Yannick Leo & Florent Robic
Algorithmique des réseaux
8 janvier 2012
http://perso.ens-lyon.fr/yannick.leo/

TRAITEMENT DES DONNÉES

Nous utilisons toujours, pour afficher le graphe, une version modifiée de la bibliothèque de Ivan Kuckir (http://g.ivank.net). Nous avons retiré toutes les parties concernant le placement, et modifié les parties qui nous intéressaient.

Les fichiers concernés sont :

- Control.js : S'occupe de tracer le graphe, de faire bouger les points, les zooms et lance le calcul des communautés.
- Grapher2D.js : C'est un graphe avec quelques primitives supplémentaires.
- slider.js : Récupération des événements de la souris.
- Vertex.js : Définition d'un sommet de graphe.
- Point.js : Définition de la structure d'un point de l'espace.

Les données sont acquises à l'aide de la fonction « batch ». Une fois celles-ci récupérées, la position des points est calculée grâce à un algorithme basé sur les interactions physiques (répulsions et attractions), les détails sont dans la suite du rapport. Une fois celui-ci affiché, les communautés sont calculées, l'algorithme utilisé est décrit plus loin dans le rapport.

Les fichiers concernés sont :

- app.js : Récupération des données, et calcul des positions initiales.
- worker.js : Calcul des communautés.

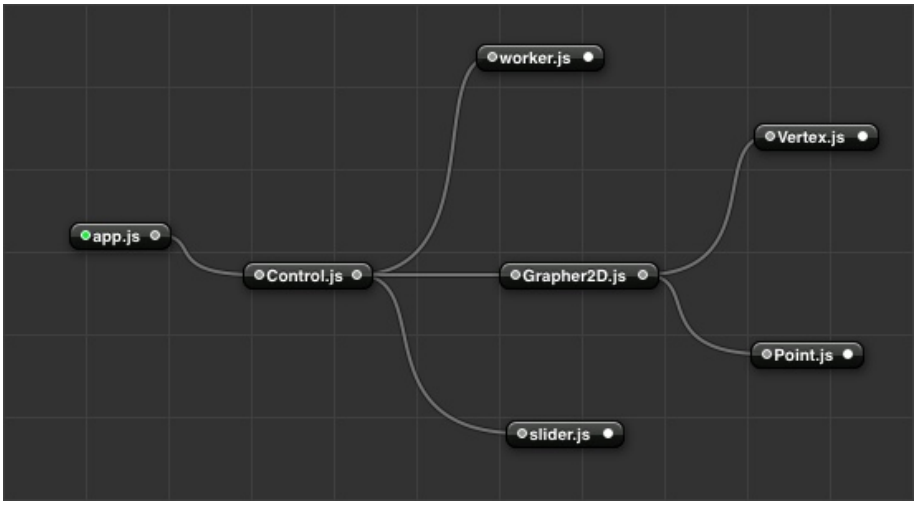

Fig.1 : Liens entre les fichiers

L'ALGORITHME DE PLACEMENT

Force-based Layout

Nous utilisons toujours le même algorithme de placement, chaque sommet est une particule soumise à 2 forces :

- Une force de Coulomb qui repousse les particules entre elles.
- Une force de rappel (ressort), qui attire les points voisins entre eux.

Cela permet de regrouper les communautés en tas. Nous avons rajouté un coefficient de frottements proportionnel au nombre de points considérés, pour éviter un trop fort écartement des communautés en cas de graphe très important.

Après ce calcul, des points sont souvent très rapprochés ainsi pour permettre une meilleure visibilité, nous effectuons quelques boucles supplémentaires en supprimant la force d'attraction.

Plutôt que d'attendre la convergence qui pourrait s'avérer trop longue nous limitons le calcul à 100 boucles pour le placement initial, puis entre 1 et 15 boucles en fonction du nombre total de points pour espacer le graphe.

Enfin, pour permettre l'affichage on réalise une homothétie qui permet d'avoir un graphe centré en 0 et aux proportions de notre fenêtre.

Complexité et temps de calcul

Pour les 100 boucles :

- Calcul des forces de Coulomb : $\mathcal{O}(n^2)$
- Calcul des forces d'attraction : $\mathcal{O}(m)$

On a donc une complexité qui reste raisonnable.

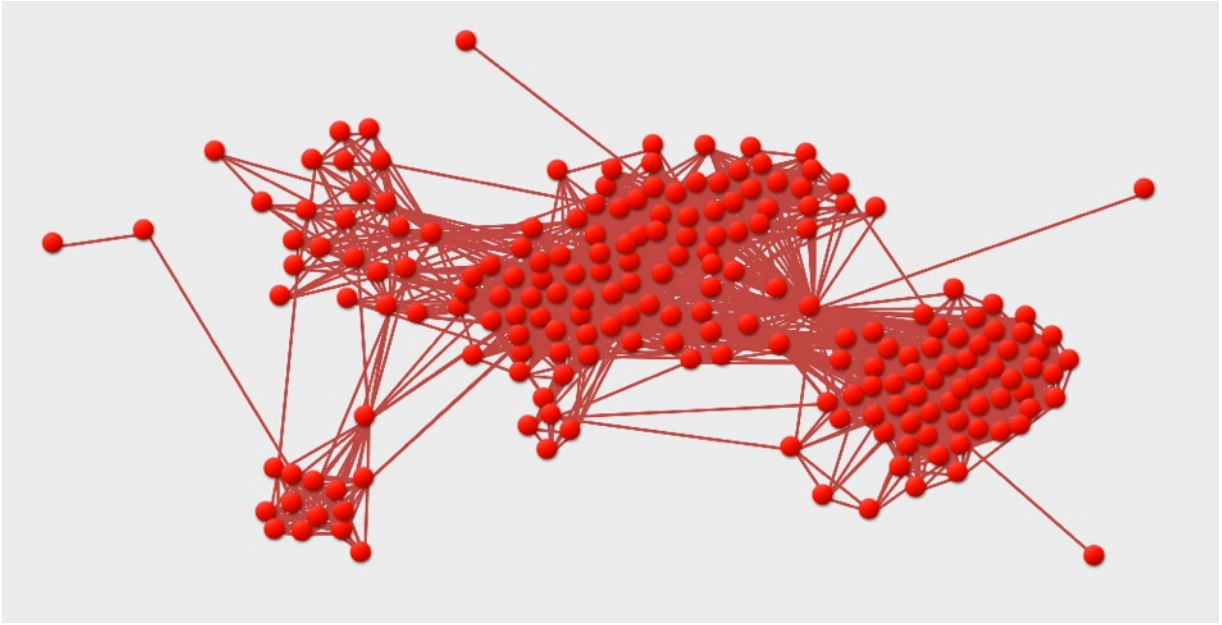

Fig.2 : Exemple d'un graphe à 200 nœuds.

L'ALGORITHME DE COMMUNAUTÉ

Fast unfolding of communities algorithm

Nous avons choisi d'implémenter l'algorithme de l'université de Louvain, sa rapidité lui permet de s'exécuter sans poser de problèmes malgré les limitations de puissances inhérentes au javascript (Cependant une nouvelle fois nous n'attendrons pas forcement l'équilibre pour arrêter le calcul enfin d'éviter une attente trop longue). L'algorithme se trouve dans le fichier worker.js (l'utilisation d'un worker permet de s'occuper d'afficher le graphe durant le calcul des communautés).

Pour résumer : l'algorithme consiste à minimiser une fonction énergie :

- Au départ tous les points ont leur propre communauté.

- Puis pour chacun d'entre eux on essaye de le déplacer dans une communauté voisine, si cela améliore l'énergie le changement est validé.

- Et ainsi de suite, lorsqu'il n'y a plus aucun changement, l'algorithme s'arrête.

Pour limiter la complexité il s'agit de calculer efficacement la différence d'énergie causée par le retrait / ajout d'un sommet à une communauté.

Ainsi l'on procède en deux étapes d'abord le retrait du sommet de sa communauté, ceci coûte un parcours des voisins, et un parcours de la communauté.

Puis pour chacun de ses voisins on essaye de le rajouter dans la communauté de celui-ci si celle-ci n'a jamais été essayée. À la fin nous ajoutons le point dans la communauté ayant maximisé son énergie (si toutes les possibilités sont néfastes par rapport à la configuration initiale il est réintégré).

Complexité et temps de calcul

Si le dans le pire cas nous pourrions avoir un algorithme en $\mathcal{O}(n^3)$ opérations, il s'exécute en fait en $\mathcal{O}(n^2)$ opérations, permettant ainsi de finir le calcul rapidement.

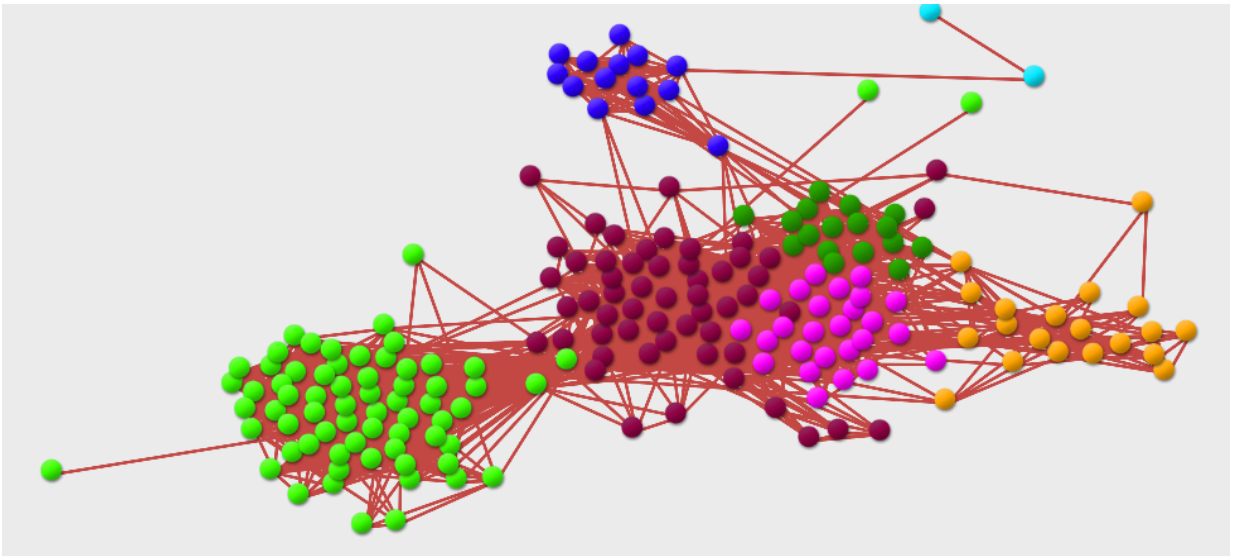

Fig3. : Affichage des Communautés

FONCTIONS SUPPLÉMENTAIRES

- Zoom sur les communautés
- Application Facebook : https://apps.facebook.com/126547340780835/?ref=ts
- Recherche d'un ami à l'aide d'un champ avec auto-complétion
- Fiche de personne :
  - Nom, Prénom, Photographie
  - Nombre d'amis communs
  - Coefficient de Clustering
  - Coefficient de Clustering dans sa Communauté.

    Note : L'algorithme se trouve dans Grapher2D.js : BuildComm() ;

    Il s'exécute dans le pire cas en $\mathcal{O}(n^3)$ mais le plus souvent en $\mathcal{O}(n^2)$.

Fait avec amour...