HAL open science

Recherche et représentation de communautés dans un grand graphe. Une approche combinée

Nathalie Villa-Vialaneix, Taoufiq Dkaki, Sébastien Gadat, Jean-Michel Inglebert, Quoc-Dinh Truong

To cite this version:

Nathalie Villa-Vialaneix, Taoufiq Dkaki, Sébastien Gadat, Jean-Michel Inglebert, Quoc-Dinh Truong. Recherche et représentation de communautés dans un grand graphe. Une approche combinée. Document numérique - Revue des sciences et technologies de l'information. Série Document numérique, 2011, 14 (1), pp.59-80. 10.3166/dn.14.1.59-80 . hal-00590612

HAL Id: hal-00590612

https://hal.science/hal-00590612v1

Submitted on 4 May 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Recherche et représentation de communautés dans un grand graphe : une approche combinée

Nathalie Villa-Vialaneix*, — Taoufiq Dkaki*** — Sébastien Gadat* — Jean-Michel Inglebert*** — Quoc-Dinh Truong******

** Institut de Mathématiques de Toulouse, UMR 5219*
Université de Toulouse
118 route de Narbonne
F - 31062 Toulouse cedex 9
France
{nathalie.villa,sebastien.gadat}@math.univ-toulouse.fr
*** Université de Perpignan Via Domitia, IUT, Département STID*
Domaine Universitaire d'Auriac
Avenue du Dr Suzanne Noël
F - 11000 Carcassonne
France
**** Institut de Recherche en Informatique de Toulouse, UMR 5505*
Université de Toulouse
118 route de Narbonne
F - 31062 Toulouse cedex 9
France
dkaki@irit.fr, inglebert@iut-blagnac.fr
***** Université de Cantho*
1 rue Ly Tu Trong, Cantho
Vietnam
truong@univ-tlse2.fr

RÉSUMÉ. *Le travail présenté ici concerne l'analyse, la compréhension et la représentation de grands graphes. Ce type de données se retrouve de manière naturelle dans un nombre croissant de problèmes concrets : web, recherche d'information, réseaux sociaux, réseaux d'interaction biologiques... La progression des moyens de recueil et de stockage des données rend la taille de ces graphes croissante : le développement de méthodes permettant leur analyse et leur représentation est donc un domaine de recherche dynamique et important à l'interface de nombreuses disciplines (mathématiques, statistique, informatique, ...). Dans cet article, nous développons une méthode de représentation de graphes basée sur une classification préalable des sommets destinée à simplifier la structure macroscopique du graphe avant sa représentation complète. La phase de classification consiste en*

l'optimisation, par un algorithme stochastique de recuit simulé, d'une mesure de qualité spécialement adaptée à la recherche de groupes denses dans les graphes : la modularité. Enfin, la représentation finale est basée sur un algorithme de « forces » contraint. Deux exemples issus de l'analyse de réseaux sociaux sont ensuite présentés en partie 4.

ABSTRACT. *This paper deals with the analysis and the visualization of large graphs. Graphs are convenient widespread data structures that are encountered in a growing number of concrete problems: web, information retrieval, social networks, biological interaction networks… The sizes of these graphs become increasingly large as data acquisition and storage are developing. This calls for new methods in graph analysis and visualization which are now important and dynamic research fields at the interface of many disciplines such as mathematics, statistics, computer science and sociology. In this paper, we propose a method for graphs representation and visualization based on a prior clustering of the vertices that aims at providing a simplification of the graph at a macro-scale. The clustering step consists in optimizing a quality measure specifically suited for the search of dense groups in graphs, the* modularity*, by a simulated annealing algorithm. The visualization/representation step is based on a constrained force-directed algorithm. Finally, the approach is illustrated by two applications coming from the social network field.*

MOTS-CLÉS *:Graphe, Représentation de graphe, Classification, Réseaux sociaux, Communautés, Fouille de données*

KEYWORDS: Graphs, Graph drawing, Clustering, Social networks, Communities, Data mining

1. Introduction

La classification de sommets d'un grand graphe a déjà fait l'objet de nombreux travaux. Cette approche a été présentée depuis longtemps comme un moyen de simplifier la structure d'un grand graphe, dont le nombre de sommets peut être égal à plusieurs milliers, pour en faire ressortir sa structure macroscopique :

« *reducing [the] level of complexity [of a network] to one that can be interpreted readily by the human eye, will be invaluable in helping us to understand the large-scale structure of these new network data* »

(voir (Newman *et al.*, 2004)). L'idée de la classification des sommets est de mettre en valeur des « communautés » (pour reprendre la terminologie couramment employée dans le domaine des réseaux sociaux). Même si aucune définition formelle de ce qu'est une communauté n'est aujourd'hui consensuelle, on désigne généralement sous ce terme un sous-ensemble de sommets du graphe, fortement connectés entre eux et faiblement connectés aux autres sommets. Aucune structure euclidienne, a priori, n'est définie entre les sommets d'un graphe et la classification des sommets requiert donc l'implémentation de méthodes spécifiques par rapport aux données multidimensionnelles usuelles. Une très grande variété de méthodes s'attachent à trouver des solutions satisfaisantes au problème de la partition des sommets d'un graphe : (Schaeffer, 2007) fait une revue de ces méthodes qui sont basées sur la définition de similarités entre sommets ou sur l'optimisation de critères de qualité. Parmi celles-ci, la classification spectrale (voir (von Luxburg, 2007)) est une approche devenue très populaire mais qui est très sensible à la taille des données puisqu'elle nécessite la décomposition spectrale d'une matrice carrée dont la taille est égale au nombre de sommets du graphe.

Nous proposons, dans cet article, d'utiliser la classification des sommets d'un graphe en étape préalable à la représentation de celui-ci. La pré-étape de classification permet de simplifier la structure macroscopique du graphe et de décomposer le problème de la représentation en sous-problème plus simple. A l'inverse, l'étape de représentation, permet de mieux visualiser les relations inter et intra communautés. Ainsi, la combinaison des deux approches se veut une méthode efficace d'aide à la compréhension du graphe que l'on peut proposer à un utilisateur non expert. Une approche similaire est décrite dans (Karouach *et al.*, 2003) où les auteurs utilisent les résultats d'une classification hiérarchique de la matrice d'adjacence du graphe comme méthode d'initialisation d'un algorithme attraction répulsion. Nous présentons ici une approche encore plus dirigée puisque la classification est utilisée comme étape préalable à une représentation contrainte des sommets du graphe.

Dans cet article, la classification des sommets est effectuée par une approche optimisant une mesure de qualité populaire de la classification des sommets dans un graphe : la modularité, introduite par (Newman *et al.*, 2004). L'originalité de notre approche réside dans l'utilisation d'un algorithme stochastique d'optimisation de ce

critère, algorithme permettant sa mise en œuvre pour des graphes dont la taille peut être importante. Enfin, nous montrons comment cette méthode de classification peut être combinée avec un algorithme de représentation adapté au cas de graphes dont les sommets ont été partitionnés en classes et que l'on appelera, dans la suite *graphes en classes*. Cette approche aboutit à une représentation lisible de l'ensemble du graphe, des principaux groupes mis en valeur par la phase de classification et des liens existant entre sommets et entre groupes.

Le reste de l'article s'organise comme suit : la Section 2 présente l'étape de classification de sommets et évalue l'algorithme proposé sur un jeu de données simulé. La Section 3 présente l'algorithme de représentation de graphes en classes et, finalement, la Section 4 illustre l'approche sur deux jeux de données issus du domaine des réseaux sociaux.

2. Classer les sommets d'un graphe en optimisant la modularité par recuit simulé

2.1. *Classification de sommets et modularité*

(Schaeffer, 2007) présente les principales méthodes de classification de sommets dans un graphe : certains auteurs se sont concentrés sur la construction d'une mesure de similarité ou d'une distance entre sommets du graphe pour pouvoir ensuite appliquer directement les méthodes de classification existantes pour des données définies par des tableaux de similarités (voir par exemple, (Capoccia *et al.*, 2005), (Lakroum *et al.*, 2005) ou (Dong *et al.*, 2006), pour des exemples de telles approches). Une méthode alternative consiste à optimiser une mesure de qualité de la classification, mesure définie à partir de la structure du graphe, c'est-à-dire, de la donnée de ses arêtes. Classiquement, on peut minimiser le nombre ou le poids des arêtes inter-classes : si G est un graphe non orienté, pondéré, de taille n, de sommets $V = \{1, 2, ..., n\}$ et dont l'ensemble des arêtes, E, est décrit par la matrice (symétrique, à diagonale nulle et coefficients positifs) des poids W, on appelle *coupe* d'une classification $c(i)$ (pour $i \in V$) la quantité :

$$\frac{1}{2} \Sigma_{i,j \in V, c(i) \neq c(j)} W_{ij} \ .$$

Cette mesure de qualité est fortement reliée à la définition d'une certaine représentation du graphe, appelé plongement, dans un espace euclidien (voir par exemple, (von Luxburg, 2007)). Son optimisation, qui est un problème NP-complet, est fréquemment approchée par des méthodes de classification de type k-means sur ce plongement. Cette méthode est connue sous le nom de classification spectrale (« spectral clustering »).

Cependant, si la classification spectrale est séduisante par sa simplicité et son interprétation très naturelle ; elle peut s'avérer décevante en pratique. La raison est exposée dans (Newman, 2006) : la mesure de coupe considère toutes les arêtes de manière similaire. Or, Newman fait remarquer qu'être lié à un sommet de fort degré est moins exceptionnel qu'être lié à un sommet de faible degré. Pour tenir compte de cet aspect, il introduit une mesure de qualité basée sur le calcul d'une distance à un modèle nul dans lequel la répartition des arêtes entre et à l'intérieur des classes ne dépend que du degré des sommets. Pour une classification donnée C_1, ... C_p des sommets du graphe, la modularité est la quantité :

$$Q = \frac{1}{2m} \Sigma_{k=1\ldots p} \Sigma_{i,j \in C_k, i \neq j} \left(W_{ij} - P_{ij}\right)$$

où m est le nombre total d'arêtes du graphe et les P_{ij} sont les poids estimés des arêtes intra classes entre les sommets i et j dans le modèle nul où la répartition des arêtes dépend uniquement du degré et non des classes. Ainsi, une valeur de Q grande (et positive) est le signe d'une sur-représentation (par rapport au modèle nul) des arêtes à l'intérieur des classes alors qu'une valeur de Q fortement négative est le signe d'une sur-représentation des arêtes à l'extérieur des classes. La notion naturelle de communautés, exposée plus haut, correspond au premier cas.

Le choix de la valeur des P_{ij} est fait en fonction des hypothèses correspondant au modèle nul. Des contraintes raisonnables sur ces valeurs sont, usuellement :

1. $\Sigma_{i,j \in V} P_{ij} = 2m$ (i.e., le nombre d'arêtes du modèle nul est identique au nombre d'arêtes du graphe considéré) ;
2. $\Sigma_{j \in V} P_{ij} = d_i$ où, pour tout $i \in V$, d_i désigne le degré du sommet i : $d_i = \Sigma_{j \in V} W_{ij}$. Cette hypothèse est simplement équivalente au fait que le degré de chaque sommet du modèle nul est le même que le degré des sommets dans le graphe considéré.

Enfin, par hypothèse, on suppose que P_{ij} est de la forme $f(d_i)\ f(d_j)$ qui traduit, d'une part, le fait que les poids attendus du modèle nul ne dépendent que des degrés des sommets considérés et non de leur répartition entre les différentes classes et, d'autre part, le fait que les poids entre deux arêtes données sont indépendants. (Newman, 2006) montre qu'alors la valeur des P_{ij} est donnée par $P_{ij} = \frac{d_i d_j}{2m}$.

2.2. *Optimisation des sommets par recuit simulé*

L'optimisation de Q sur l'ensemble des partitions des sommets du graphe est un problème NP-complet qui ne peut être effectuée par une recherche exhaustive sur

l'ensemble des partitions possibles des sommets en p classes pour la plupart des graphes réels. Diverses approches d'approximation de ce critère ont été proposées récemment : (Newman *et al.*, 2004) propose une optimisation hiérarchique qui peut, dans certains cas, conduire à des solutions assez inadéquates (voir, entre autres, l'exemple donné dans (Newman, 2006)). (Newman, 2006) propose une approximation par optimisation d'un critère basé sur une décomposition spectrale d'une matrice de taille n^2. (Lehman *et al.*, 2007) et (Rossi *et al.*, 2010) proposent, respectivement, l'optimisation de ce critère ou d'un critère adapté à une classification organisée, par un algorithme de recuit déterministe. Ce type d'algorithme est rapide et efficace mais nécessite un réglage adéquat des paramètres de refroidissement du système qui nécessite de connaître une partie du spectre d'une matrice de taille n^2 liée au graphe.

Nous présentons ici une approche par recuit simulé de l'optimisation de la modularité : cette approche, basée sur une démarche stochastique, présente l'avantage de ne nécessiter que des opérations de très faible complexité et peu de réglage de paramètres pour sa mise en œuvre : elle est donc particulièrement bien adaptée à des données de grande taille. Le recuit simulé est un algorithme stochastique, présenté dans (Kirpatrick *et al.*, 1983) et destiné à optimiser une fonction de coût lorsque d'autres algorithmes plus simples (de type descentes de gradient) sont inadaptés. Il est particulièrement utile lorsque l'espace des possibles est discret ou bien lorsque la fonction possède de nombreux maxima locaux. Ici, l'optimisation de la modularité sur l'espace des partitions possibles du graphe est typiquement le cadre d'application d'un algorithme stochastique tel que le recuit simulé. Cette approche a d'ailleurs été utilisée dans (Reichardt *et al.*, 2006) pour des simulations sur la modularité. L'algorithme de recuit simulé et son application à l'optimisation de la modularité sont présentés ici brièvement. Pour plus de détails, le lecteur pourra se référer à (Kirpatrick *et al.*, 1983).

De manière similaire à l'algorithme de Métropolis-Hastings (Hastings, 1970), le principe de l'algorithme de recuit simulé est de simuler, par une chaîne de Markov, une variable prenant ses valeurs dans l'ensemble des classifications possibles des sommets du graphe. Cette chaîne de Markov a pour loi de probabilité invariante :

$$\mu_T(C) = \frac{e^{Q(C,G)/T}}{Z}$$

où $Q(C,G)$ désigne la modularité de la classification C du graphe G, T est un paramètre de température positif et Z une constante de normalisation qu'il est inutile de calculer. L'intérêt de cette loi apparaît lorsque l'on fait tendre le paramètre de température, T, vers 0 : la chaîne de Markov a alors tendance à se figer sur les états de grande probabilité, c'est-à-dire, sur les maxima de Q. Un choix généralement admis pour la décroissance du paramètre de température est $T_l = \frac{\gamma}{\log l}$ où l désigne

l'état l de la chaîne de Markov (i.e., le nombre d'itérations de l'algorithme) et γ est une constante strictement positive.

La mise en œuvre pratique de la simulation de la chaîne de Markov est faite par le biais de la méthode d'« acceptation / rejet » : étant donnée une classification C_I, des sommets du graphe G, on choisit, de manière aléatoire, une classification « voisine », C_F, selon un processus réversible. La classification C_F est choisie à partir de C_I avec une probabilité $P(C_I,C_F)$ (dans notre cas, on prend, pour simplifier, $P(C_I,C_F) = P(C_F,C_I)$). Enfin, on accepte la classification C_F (c'est à dire que l'on conserve cette classification comme nouvel état de la chaîne de Markov) avec la probabilité $\min\left(1, e^{\frac{\Delta Q(C_I, C_F, G)}{T}}\right)$ où $\Delta Q(C_I, C_F, Q)$ désigne la différence de modularité entre la classification C_I et la classification C_F (plus précisément, $\Delta Q(C_I, C_F, Q) = Q(C_F, G) - Q(C_I, G)$). L'algorithme d'optimisation de la modularité par recuit simulé est décrit dans l'Algorithme 1.

1. **Choisir, de manière aléatoire, une partition initiale de *V* en *p* classes** C_1^0 **, ...,** C_p^0 **.**
2. **Calculer la modularité associée à cette partition,** $Q^0\left(C_1^0,\ldots,C_p^0,G\right)$.
3. **Pour l = 1 ..., L, répéter :**
   a. **Tirer deux classes,** C_i^{l-1} **et** C_j^{l-1} **, au hasard, parmi** $\left(C_i^{l-1}\right)_{i=1\ldots p}$
   b. **Si** $C_i^{l-1} \cup C_j^{l-1} \neq \varnothing$ **, tirer un sommet, *x*, au hasard dans** $C_i^{l-1} \cup C_j^{l-1}$ **; sans perte de généralité, on peut supposer, dans la suite, que *x* ∈** C_i^{l-1} **.**
   c. **Définir une nouvelle classification telle que** $C_k = C_k^{l-1}$ **pour tout** $k \neq i, j$ **,** $C_i = C_i^{l-1} - \{x\}$ **et** $C_j = C_j^{l-1} \cup \{x\}$ **(changement de classe de *x*).**
   d. **Calculer** $\Delta Q = Q\left(C_1,\ldots,C_p,G\right) - Q^{l-1}\left(C_1^{l-1},\ldots,C_p^{l-1},G\right)$.
   e. **Selon que :**
      - $\Delta Q > 0$ **, on accepte la nouvelle classification :** $C_k = C_k^{l-1}$ **pour tout *k* et on a donc** $Q^l\left(C_1^l,\ldots,C_p^l,G\right) = \Delta Q + Q^{l-1}\left(C_1^{l-1},\ldots,C_p^{l-1},G\right)$ **.**
      - $\Delta Q < 0$ **, on accepte la nouvelle classification avec probabilité** $e^{\gamma \log(l) \Delta Q}$ **(pour un γ > 0 fixé) ; on en déduit** $Q^l\left(C_1^l,\ldots,C_p^l,G\right)$ **.**

Algorithme 1 : Algorithme de recuit simulé pour l'optimisation de la modularité

L'étape 3.d de l'Algorithme 1 ne nécessite pas le calcul intégral de la modularité $Q^{l-1}\left(C_1^{l-1},\ldots,C_p^{l-1},G\right)$. En effet, (Lehman *et al.*, 2007) montrent que la modularité peut être mise sous la forme matricielle

$$Q\left(C_1,\ldots,C_p,G\right) = \frac{1}{2m}\mathrm{Tr}\left(S^T BS\right)$$

où

- S désigne la matrice de taille $p \times n$ dont les éléments sont, pour tout sommet x dans V et toute classe C_k, $k = 1, \ldots, p$, $S_{kx} = \begin{cases} 1 & \text{si } x \in C_k \\ 0 & \text{sinon} \end{cases}$ (en d'autres termes, S est la matrice d'appartenance des sommets aux classes) ;

- B désigne la matrice $(W - P)$;

- $\mathrm{Tr}(S^T BS)$ est la trace de la matrice $S^T BS$, c'est-à-dire, la somme de ses éléments diagonaux.

On montre alors facilement que l'étape 3.d se réduit à :

$$\Delta Q = \frac{1}{m} \sum_{y \neq x} \left(S_{jy}^{l-1} B_{xy} - S_{iy}^{l-1} B_{xy} \right) = \frac{1}{m} \left(\sum_{y \in C_j^{l-1}} B_{xy} - \sum_{y \in C_i^{l-1} - \{x\}} B_{xy} \right)$$

ce qui nécessite moins de $(n - 1)$ additions, c'est-à-dire, bien moins que le calcul de la modularité d'une classification donnée qui nécessite de l'ordre de n^2 opérations élémentaires.

2.3. *Un exemple simulé*

Nous nous proposons d'analyser la qualité de la classification obtenue par l'approche décrite dans l'Algorithme 1. Pour ce faire, nous utilisons un modèle de graphe aléatoire (non orienté et non pondéré) plausible du point de vue de l'étude des réseaux sociaux : il s'agit du « planted p-partition model » décrit, par exemple, dans (Condon *et al.*, 2001). Ce modèle est composée de groupes de sommets denses et faiblement reliés entre eux et présente donc une structure naturelle en communautés que l'algorithme de classification doit être capable de pouvoir retrouver. De manière plus précise, un graphe aléatoire généré par ce modèle dépend de 4 paramètres, notés p (nombre de classes, appelées dans la suite *classes naturelles*), k (nombre de sommets dans chaque classe), r (probabilité d'une arête intra-classes) et q (probabilité d'une arête inter-classes). Nous noterons, dans la suite, $G(p,k,r,q)$ ce modèle. Une réalisation du modèle pour 3 classes de 4 sommets est donnée dans la Figure 1.

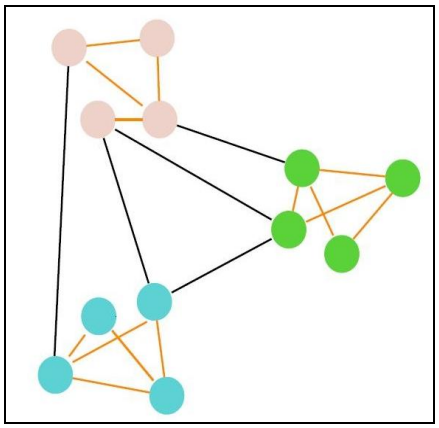

Figure 1 : Exemple de « planted *p*-partition graph » : le graphe est composé de 3 classes de 4 sommets avec une proportion d'arêtes intra-classes de l'ordre de 58% et une proportion d'arêtes inter-classes de l'ordre de 10 %.

Pour évaluer si l'Algorithme 1 était capable de retrouver les classes naturelles du modèle « planted *p*-partition », nous avons généré pour des valeurs fixées des paramètres du modèle, 50 graphes aléatoires. Deux ensembles de paramètres ont été testés :

- $G(5,100,0.7,0.02)$: sur les 50 graphes aléatoires générés selon ce modèle, la modularité moyenne de la partition naturelle est de l'ordre de 0,70. Cette valeur est compatible avec les valeurs trouvées dans les réseaux sociaux réels ;

- $G(50,10,0.7,0.2)$: sur les 50 graphes aléatoires générés selon ce modèle, la modularité moyenne de la partition naturelle est de l'ordre de 0,37. Cette valeur est bien inférieure aux valeurs généralement observées sur des réseaux sociaux réels. La recherche des classes naturelles dans ce modèle peut donc être considérée comme un problème relativement compliqué.

Par ailleurs, l'algorithme a été testé en initialisant le nombre de classes selon deux stratégie : une stratégie dans laquelle le nombre de classes est supposé connu (on initialise donc le nombre de classes de l'algorithme à la valeur p du nombre de groupes de sommets dans les graphes aléatoires que l'on a générés) et une stratégie dans laquelle le nombre de classes est inconnu (et on initialise donc le nombre de classes de l'alorithme à une valeur $p_{algo} > p$). Ceci permet d'évaluer si l'algorithme est sensible à l'initialisation du nombre de classes. Enfin, la qualité de reconstruction des classes a été mesurée de deux manières différentes :

- par le calcul du taux de mauvais classement des sommets dans leur classe naturelle (pourcentage de sommets qui ne sont pas classés dans la classe la plus

fréquemment affectée aux sommets de la classe naturelle à laquelle ils appartiennent) ;

- par calcul de la différence entre la modularité de la classification naturelle et la modularité de la classification fournie par l'algorithme (cette mesure de qualité sera appelée *défaut de modularité*).

Les résultats obtenus sont résumés dans le Tableau 1. Le taux de bonne classification des sommets dans leurs classes naturelles est tout à fait satisfaisant, avec moins de 10% de sommets mal classés même dans la situation la plus difficile (celle décrite par la dernière colonne du tableau). De même, la modularité est correctement optimisée puisque le défaut de modularité représente environ 10% de la modularité attendue pour le problème le plus difficile (dernière colonne) contre 5% environ pour le problème le plus facile (première colonne). Enfin, même dans la situation où le nombre de classes recherchées par l'algorithme est très supérieur au nombre de classes naturelles (deuxième colonne), l'algorithme se comporte correctement et tend à s'adapter en donnant un nombre de classes non vides automatiquement plus faible que le nombre de classes proposées par défaut par l'utilisateur.

Tableau 1 : Performances moyennes de l'algorithme d'optimisation de la modularité par recuit simulé sur le modèle « p-planted partition graph ».

$r \times q$ $p \times k$ p_{algo}	0.7×0.02 5×100 5	0.7×0.02 5×100 10	0.7×0.2 50×10 50
Taux d'erreur (classification)	1,36 %	2,34 %	6,32 %
Défaut de modularité	0,036	0,040	0,036

Les résultats de cette section sont le signe que l'algorithme proposé est une approche de classification des sommets d'un graphe dont les performances sont tout à fait satisfaisantes. Dans la suite, nous expliquons comment utiliser une classification pertinente obtenue par cette approche pour représenter l'intégralité du graphe de manière à en faire ressortir les principales structures.

3. Combinaison classification et représentation pour une visualisation complète du graphe et des communautés

3.1. *Représentation des sommets d'un graphe en classes*

Il est admis que la visualisation contribue fortement à l'efficacité des processus d'analyse et d'exploitation des données. C'est dans cette optique que plusieurs travaux se sont intéressés au dessin de graphes. Construire un dessin de graphe consiste à positionner les sommets et les arcs d'un graphe dans un espace d'affichage selon des critères et des contraintes dont l'objectif est d'offrir les visualisations les plus facilement interprétables. Les travaux les plus connus se fondent sur une approche dirigée par des forces. Dans cette approche, que l'on peut attribuer à Eades (voir, par exemple, (Eades, 1984)), les sommets sont assimilés à des anneaux et les arcs à des ressorts. Les sommets sont placés dans une disposition initiale souvent aléatoire et laissés libres de sorte que les forces des ressorts les déplacent vers un état d'équilibre où l'énergie du système – anneaux et ressorts – est minimale. Une autre variante de cette méthode, appelée « force et repousse », consiste à modifier la manière dont les forces sont appliquées : des forces répulsives sont activées pour toutes les paires de sommets tandis que des forces attractives sont instaurées uniquement pour les sommets connectés.

Ces algorithmes de forces sont utilisés avec succès pour le dessin de graphes possédant relativement peu de sommets. La difficulté de visualiser un graphe dépend, en effet, de sa taille, qui est l'obstacle majeur pour la réalisation de l'objectif de visualisation. Plus le nombre de sommets et d'arcs est grand, plus les critères de visibilité et de qualité de l'interprétation sont difficiles à atteindre. Pour prendre en charge des graphes contenant plus que quelques centaines de sommets, on a recourt soit à des algorithmes plus efficaces (voir, par exemple, (Hachul *et al.*, 2005)), soit à une classification préalable ayant pour but de réduire la complexité du graphe. Cette réduction de complexité consiste souvent à considérer le graphe des classes plutôt que le graphe initial (voir, (Rossi *et al.*, 2010) pour un exemple d'utilisation de cette approche) et s'accompagne donc indéniablement d'une perte d'information. C'est cette perte d'information que nous cherchons à minimiser par l'utilisation d'algorithmes de représentation de graphes en classes. Ce type d'algorithme fournit une visualisation qui permet d'apprécier non seulement la place de chaque sommet dans le graphe (relativement à ses voisins) mais aussi la structure en communautés du graphe. Elle permet ainsi de montrer les relations entre communautés (ou classes) et de juger l'influence de l'appartenance à une classe sur le positionnement de chaque sommet. Elle bénéficie de la simplification de structure apportée par l'étape de classification tout en conservant la représentation du graphe dans son ensemble.

En réalité, très peu de travaux ont été dédiés à la visualisation des graphes en classes. Parmi ces travaux, nous pouvons mentionner ceux de (Noack, 2004), de

(Chuang *et al.*, 2004) et de (Eades *et al.*, 2000) comme les plus représentatifs. Ces travaux se basent sur le modèle de forces traditionnel de (Kamada *et al.*, 1989), ajoutent un sommet virtuel représentatif de chaque classe et relient chacun de ces nouveaux sommets à tous les sommets de la classe à laquelle ils sont associés. Ces modèles donnent de bons résultats pour des graphes dont les classes sont beaucoup plus denses que le graphe global. Ainsi, outre l'introduction d'un sommet virtuel dont le sens n'est pas réellement défini de manière concrète, cette approche s'avère inutilisable lorsque le rapport de la densité dans les classes à la densité globale du graphe n'est pas faible. Cela peut être le cas dans de nombreux exemples concrets, notamment lorsque le nombre de classes de la classification est faible devant la taille du graphe.

Dans cet article, nous proposons d'utiliser une approche alternative de visualisation capable de gérer la visualisation des graphes en classes quelle que soit le rapport des densités intra et inter classes. Cette approche a été présentée dans (Truong *et al.*, 2007). Elle est basée sur le modèle de forces de (Fruchterman *et al.*, 1991) et permet d'associer à chaque classe une zone de visualisation prédéfinie dans laquelle chacun de ses sommets sera représenté. L'algorithme proposé présente deux différences par rapport à celui de Fruchterman et Reingold : tout d'abord, les forces mises en œuvre sont de deux types : celles intervenant entre les sommets d'une même classe d'une part et celles intervenant entre les nœuds de classes différentes d'autre part. Ces forces sont caractérisées par des intensités différentes dans les deux cas. Ensuite, les sommets sont contraints à l'intérieur des zones associées à leur classe d'appartenance par l'ajout de forces répulsives entre les sommets et la frontière de leur zone. Pour ce faire, les sommets et les frontières des zones associées aux différentes classes sont chargés de forces électriques de même nature. Ceci assure que chaque sommet reste toujours à l'intérieur de la zone qui lui a été initialement associée.

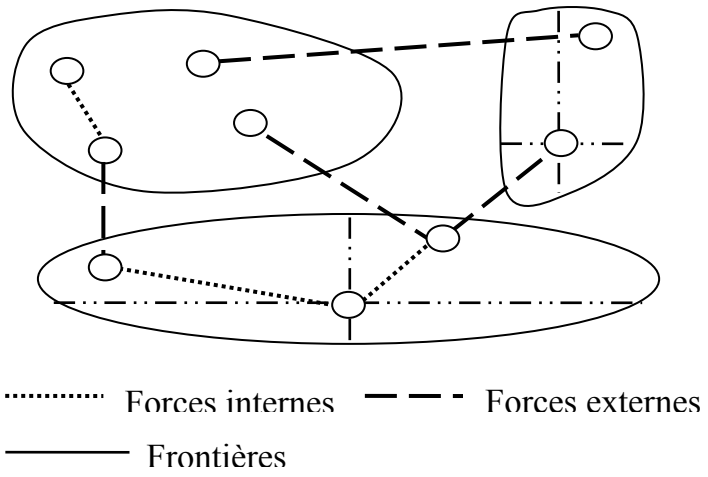

Figure 2 : Modèle de forces pour la représentation d'un graphe « clusterisé ».

La Figure 2 résume l'intégralité du modèle de forces mis en œuvre lors de la représentation d'un graphe en classes. C'est cette approche que nous combinons avec la méthode de classification de sommets décrite dans la Section 2 pour obtenir une représentation globale d'un réseau social découpé en communautés.

3.2. *Combinaison avec la phrase de classification*

Nous proposons d'utiliser la classification des sommets du graphe comme étape préalable à la représentation globale de celle-ci, tout en préservant la structure en communautés trouvées. De manière plus précise, après avoir effectué la classification des sommets du graphe en optimisant la modularité par recuit simulé, le graphe des classes est d'abord représenté grâce à un algorithme de forces ordinaire. Cette étape est similaire à celle décrite dans (Rossi *et al.*, 2010) : le graphe des classes est le graphe dont les sommets sont les classes et il existe, dans ce graphe, une arête entre deux classes C_k et C_l si et seulement si au moins une paire de sommets du graphe initial, (i,j), est telle que $i \in C_k$, j $\in C_l$ et $(i,j) \in E$ (la paire de sommets (i,j) est connectée dans le graphe initial). Les arêtes entre classes sont pondérées par la somme des poids des arêtes reliant des paires de sommets dans le graphe initial : pour tout k, l dans $\{1, \ldots, p\}$, le poids de l'arête reliant les classes C_k et C_l est :

$$W_{kl}^{c} = \sum_{i \in C_k,\, j \in C_l} W_{ij} \ .$$

Lorsque l'on représente ce graphe chaque classe peut être figurée par un symbole (un cercle, un carré) dont la surface est proportionnelle à l'effectif de la classe. La représentation est très simplifiée mais facile à lire si le nombre de classes est peu élevé.

Ensuite, l'algorithme de représentation d'unen classes, décrit dans la Section 3.1., est utilisé pour représenter l'intégralité des sommets du graphe initial : les positions des classes déterminées par l'algorithme de forces appliqué au graphe des classes sont utilisées comme zone de contrainte dans laquelle les sommets appartenant à la classe considérée doivent se situer.

4. Applications

4.1 *Représentation du réseau social issu du roman « Les Misérables »*

Dans ce premier exemple, nous traitons un réseau élémentaire très connu pour montrer comment la combinaison d'outils de classification et de visualisation permet de donner une représentation simple des structures principales d'un graphe. Il s'agit d'un réseau social issu du livre « Les Misérables » de Victor Hugo et décrit dans (Knuth, 1993) qui dénombre, les apparitions simultanées des 77 personnages du roman. Ces co-apparitions peuvent être modélisées par un graphe pondéré et symétrique, représenté dans la Figure 3 ci-dessous[1].

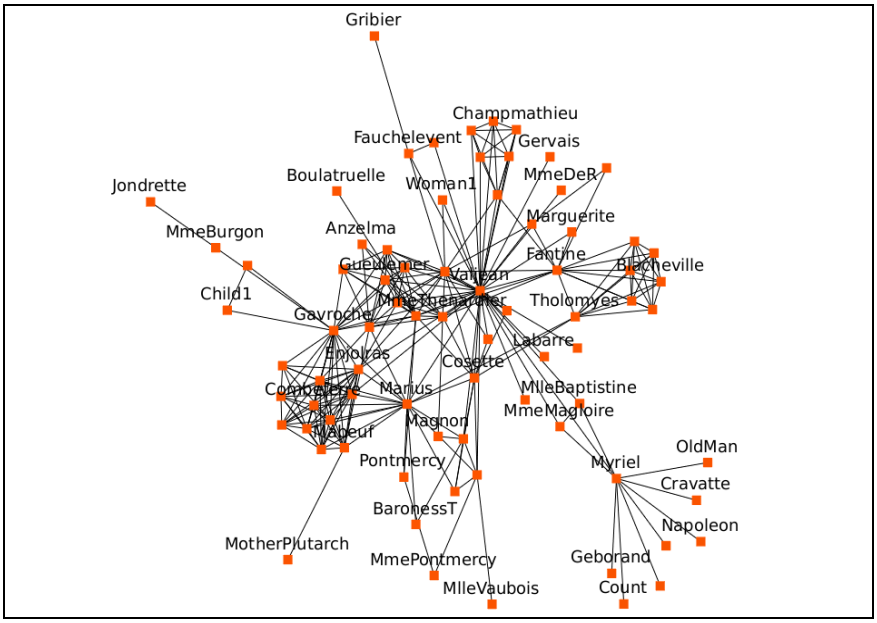

Figure 3 : Représentation du graphe initial par un algorithme de forces.

La première phase de l'analyse consiste à mettre en œuvre l'algorithme de recuit simulé pour la classification des sommets. Celui-ci a été testé avec :

- différentes valeurs du nombre de classes (variant de 5 à 10). En effet, la modularité n'est pas une fonction monotone du nombre de classes et peut donc permettre un choix optimal (au sens de ce critère) du nombre de classes ;

[1] Cette représentation a été réalisée à l'aide du logiciel Tulip, voir (Auber, 2001).

- différentes valeurs du paramètre γ de l'Algorithme 1 .

Par ailleurs, plusieurs itérations de l'algorithme ont été effectuées : l'algorithme étant basé sur un processus stochastique, chaque itération produit une solution potentiellement différente, les répétitions ayant pour but d'approcher au mieux l'optimum en temps fini. L'évolution de la valeur optimale de la modularité optimale trouvée sur l'ensemble des répétitions, en fonction du nombre de classes initial de l'algorithme, est donné dans la Figure 3.

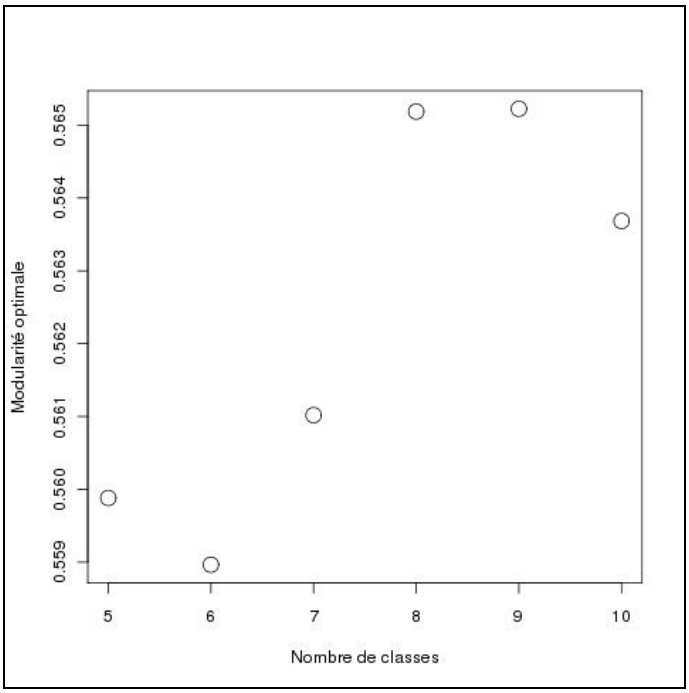

Figure 4 : Evolution de la modularité optimale trouvée en fonction du nombre de classes

La modularité optimale est obtenue pour une initialisation à 9 classes ; la classification obtenue pour cette solution optimale comporte seulement 8 classes. La Figure 4 ainsi que le fait que le nombre de classes finalement fourni par l'algorithme est inférieur au nombre de classes initial (9) montre que la recherche d'une bonne classification dont le nombre de classes serait supérieur à 10 est inutile.

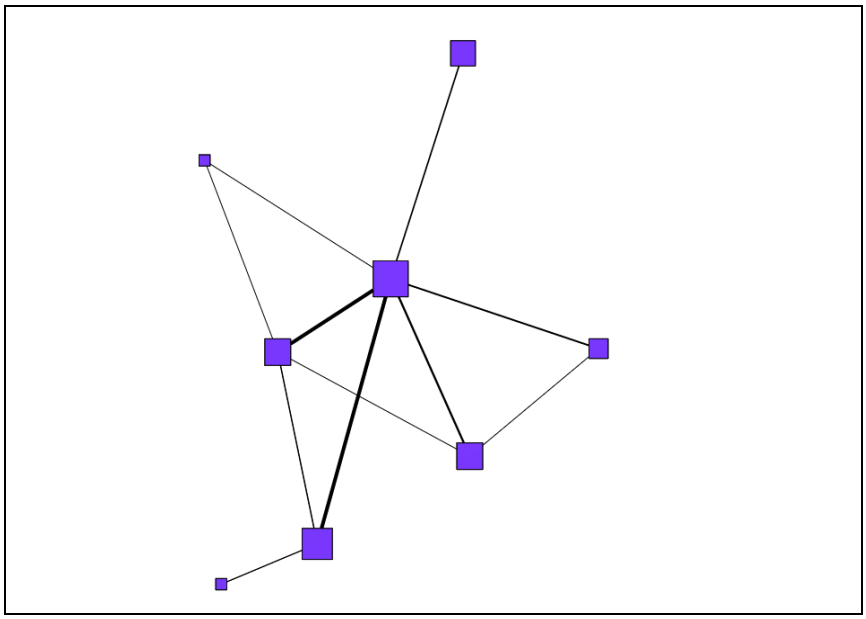

Figure 5 : Représentation du graphe des classes par un algorithme de forces : chaque carré représente une classe dont la surface est proportionnelle à l'effectif de la classe ; l'épaisseur des arêtes est proportionnelle au poids total des arêtes entre les deux classes.

La Figure 5 est le résultat de la représentation du graphe des classes par un algorithme de forces[2]. On y remarque un groupe important, central, autour duquel gravitent les autres personnages du roman et on identifie également aisément 3 autres groupes de taille importante et qui permettent de structurer les relations autour du groupe principal.

La combinaison de la représentation du graphe des classes avec l'algorithme de représentation d'un graphe en classes donne finalement la représentation fournie dans la Figure 6 que l'on peut la comparer à la représentation initiale du graphe donnée par la Figure 3.

[2] Cette représentation a également été réalisée à l'aide du logiciel Tulip.

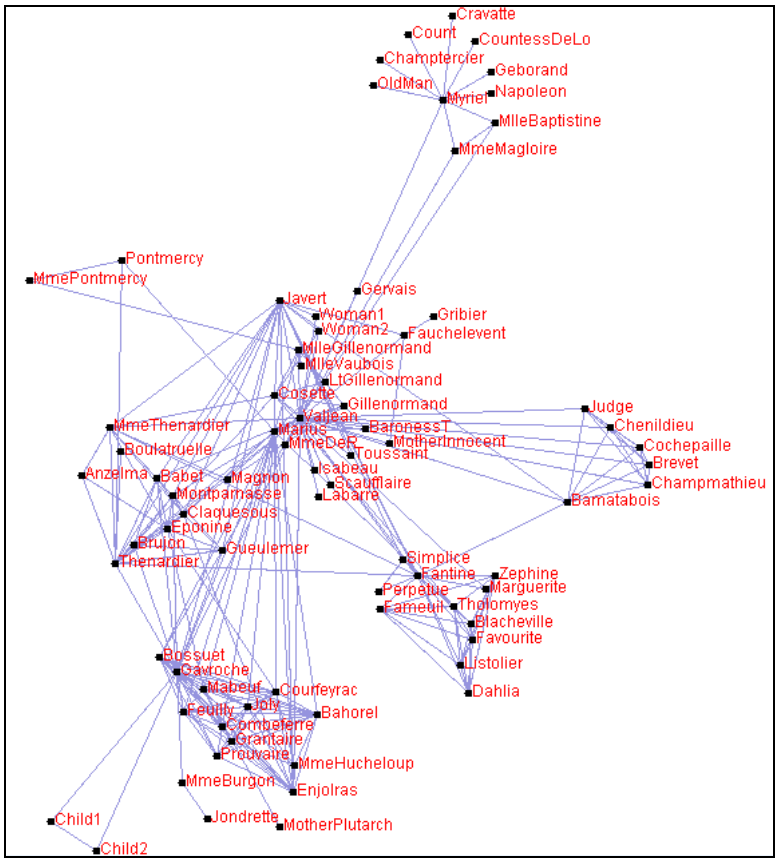

Figure 6 : Représentation du graphe de co-apparition du roman « Les Misérables » obtenu par combinaison d'un algorithme de classification des sommets et d'un algorithme de représentation d'un graphe en classes.

Si certains groupes denses sont facilement identifiables dans les deux représentations (comme par exemple celui qui est organisé autour de Myriel), la représentation du graphe en classes permet d'identifier plus clairement des groupes dans la partie la plus dense du graphe. Elle met bien en valeur la position centrale de certains personnages du groupe principal : Valjean, Cosette, Marius, Javert (…), ainsi que la proximité de ces personnages centraux. On rappelle que la trame principale du roman est organisée autour de la vie de Jean Valjean, ancien forçat, qui mourra dans les bras de sa jeune protégée Cosette et du jeune étudiant tombé amoureux d'elle, Marius, après avoir été poursuivi durant des années par l'intransigeant Javert. Or, hormis pour Valjean, cette position centrale n'est pas aussi claire dans le graphe initial. Les trois sous-groupes principaux se présentent comme autant d'histoires secondaires gravitant autour de cette histoire principale : le groupe

important situé le plus en bas du graphe est organisé autour de Gavroche, gamin des rues, capable de gestes de générosité envers Mabeuf (aussi dans ce groupe) et deux enfants perdus, appelés ici « Child 1 » et « Child 2 », qui sont ses frères. Un second groupe principal se situe au-dessus de celui-ci, à droite. Il est organisé autour de Fantine, ouvrière obligée de confier sa fille, Cosette à des inconnus. On retrouve justement ces inconnus dans le troisième groupe important, situé à gauche du groupe principal : ce sont les Thénardier ; c'est leur présence dans ce groupe qui influence la position de Cosette dans la partie gauche de son groupe.

Ainsi, la représentation du graphe en classes est facilement mise en relation avec l'histoire du roman. Elle permet d'en identifier les personnages principaux et les histoires secondaires. Sur cet exemple simple, on comprend l'apport d'une phase préalable de classification à la représentation du graphe afin de diriger la structuration du dessin et de simplifier l'interprétation.

4.2. *Représentation d'un réseau de collaborations scientifiques*

Nous traitons, dans cette partie un autre exemple simple mais plus réaliste destiné à montrer que l'approche est utilisable pour des graphes de plusieurs centaines de sommets. L'exemple traité ici est issu de (Newman, 2006)[3] : il s'agit d'un réseau de collaborations scientifiques d'individus travaillant autour de la thématique des réseaux sociaux, modélisé sous la forme d'un graphe pondéré (non orienté). La plus grande composante connexe du graphe contient 379 scientifiques. Une représentation basée sur un algorithme de forces est donnée dans les Figures 7 et 8.

[3] Les données sont disponibles à l'adresse http://www-personal.umich.edu/~mejn/netdata/netscience.zip.

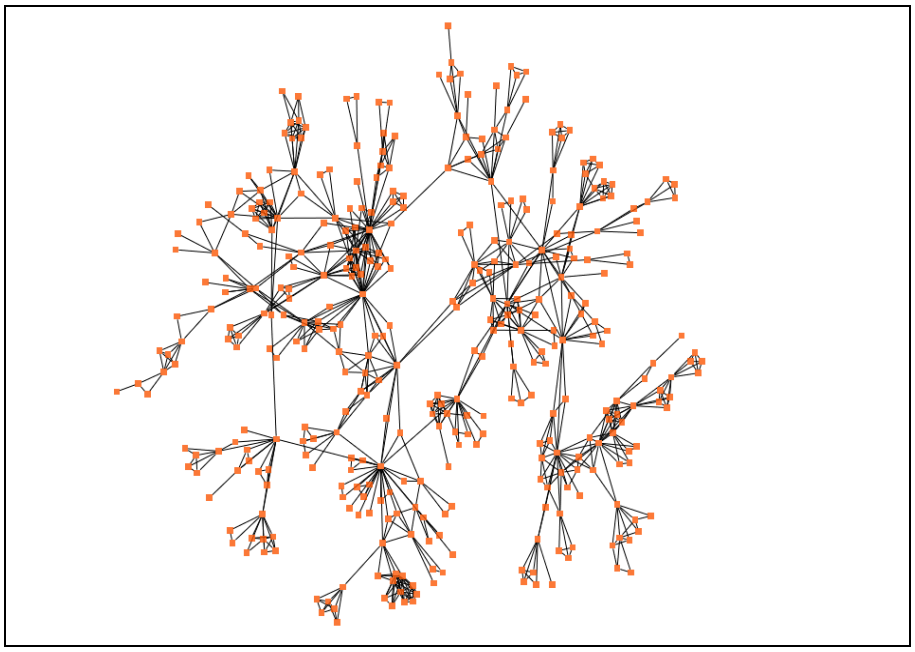

Figure 7 : Représentation du réseau de collaboration scientifique par un algorithme de forces (sans légende).

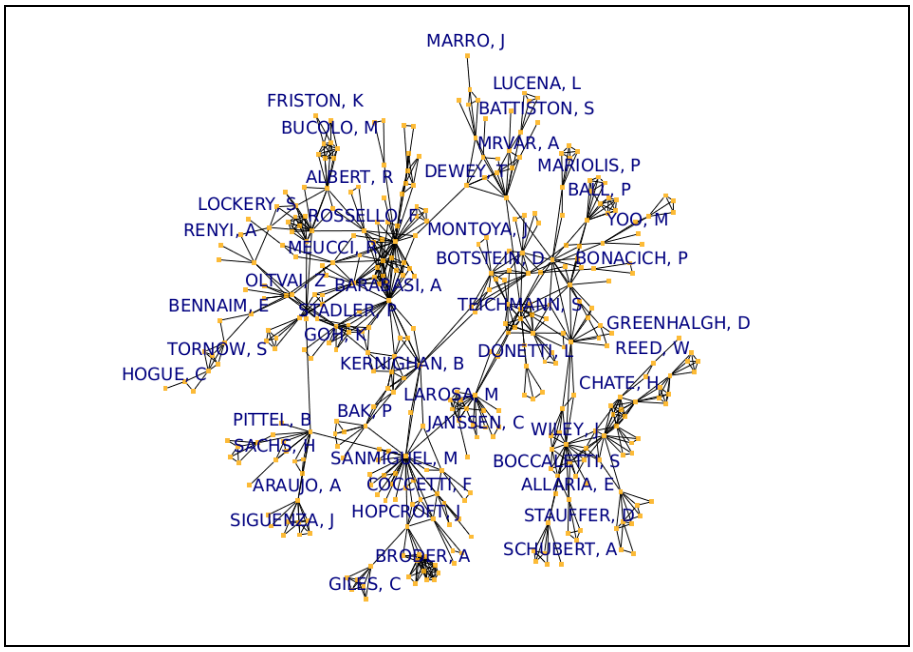

Figure 8 : Représentation du réseau de collaborations scientifiques par un algorithme de forces (avec légende).

Sur ce graphe, nous avons, tout d'abord, mis en œuvre l'algorithme de recuit simulé pour optimisation de la modularité. Afin d'obtenir des classes de petites tailles, nous avons opté pour un nombre de classes égal à 20 ; plusieurs initialisations aléatoires ont été effectuées et plusieurs valeurs du paramètre γ de l'algorithme de recuit simulé ont été testées. La meilleure classification obtenue avait une modularité finale égale à 0,816, ce qui est une valeur optimale attendue pour ce type de réseau (voir (Newman *et al.*, 2004)). La Figure 9 montre l'évolution de la modularité lors des itérations de l'algorithme de recuit simulé : la vitesse de convergence de l'algorithme est forte avant de se stabiliser vers 50 000 itérations.

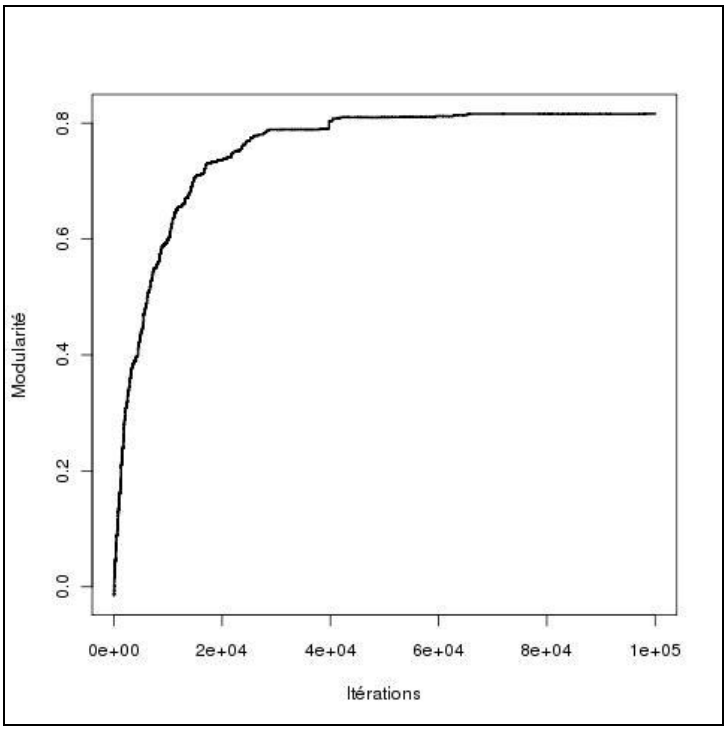

Figure 9 : Evolution de la modularité lors des itérations de l'algorithme de recuit simulé.

La Figure 10 donne la représentation, par un algorithme de forces, du graphe des classes. On y remarque des groupes isolés et des groupes très centraux autour desquels le réseau est structuré en étoile. Ce phénomène était déjà perceptible sur la représentation initiale du graphe mais il est, sur la représentation de la classification, plus évident et lisible, démontrant ainsi la validité et l'utilité d'une telle approche pour l'utilisateur.

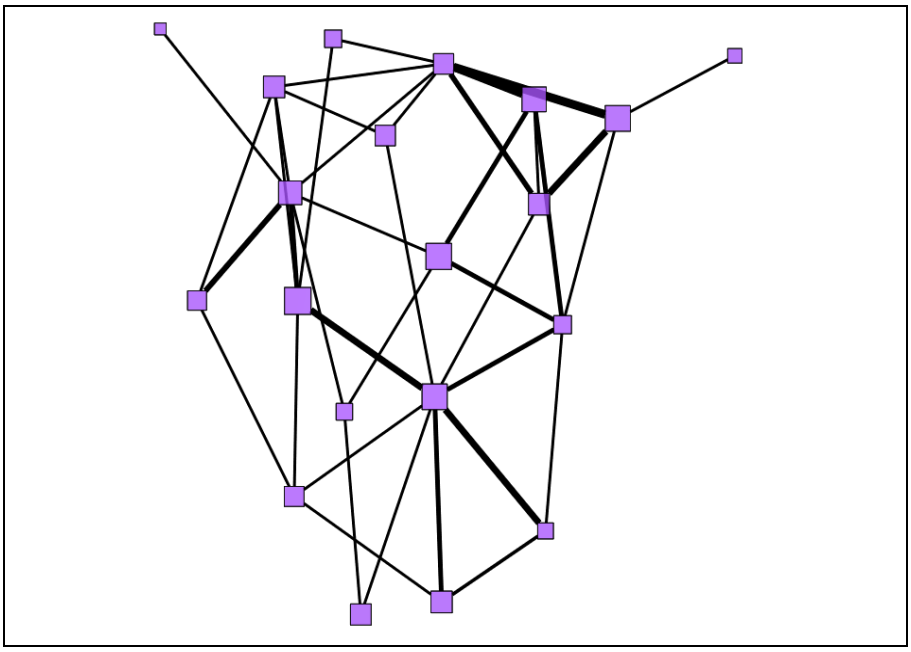

Figure 10 : Représentation du graphe des classes du réseau de collaborations scientifiques.

Enfin, notre approche permet d'obtenir la représentation finale des Figures 11 et 12. On y voit clairement l'avantage de la représentation du graphe entier plutôt que du graphe de classes : les densités locales d'arêtes sont plus facilement identifiables, que ce soit à l'intérieur des classes ou entre celles-ci. Par rapport à la représentation initiale, la représentation peut sembler perdre légèrement en clarté mais elle présente l'avantage de mettre en valeurs des communautés et les relations entre celles-ci. C'est une information d'importance pour l'utilisateur (sociologue, biologiste, …), qui peut ainsi directement visualiser sa classification sur le graphe. L'information donnée par la visualisation directe des sommets permet de comprendre les raisons de la formation des classes et de leur organisation dans le plan.

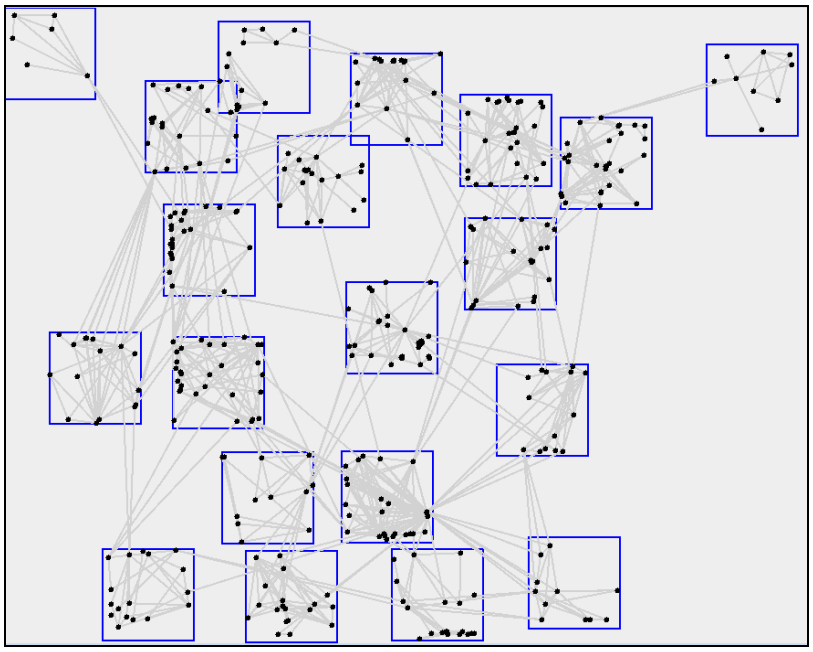

Figure 11 : Représentation globale du réseau de collaborations scientifiques : les zones de localisation des classes sont indiquées.

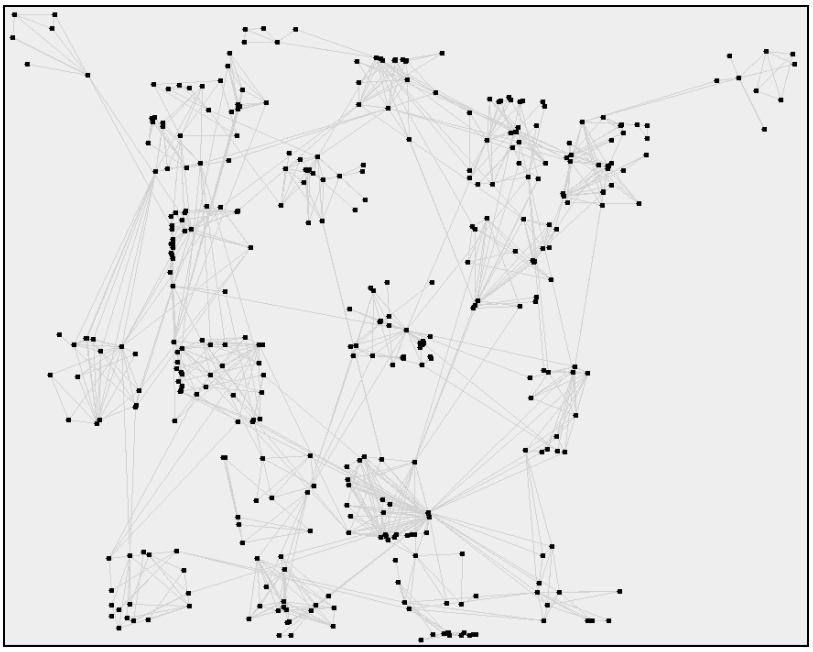

Figure 12 : Représentation globale du réseau de collaborations scientifiques : les zones de localisation des classes ne sont pas indiquées.

5. Conclusion

Le travail présenté ici décrit donc la combinaison d'une méthode de classification de sommets d'un graphe et d'une méthode adaptée de visualisation de graphes en classes pour donner une représentation simplifiée et lisible des grandes structures de relations existant dans un graphe. La méthode de classification des sommets, basée sur un algorithme stochastique et l'utilisation d'un critère de qualité spécifique au graphe permettent d'utiliser cette approche pour la représentation de graphes de taille importante. L'application dans des domaines tels que la classification de documents ou de réseaux biologiques est en cours d'investigation : ces domaines sont des challenges importants pour la fouille de graphes par la taille des données qu'elles mettent en œuvre. Enfin, une autre voie d'investigation consiste en l'utilisation, à la place d'une classification, d'un algorithme d'organisation des sommets du graphe (par exemple, les cartes auto-organisatrices de Kohonen) : de tels algorithmes adaptés à des données de type graphes ont été présentés dans (Villa *et al.*, 2007) et (Rossi *et al.*, 2010). Une approche similaire à celle que nous présentons ici pourrait être mise en œuvre pour la représentation de graphes, permettant d'éviter la phase de choix de la position des classes (ou de représentation

du graphe des classes) en la remplaçant par une carte définie a priori. Cependant, l'avantage d'un choix automatique de la position des classes est contrebalancé par la contrainte que représente la forme de la carte qui doit être choisie a priori et qui peut être peu adaptée au plongement du graphe étudié.

6. Bibliographie

D. Aubert, « Tulip ». *Proceedings of the Graph Drawing*, P. Mutzel, M. Jünger, S. Leipert (Eds.), Lecture Notes in Computer Science, vol. 2265, Springer-Verlag, 2001, p. 335-337.

A. Capoccia, V. Servedioa, G. Caldarellia, F. Colaiorib, « Detecting communities in large networks », *Physica A, Statistical Mechanics and its Applications*, vol. 352, n° 2-4, 2005, p. 669-676.

J.H. Chuang, C.C. Lin, H.C. Yen, « Drawing graphs with nonuniform nodes using potential fields ». *Proceedings of the Graph Drawing*, Perugia, Italy, G. Liotta (Eds.), 2004, p. 460-465.

A. Condon, R.M. Karp, « Algorithms for graph partitioning on the planted partition model », *Random Structures and Algorithms*, vol. 18, n° 2, 2001, p. 116–140.

Y. Dong, Y. Zhuang, K. Chen, X. Tai, « A hierarchical clustering algorithm based on fuzzy graph connectedness », *Fuzzy Sets and Systems*, vol. 157, n° 13, 2006, p. 1760-1774.

P. Eades, « A heuristic for graph drawing ». *Congressus Nutnerantiunt*, vol. 42, 1984, p. 149-160.

P. Eades, M.L. Huang, « Navigating clustered graphs using force-directed methods ». *Journal of Graph Algorithms and Applications*, 2000, p. 157-181.

T. Fruchterman, E. Reingold, « Graph drawing by force-directed placement ». *Software Practice and Experience*, vol. 21, 1991, p. 1129-1164.

S. Hachul, M. Juger, « An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs », *Proceedings of the Graph Drawing 2005*, Limerick, Ireland, Lecture Notes in Computer Science, vol. 3843, Springer-Verlag, 2006, p. 235-250.

T. Kamada, S. Kawai, « An algorithm for drawing general undirected graphs ». *Information Processing Letters*, vol. 31, 1989, p. 7-15.

S. Karouach, B. Dousset, « Visualisation de relations par des graphes interactifs de grande taille », *Journal of Information Sciences for Decision Making*, vol. 6, n° 57, Mars 2003.

S. Kirpatrick, C.D. Gelatt, M.P. Vecchi, « Optimization by simulated annealing », *Science*, vol. 220, n° 4598, 1983, p. 671-680.

S. Lakroum, V. Devlaminck, P. Terrier, P. Biela Enberg, J.G. Postaire, « Clustering of the Poincare vectors », *Proceedings of IEEE International Conference on Image Processing ICIP 2005*, September 11-14, 2005, Genoa, Italy.

S. Lehmann, L.K. Hansen, « Deterministic modularity optimization », *The European Physical Journal B*, vol. 60, n° 1, 2007, p. 83-88.

H. von Luxburg, « A tutorial on spectral clustering », *Statistics and Computing*, vol. 17, 2007, p. 395-416.

W.K. Hastings, « Monte Carlo sampling methods using Markov chains and their applications », *Biometrika*, vol. 57, n° 1, 1970, p. 97-109.

D.E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing*, Addison-Wesley, Reading, MA, 1993.

M. Newman, « Finding community structure in networks using the eigenvectors of matrices », *Pysical Review E*, vol. 74, n° 036104, 2006.

M. Newman, M. Girvan, « Finding and evaluating community structure in networks », *Physical Review E*, vol. 69, n° 026113, 2004.

A. Noack, « An Energy Model for Visual Graph Clustering ». *Proceedings of the Graph Drawing*, Perugia, Italy, G. Liotta (Eds.), 2004, p. 425-436.

J. Reichardt, S. Bornholdt, « Statistical mechanics of community detection », *Physical Review E*, vol. 74, n° 016110, 2006.

F. Rossi et N. Villa-Vialaneix, « Optimizing an organized modularity measure for topographic graph clustering : a deterministic annealing approach », *Neurocomputing*, vol. 73, n° 7-9, 2010, p. 1142-1163.

S.E. Schaeffer, « Graph clustering », *Computer Science Review*, vol. 1, 2007, p. 27-64.

Q.D. Truong, T. Dkaki, P.J. Charrel, « An energy model for the drawing of clustered graphs », *Actes du Vème colloque international VSST*, Marrakech, Maroc, 21-25 octobre 2007.

N. Villa, F. Rossi, « A comparison between dissimilarity SOM and kernel SOM for clustering the vertices of a graph », *Proceedings of WSOM 2007*, Bielefeld, Allemagne, September, 3-6, 2007.