parle alors de *composante connexe géante*.

Il peut donc être utile de faire un *préconditionnement* de vos graphes :

1. On calcule la *composante connexe géante* via Igraph.
2. On la stocke une fois pour toute.

Cela fait gagner du temps et vous permettra d'avoir des résultats "lisibles".

Le code diam.c permet de faire cela très rapidement, avec juste un peu de modifications. Utile donc pour **tous** les projets.

3.3 Quels graphes

Sur la page http://data.complexnetworks.fr/Diameter/ on trouve 4 graphes, dont voici une brève description [2] :

1. *Very Light Spanner* .
2. Pour plus de détails sur ces graphes : lire l'article *Measuring Fundamental Properties of Real-World Complex Networks*, Matthieu Latapy & Clémence Magnien.

1. Inet – Internet IP topology (via traceroute) : plus de 1,7 millions de sommets, plus de 11 millions de liens
2. IP – échanges entre des ordinateurs : plus de 2,2 millions de sommets, plus de 19 millions de liens
3. P2P – exchanges between P2P users : plus de 5,8 millions de sommets, plus de 140 millions de liens
4. Web – links between web pages plus de 40 millions de sommets, plus de 780 millions de liens.

Sinon :

— Le site (fantastique) de l'équipe Lasagne : https://www.pilucrescenzi.it/wp/software/lasagne/ qui a migré, mais on y trouve encore de tout, des codes, des articles, des graphes etc.

Une très bonne "base" de graphes Stanford Network Analysis Project : http://snap.stanford.edu/

3.4 Quels stockage de graphes ?

Pour chaque fichier de "graphe" que vous téléchargerez, vérifiez quel type de stockage est utilisé : en clair, comment est géré la "sparsity" (la *creusité* ce n'est pas très beau).

Dans certains fichiers la 1ère ligne contient le nombres de sommets, dans certains fichiers on a juste la liste des arcs. Certains programmes ou bibliothèques attendent un format *binaire*.

Si vous avez un graphe dans le "mauvais" format, vous pouvez songer à utilisez Gconvert (ça règle certains des problèmes mais pas tous, hélas) :

https://www-complexnetworks.lip6.fr/~latapy/PP/gconvert.html

Dans le cas contraire, vous pouvez écrire votre propre petit code de conversion en C/C++.

3.5 Format des fichiers graphes

Attention ! Il existe de nombreux formats de fichiers de graphes "creux".

Par exemple, le format utilisé par C. Magnien est particulier :

1. La 1ère ligne comporte un entier n : c'est nombre de sommets dans le graphe.
2. Les n lignes suivantes lignes indiquent le degré de chacun des sommets du graphe.
3. Chaque ligne suivante représente une arête entre un sommet a et un sommet b.

Exemple :

```
3
0 2
1 2
2 2
0 1
0 2
2 1
(3 nodes, thus numbered from 0 to 2, node 0 has degree 2, node 1
has degree 2, and node 2 has degree 2 too, and the links are 0 1,
0 2 and 2 1)
```

On trouve aussi comme format :

```
0 1
0 2
2 1
(3 nodes, list of links: 0 1, 0 2 and 2 1)
```

3.6 Quels sont les objectifs ?

En quelques mots :
— Explorer quelques problèmes sur les graphes géants
— (Re)coder à l'aide de la bibliothèque Igraph (en C) un ou plusieurs algorithme(s)
  1. en suivant la *statégie* des auteurs des articles cités (pour le projet BE).
  2. en suivant des statégies et tactiques à explorer pour le projet VLS.
— Explorer d'autres tactiques pour accélérer le calcul.
— Se faire une *culture* sur les **graphes géants creux**, appelés aussi *graphes de terrain* car issus du monde réel.

3.7 Stratégies et tactiques

3.8 Stratégies

Soit G un graphe $G = (V, E)$. Pour les deux projets la **statégie** est la même :

1. On choisit $S = \{s_1, s_2 \cdots s_l\}$ un sous-ensemble de k sommets de V : $S \subset V$.
2. *on choisit un sommet u dans cette liste $S \subset V$* (voir "Tactiques plus haut).
3. On calcule l'arbre BFS(u) (avec u comme racine)
4. on *l'utilise*.
5. On supprime u de S, ce qui donne $S = S - \{u\}$.
6. On choisit un autre sommet v de S, on va en (2).
7. etc.

La taille de S donnera le nombre maximum de BFS et donc le "coût" de l'algorithme. **Ceci est valable pour les 2 projets.**

3.9 Tactiques

Par contre, et ceci est vrai pour presque tous les algorithmes approchés sur les graphes géants creux, ce qui différencie les algorithmes c'est la tactique pour choisir $S \subset V$.

3.9.1 Projet BE

Quelques tactiques de choix de S sont possibles :
— On choisit S dès le départ, par exemple dans le cas où on "connaît" très bien le graphe.
— On choisit les sommets de S *au fur et à mesure*, en fonction des résultats des BFS précédents.
— On fait un mix des deux précédentes tactiques.

Comment choisir les sommets de $S = \{s_1, s_2, s_3, \cdots\}$ *au fur et à mesure* ?

1. Choisir les sommets s_i au hasard
2. Choisir les sommets s_i parmi ceux de haut degré.
3. Choisir les sommets s_i parmi ceux de bas degré.

4. On démarre l'algorithme de Takes-Kosters, et on choisit les sommets s_i de "grand" delta [diff entre les deux borne.
5. Plus intéressant (c'est ce qu'on vous suggère de tester) : on calcule la décomposition en commununautés de G, et on choisit des sommets dans les plusieurs communautés [la plus grand + la plus petite, etc.]
6. Évidemment, on peut (et on doit en général) mixer les précédentes tactiques.

3.9.2 Projet VLS

Pour ce projet, ce qu'on a dit précédemment sur l'autre projet reste tout à fait valable. Mais il y a un problème supplémentaire : comment fixer le critère d'arrêt ?

Donnons un exemple naïf :

1. on choisit $u = s_1$ au hasard,
2. on tombe sur un des points diamétraux
3. On calcule le BFS(u)
4. Résultat : l'arbre BFS a le diamètre du graphe G.

Donc, si on ne prend comme critère que le diamètre, on risque de louper des arêtes intéressantes.

Suggestions :

1. Imposer un minimum d'arêtes au sous-graphe H (le spanner). On peut donner un pourcentage comme paramètre d'entrée du programme.
2. Imposer que H ait un rayon proche de celui de G.
3. On peut faire quelques itérations de l'algorithme de Takes-Kosters (BE).

Une tactique très prometteuse consiste à imposer au graphe spanner d'avoir une valeur de la "célèbre" fonction de modularité Q.

4 Projet BE : l'algorithme de Takes-Kosters pour le calcul de toutes les excentricités

4.1 Introduction

L'algorithme *Bounding Diameters* ou *diameterBD* de Takes et Kosters a été créé pour calculer non seulement le diamètre exact d'un VLG mais peut s'adapter pour le calcul de toutes les excentricité et s'appelle alors l'algorithme *Bouding Eccentricities*. On a ainsi deux algorithmes très proches :

1. Algorithme BoundingDiameter (BD) :
2. Algorithme BoundingEccentricities (BE).

Un code implémentant cet algorithme en C++ peut être trouvé sur le github de Frank Takes. Des liens vers les articles et le code sont disponibles dans la section Liens utiles. Cet algorithme est une amélioration de l'algorithme déjà existant *All Pairs Shortest Path* (ou APSP) trouvé par RW Floyd en 1962. Il se base sur les bornes supérieures et inférieures du diamètre ainsi que la sélection et l'élagage de sommets spécifiques pour arriver au résultat final dans un délai court. Pour ce faire, on utilise certaines propriétés de l'excentricité pour retirer au fur et à mesure des calculs les sommets qui ne peuvent pas nous rapprocher de la valeur exacte du diamètre. Pour rappel, un graphe n'est pas toujours connexe, mais le diamètre n'a de sens que dans le cadre d'une composante connexe. Il est donc important d'utiliser des graphes connexes (ou de calculer la plus grande composante connexe du graphe). Nous considérerons dans la suite que les graphes utilisés sont connexes.

4.2 Fonctionnement

4.2.1 Formules préliminaires

Une formule intéressante pour les déductions d'excentricités est :

$$max_{v \in V} \mathcal{E}cc_L(v) \leq \Delta(G)) \leq min(max_{v \in V} \mathcal{E}cc_U(v), 2 * min_{v \in V} \mathcal{E}cc_U(v) \quad (1)$$

Pour plus de lisibilité plus tard, on remplace la formule précédente par celle-ci :

$$\Delta_L(S) \leq \Delta(G) \leq \Delta_U(S). \quad (2)$$

où on a :

- $S \subset V$: un sous-ensemble des sommets de G
- $\Delta(G))$: La valeur du diamètre de G
- $max_{v \in V} \varepsilon_L(v)$: la valeur maximale de toutes les bornes inférieures des excentricités des points $v \in V$
- $max_{v \in V} \varepsilon_U(v)$: la valeur maximale de toutes les bornes supérieures des excentricités des points $v \in V$
- $min_{v \in V} \varepsilon_U(v)$: la valeur minimale de toutes les bornes supérieures des excentricités des points $v \in V$
- $\Delta_L(S)$: La borne inférieure du diamètre calculée sur l'ensemble des sommets de S
- Δ_U : La borne supérieure du diamètre calculée sur l'ensemble des sommets de S.

L'algorithme se base sur les excentricités d'un graphe $G(V, E)$ (contenant V sommets et E arêtes) :

- Soit l'excentricité $\mathcal{E}cc(v)$ d'un sommet $v \in V$
- Soit un sommet w situé à une distance k de v : $d(v, w) = k$
- Alors $\mathcal{E}cc(v) - k \leq \mathcal{E}cc(w) \leq \mathcal{E}cc(v) + k$

Avec cette formule, on peut déduire une approximation avec une borne inférieure et une borne supérieure de l'excentricité de tous les sommets connexes à v et plus généralement, on peut déduire que pour **tout** sommet w :

$$\max(\mathcal{E}cc(v) - d(v, w), d(v, w)) \leq \mathcal{E}cc(w) \leq \mathcal{E}cc(v) + d(v, w). \quad (3)$$

4.2.2 L'Algorithme *DiameterDB*

Cet algorithme prend en entrée un graphe $G = (V, E)$ et donne en sortie :

1. la valeur du diamètre de G
2. l'excentricité de chaque sommet.

Il fonctionne de la façon suivante :

1. Mise en place des valeurs de base
   - On stocke tous les sommets de V dans W
   - La valeur de Δ_L est mise à $-\infty$
   - La valeur de Δ_U est mise à $+\infty$
   - Pour tout point $w \in W$, l'excentricité minimale $\epsilon_L(w)$ est mise à $-\infty$
   - Pour tout point $w \in W$, l'excentricité maximale $\epsilon_U(w)$ est mise à $+\infty$
2. On boucle tant que Δ_L et Δ_U sont différents ou que W est non-vide
   - On sélectionne v, un sommet de W (différentes stratégies possibles pour la sélection)
   - On calcule l'excentricité de v

— On recalcule Δ_L qui est la valeur max entre Δ_L et $\epsilon(v)$
— On recalcule Δ_U qui est la valeur min entre Δ_U et $2 * \epsilon(v)$
— On recalcule les excentricités avec une boucle sur tous les sommets $w \in W$
  — $\epsilon_L(w)$ est le max entre :
    (a) $\epsilon_L(w)$
    (b) le max entre $\epsilon(v) - d(v, w) et d(v, w)$
  — $\epsilon_U(w)$ est le min entre $\epsilon_U(w)$ et $\epsilon(v) + d(v, w)$
  — Elagage d'un sommet en retirant w de W si une des deux conditions suivante est remplie :
    (a) $\epsilon_U(w) \leq \Delta_L$ **et** $\epsilon_L(w) \geq \Delta_U/2$
    (b) $\epsilon_L(w) = \epsilon_U(w)$

3. On retourne Δ_L, la valeur exacte du diamètre

4.3 Code de Takes

4.3.1 Description

Frank Takes fournit le code (*Teexgraph*) qui a été utilisé pour les expérimentations. Un lien vers ce dernier est disponible plus tard dans la section "Liens utiles". À noter que le code utilisé a une limite MAXN de 10M de sommets hard-codée.

Pour utiliser le code de Takes, utilisez **make**. Ensuite, avec une ligne de commande vous pourrez utiliser *DiameterDB* :

1. ./teexgraph [lien du fichier] : *Lance l'algorithme sur le fichier*
2. ./teexgraph [lien du fichier] nb_sommets : *Lance l'algorithme sur le fichier en modifiant la valeur de MAXN par celle de nb_ sommets (Cela dit, cette fonctionnalité n'a pas l'air de fonctionner).*

4.3.2 Fonctionnement

1. Le script vérifie si le graphe est orienté (directed) ou non orienté (undirected) :
   — Pour qu'un graphe soit non orienté, il faut que le nombre de lignes soit égal au double du nombre d'arêtes
   — Si un graphe n'est pas considéré comme non orienté, il est modifié pour le devenir en ajoutant des arêtes "retour" à toutes les arêtes qui n'en ont pas déjà
   — Les arêtes sont triées et les doublons sont retirés.
2. Le script calcule le WCC (Weakest Connected Component) du graphe, ce qui revient à calculer les composantes connexes du graphe lorsque celui-ci est orienté. **Pour ce projet il est vivement conseillé de ne pas traiter que des graphes** *non orientés.*
3. Des stats sur le graphe sont affichées
4. Le script calcule le diamètre du plus grand WCC avec *DiameterDB.*

4.3.3 Format

— Le script lit les fichiers sous format edgelist
— Les premières lignes qui ne commencent pas par un entier sont ignorées
— Lors de la création du graphe, le script essaye d'ajouter une arête à chaque ligne
— Pour ajouter une arête, le script check si il y a bien 2 entiers sur la ligne

— Si c'est le cas, vérifie que les entiers sont acceptables (> 0 et < 10M, sauf si le max a été changé)
— Vérifie que l'arête n'est pas une self-loop
— Ajoute l'arête si tout est bon et met à jour le nombre d'arêtes/compte de sommets
— Si on ne peut pas ajouter une arête, la ligne est ignorée et le nombre d'arêtes ignorées est incrémenté.

4.4 Illustration

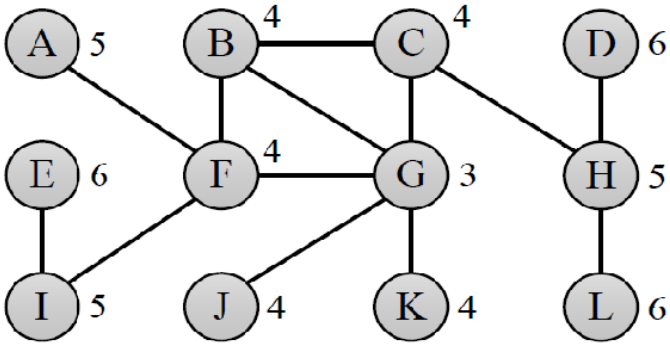


FIGURE 1 – Un exemple "jouet" des excentricités.

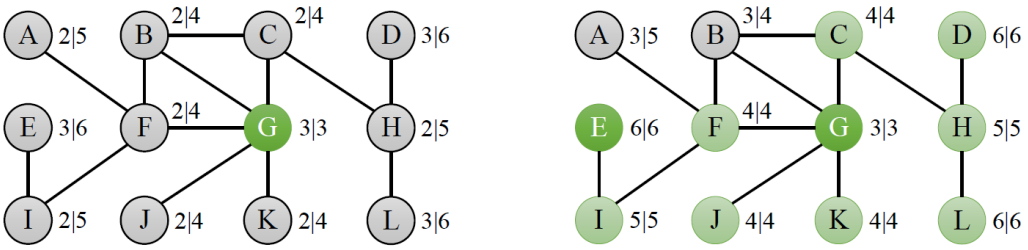


FIGURE 2 – Les bornes inférieures et supérieures de l'excentricité de chaque sommet après avoir calculé l'excentricité du sommet G (gauche) et E (à droite).

4.5 L'algorithme *BOUNDINGECCENTRICITIES* de Takes et Kosters

Voir figure (4.5).

4.6 Projet

Il vous est demandé de coder l'algorithme *BOUNDINGECCENTRICITIES* de Takes et Kosters en utilisant la bibilothèque *Igraph*. Dès lors que les groupes auront été constitués et que chaque groupe aura fait son choix, nous préciserons ce que vous aurez à faire et à rendre.

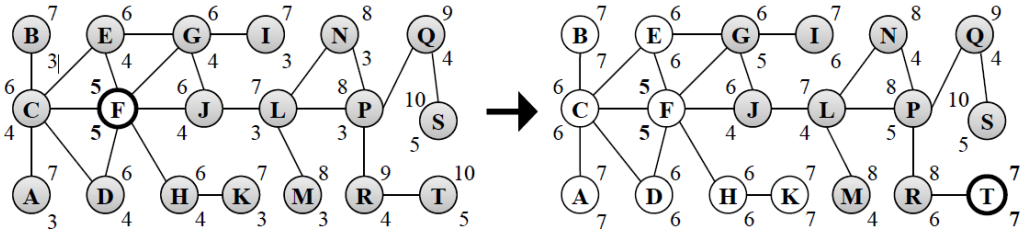


Figure 1: The path B-T realizes a diameter of length 7. Numbers beneath and above nodes denote lower an upper eccentricity bounds after first calculating the eccentricity of node F (left) and then node T (right).

FIGURE 3 – Les images de cette section sont extraites des articles de Takes et Kosters.

Algorithm 2 BOUNDINGECCENTRICITIES

Input: Graph $G(V, E)$
Output: Vector ε, containing $\varepsilon(v)$ for all $v \in V$
$W \leftarrow V$
for $w \in W$ **do**
  $\varepsilon[w] \leftarrow 0 \quad \varepsilon_L[w] \leftarrow -\infty \quad \varepsilon_U[w] \leftarrow +\infty$
end for
while $W \neq \emptyset$ **do**
  $v \leftarrow$ SELECTFROM(W)
  $\varepsilon[v] \leftarrow$ ECCENTRICITY(v)
  for $w \in W$ **do**
    $\varepsilon_L[w] \leftarrow \max(\varepsilon_L[w], \max(\varepsilon[v] - d(v, w), d(v, w)))$
    $\varepsilon_U[w] \leftarrow \min(\varepsilon_U[w], \varepsilon[v] + d(v, w))$
    if $(\varepsilon_L[w] = \varepsilon_U[w])$ **then**
      $\varepsilon[w] \leftarrow \varepsilon_L[w]$
      $W \leftarrow W - \{w\}$
    end if
  end for
end while
return ε

FIGURE 4 – L'algorithme de Takes et Kosters pour le calcul de toutes les excentricités.

5 Projet VLS : l'algorithme de calcul de *Very Light Spanner*

5.1 Introduction

Calculer le diamètre d'un graphe $G = (V, E)$ avec $|V| = n$ et $|E| = m$, peut-être très facile suivant le type du graphe. Par exemple, pour un graphe complet (ou clique) le diamètre est de 1 car tous les sommets sont connectés entre eux. Cependant nous ne savons pas toujours à l'avance si un graphe est complet ou non. Pour cela, il faut calculer tous les plus courts chemins entre tous les sommets pair à pair (All-pairs shortest paths = APSP) dont la complexité en temps est de l'ordre de $O(n.m)$; ce qui donne une complexité en temps de $O(n^3)$ ou $O(n^2)$ si $m = O(n)$. Le problème APSP est donc de complexité prohibitive lorsqu'on s'attaque à des problèmes sur

les graphes géants creux contenant des millions de sommets et d'arêtes.

Cependant il n'est pas nécessaire de calculer tous les chemins entre tous les points entre eux pour connaître le diamètre d'un graphe. Peut-être qu'une "compression" du graphe G peut suffire à nous donner une approximation du diamètre ?

Ceci nous pousse à chercher si on peut construire un sous graphe H de G contenant tous les sommets de G dont on peut diminuer le nombre d'arêtes mais avec lequel on peut plus facilement approximer le diamètre de G.

Un sous-graphe H qui couvre l'ensemble des sommets de G avec un nombre d'arêtes inférieur ou égal à G s'appelle un **spanner**. Formellement, un spanner H a les propriétés suivantes :

— $H \subseteq G$ (H est inclus dans G)
— $V(H) = V(G)$ (le graphe H couvre tous les sommets de G)
— $E(H) \subseteq E(G)$ (l'ensemble des arêtes de H est inclus dans G)

5.2 t-spanner

Soit un graphe connecté $G = (V, E)$, et soit $H \subset G$ un sous graphe de G, on dit que H est un spanner de *dilatation*[3] t (ou un *t-spanner*) si :

$$\forall u, v \in V : d_H(u, v) \leq t \times d_G(u, v). \tag{4}$$

Il existe de nombreuses notions d'un sous-graphe t-spanner :

— spanner additif : $\forall u, v \in V : d_H(u, v) \leq d_G(u, v) + \beta$
— spanner linéaire : $\forall u, v \in V : d_H(u, v) \leq \alpha \times d_G(u, v) + \beta$
— spanner t-diamètre : $\Delta(H) = t \times \Delta(G)$

C'est ce dernier type de graphe spanner qui nous intéresse mais avec quelques contraintes suppplémentaires.

Peleg and Schäffer ont proposé la notion de t-spanner bien que l'idée soit implicitement présente dans des travaux publiés par Awerbuch et Chew. Peleg and Schäffer ont montré que pour un graphe non pondéré G, déterminer s'il existe un t-spanner de G contenant au plus k arêtes (k donné) est un problème NP-complet. **D'où la nécessité d'voir des algorithmes approchés**.

5.3 La maille d'un graphe et la conjecture d'Erdös

En théorie des graphes, la *maille*[4] d'un graphe est la longueur du plus court de ses cycles.

Erdös a proposé la conjecture suivante :

An undirected unweighted graph G $G = (V, E)$ of girth $> t + 1$ has no proper subgraph that is a t-spanner.

5.4 Objectif

Plusieurs projets :

— Construire un spanner H dont le diamètre approxime celui de G
$|diam(H) - diam(G)| \leq \epsilon$
— Construire un spanner H qui approxime le chemin diamétral de G
$P_{diam(H)} \subseteq P_{diam(G)}$

3. *Stretch factor.*
4. *Girth.*

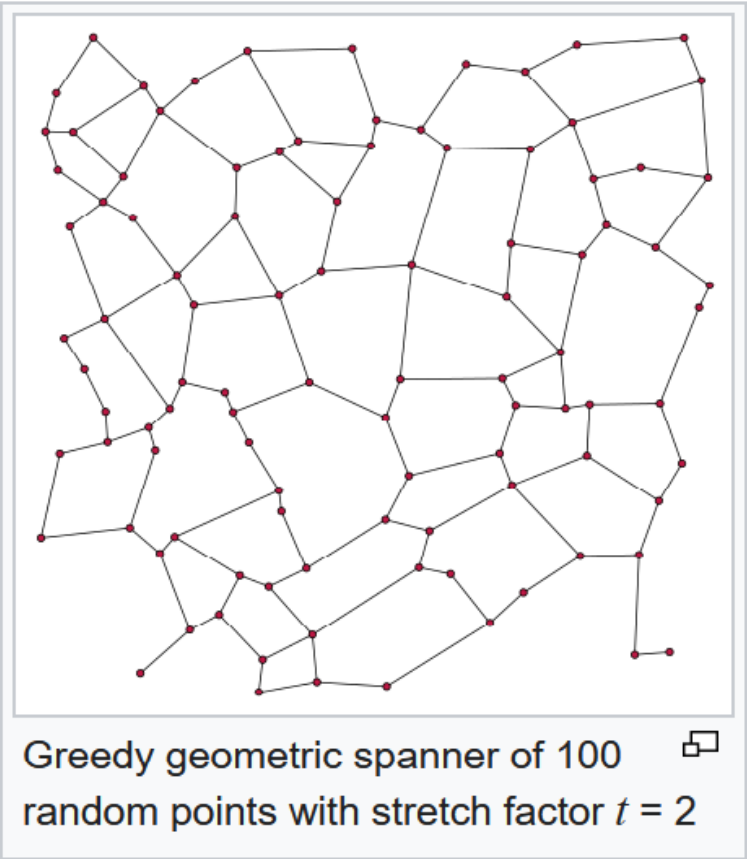


FIGURE 5 – Le graphe 2-spanner obtenu par un algorithme vorace (greedy) à partir d'un graphe complet (100 points pris au hasard dans le plan) ; .

5.5 Algorithme

Soit un graphe G avec $V(G)$ et $E(G) \subseteq V(G) \times V(G)$. L'idée est de construire un spanner le plus **creux** [5] possible avec des outils simples (BFS) et d'une complexité faible (soit : linéaire, en $O(n)$ si n est le nombre de sommets ou en $O(m)$ si m est le nombre de liens)).

1. On choisit $S = \{s_1, s_2 \cdots s_l\}$ un sous-ensemble de k sommets de V : $S \subset V$.
   (Il sera demandé de réfléchir à une tactique (ou plusieurs) du choix des sommets S autre qu'un choix aléatoire).
2. On initialise H possèdant tous les sommets de G mais aucune arête.
   $H = (V, \emptyset)$
3. Pour $s \in S$,
   (a) On calcule $BFS(s_i)$ l'arbre de parcours en largeur du sommet ayant s_i comme sommet source.
   (b) On fusionne H avec $BFS(s_i)$
   $E(H) = E(H) \cup E(BFS(s_i))$
   (c) On calcule la différence entre les bornes inf et sup des l'excentricité de chaque sommet avec l'algorithme de Takes-Kosters.

5. *Sparse.*

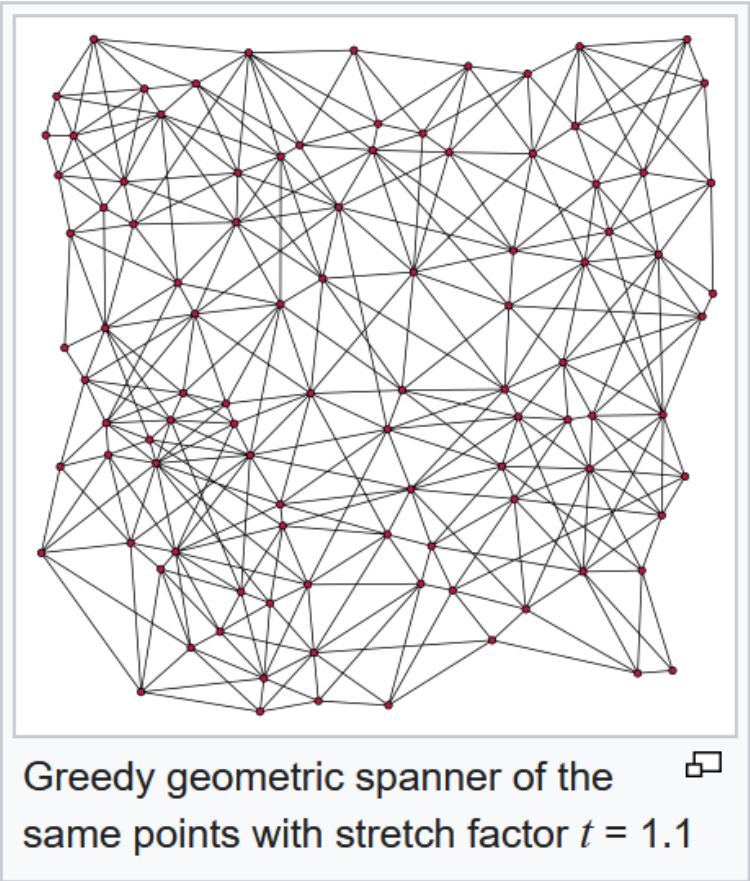


FIGURE 6 – Le graphe 1.1-spanner obtenu par un algorithme vorace (greedy) à partir du même graphe complet (100 points pris au hasard dans le plan) que la figure 5.2.

$$\mathcal{E}cc_{inf} = \begin{pmatrix} \mathcal{E}cc_{inf}(v_1) \\ \cdots \\ \mathcal{E}cc_{inf}(v_n) \end{pmatrix}$$

$$\mathcal{E}cc_{sup} = \begin{pmatrix} \mathcal{E}cc_{sup}(v1) \\ \cdots \\ \mathcal{E}cc_{sup}(v_n) \end{pmatrix}$$

$$\mathcal{E}cc_{\Delta} = \mathcal{E}cc_{sup} - \mathcal{E}cc_{inf}$$

(d) On calcule la moyenne $\overline{\mathcal{E}cc_{\Delta}}$ et la variance $Var(\mathcal{E}cc_{\Delta})$ de $\mathcal{E}cc_{\Delta}$. Il est demandé de réfléchir à d'autre indicateurs et métriques utiles. Regarder l'évolution des degrée des noeuds

(e) Donner un critère d'arrêt en fonction des métriques et indicateurs calculés précédemment.
Par exemple :
Tant que $Var(\mathcal{E}cc_H) \geq \sigma$ continuer l'algorithme sinon stop

4. Retourner $max(\mathcal{E}cc_{sup})$ (cette valeur devra approximer le diamêtre de G)

5.6 Variantes

— Dans l'algorithme précédent à l'étape 3 on peut décider de choisir nos sommets S en fonction des résultats des valeurs des excentricités à l'étape 3c

— **Mais comme on s'intéresse à des graphes du monde réel, ceux ci ont en général une structure en communautés. Il est donc très intéressant de :**

1. **Faire un Louvain sur G pour calculer les communautés $C_1, \cdots, C_m$**
2. **Choisir un sommet s_i (ou plusieurs) dans chaque communauté C_i.**
3. **On peut envisager comme tactique(s) : (i) prendre au moins un sommet dans chaque communauté (faire évidemment le BFS) ; (ii) Choisir plusieurs sommets dans une communauté si elle est "grande" ; (iii) faire un mix des deux.**

5.7 Calcul des communautés d'un graphe : Louvain ou Leiden ?

Dans Igraph on trouve un algorithme de calcul de décomposition en communautés qui se nomme Leiden [6]. Cet algorithme, **très prometteur** essaye de remédier à quelques petits problèmes de Louvain. Il a l'air assez rapide, et quelquefois plus rapide et semble souvent donner des résultats plus pertinents que Louvain. **Il devrait à terme remplacer Louvain dans le coeur des spécialistes de la recherche de communautés.**

Il vous est donc tout à fait possible :

1. soit de remplacer Louvain par Leiden si vous voulez [7]
2. soit de les utiliser ensemble pour le projet 2 et de permettre ainsi de comparer leurs performances.

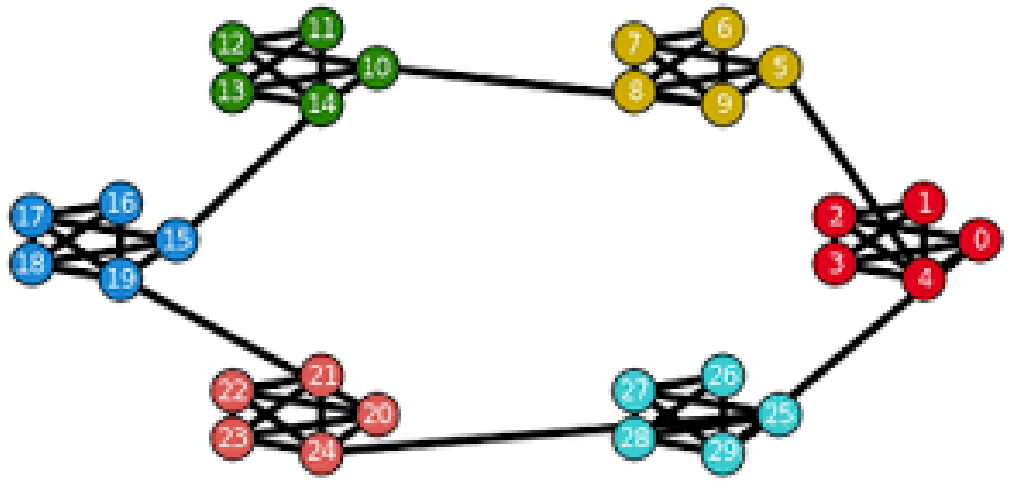
Remarque : le Louvain qu'on trouve habituellement exige un graphe non orienté. La raison est simple : le problème de calcul de communautés pour un graphe orienté est très différent du problème de calcul de communautés pour un graphe non orienté. Donc, ne testez que des graphes non orientés.

5.8 Calcul de communautés

Pour celles et ceux intéressés par le calcul de communautés, un article très fouillé, 103 pages ! https://arxiv.org/abs/0906.0612 Il date de 2009 mais reste d'actualité.

Un exemple plus "propre" de *caveman graph* que j'ai maladroitement essayé de faire avec paint :

6. https://arxiv.org/abs/1810.08473
7. En fait cela nous plairait bien qu'un groupe le fasse.



Voir aussi la figure 2 de l'article ""Fast unfolding of communities in large networks"".

6 Liens utiles

6.1 Articles

- https://arxiv.org/pdf/0904.2728.pdf : article de Clémence Magnien & al.
- https://liacs.leidenuniv.nl/~takesfw/pdf/diameter.pdf : "Determining the Diameter of Small World Networks" de Takes-Kosters
- https://www.mdpi.com/1999-4893/6/1/100 : "Computing the Eccentricity Distribution of Large Graphs" de Takes-Kosters
- https://arxiv.org/pdf/cs/0009005.pdf : "Fast Approximation of Centrality" de Eppstein et Wang
- https://kops.uni-konstanz.de/bitstream/handle/123456789/5739/algorithm.pdf : A Faster Algorithm for Betweenness Centrality de Brandes

6.2 Code et documentation

- https://www-complexnetworks.lip6.fr/~magnien/Diameter/ : Code et README de Clémence Magnien
- https://igraph.org/c/doc/igraph-Community.html : Documentation d'igraph en C
- https://github.com/franktakes/teexgraph : Github de Frank Takes contenant le code fourni pour *DiameterDB*

6.3 Data

- http://data.complexnetworks.fr/Diameter/ : Data de Clémence Magnien
- https://snap.stanford.edu/data/index.html : SNAP, site avec différents graphes à disposition.