HAL
open science

Un algorithme stable de décomposition pour l'analyse des réseaux sociaux dynamiques

Romain Bourqui, Paolo Simonetto, Fabien Jourdan

To cite this version:

Romain Bourqui, Paolo Simonetto, Fabien Jourdan. Un algorithme stable de décomposition pour l'analyse des réseaux sociaux dynamiques. Revue des Nouvelles Technologies de l'Information, 2009, pp.337-348. hal-00406437

HAL Id: hal-00406437

https://hal.science/hal-00406437v1

Submitted on 22 Jul 2009

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.

Un algorithme stable de décomposition pour l'analyse des réseaux sociaux dynamiques

Romain Bourqui*, Paolo Simonetto**
Fabien Jourdan**

*LaBRI, Université Bordeaux 1, 351, cours de la Libération F-33405 Talence cedex
{bourqui, simonett}@labri.fr
http://www.labri.fr/perso/{bourqui,simonett}
**INRA, UMR1089, Xénobiotiques, F-31000 Toulouse, France
Fabien.Jourdan@toulouse.inra.fr http://www.lirmm.fr/ fjourdan

Résumé. Les réseaux dynamiques soulèvent de nouveaux problèmes d'analyses. Un outils efficace d'analyse doit non seulement permettre de décomposer ces réseaux en groupes d'éléments similaires mais il doit aussi permettre la détection de changements dans le réseau. Nous présentons dans cet article une nouvelle approche pour l'analyse de tels réseaux. Cette technique est basée sur un algorithme de décomposition de graphe en groupes chevauchants (ou chevauchement). La complexité de notre algorithme est $O(|E| \cdot deg^2_{max} + |V| \cdot log(|V|)))$. La faible sensibilité de cet algorithme aux changements structuraux du réseau permet d'en détecter les modifications majeures au cours du temps.

1 Introduction

Les graphes sont utiles dans de nombreux domaines tels que la biologie, la micro-électronique, les sciences sociales, l'extraction de connaissance mais aussi l'informatique (e.g. Newman et Girvan (2004); Newman (2004); Palla et al. (2007); Suderman et Hallett (2007)). Il existe notamment un certain nombre de travaux portant sur la détection de communautés dans les réseaux. Par exemple, en sciences sociales, les personnes ayant les mêmes centres d'intérêt ou en biologie, les enzymes d'un réseau métabolique intervenant dans un processus commun (e.g. Newman et Girvan (2004); Newman (2004); Palla et al. (2007); Bader et Hogue (2003)). La détection de communautés offre deux atouts majeurs. En effet, elle permet non seulement une analyse initiale des données mais surtout elle permet de construire une abstraction visuelle.

Trouver des groupes (ou communautés) dans un réseau est généralement traduit en un problème de décomposition de graphe. Les algorithmes de décomposition recherchent des groupes d'éléments (ou *clusters*) ayant une (ou plusieurs) propriété(s) commune(s). Le critère le plus largement admis pour qu'un ensemble de groupes forme une « bonne » décomposition du graphe est que la densité de chaque groupe soit élevée mais aussi que la densité entre les différents groupes soit faible.

Le problème qui consiste à extraire des communautés est rendu plus difficile si l'on s'intéresse aux réseaux dynamiques. Les réseaux dynamiques sont de plus en plus fréquents du fait notamment de l'amélioration des techniques d'acquisitions ou encore de l'augmentation du

nombre de bases de données, que ce soit en biologie (e.g. l'évolution d'un organisme en fonction de son environement) ou encore en sciences sociales (e.g. réseau de co-citation, réseau des acteurs d'Hollywood). Par conséquent, il est nécessaire de pouvoir non seulement identifier les communautés de ces réseaux mais aussi de pouvoir détecter leurs changements structuraux (e.g. disparitions/créations de communautés). La décomposition des réseaux dynamiques est actuellement un domaine en pleine effervescence (e.g. Popescul et al. (2000); Li et Yoo (2005); Gaertler et al. (2006); Chakrabarti et al. (2006); Hübner (2008)). Tandis que certaines de ces approches prennent en compte l'attribut temporel pour décomposer le graphe (Popescul et al., 2000; Li et Yoo, 2005), d'autres tentent de trouver des groupes évoluant peu dans le temps (Gaertler et al., 2006; Chakrabarti et al., 2006; Hübner, 2008).

Dans cet article, nous nous intéressons à une méthode d'analyse des réseaux dynamiques basée sur la décomposition. Notre méthode consiste à «discrétiser» le réseau dynamique en un ensemble de réseaux statiques. Puis, nous appliquons un algorithme de décomposition en groupes chevauchants sur chacun de ces réseaux statiques. Afin de détecter les changements majeurs dans le réseau étudié, nous comparons ensuite les décompositions obtenues en utilisant une mesure de similarité.

Cet article est organisé comme suit : dans la section 2, nous présentons une vue d'ensemble de notre méthode, puis dans la section 3 nous décrivons notre algorithme de décomposition, et enfin nous évaluons la qualité et la «stabilité» de cet algorithme de décomposition sur un réseau social de référence dans la section 4.

2 Vue d'ensemble de la méthode

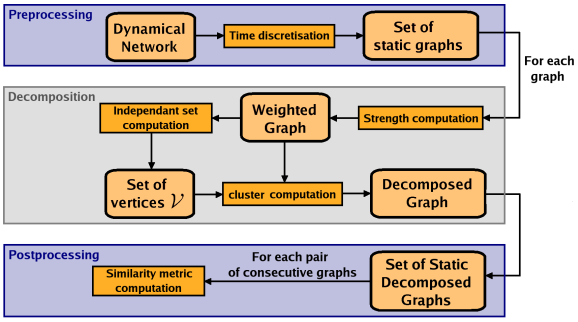

FIG. 1 – *Description des trois étapes de la méthode.*

La figure 1 illustre la méthode que nous présentons dans cet article. La première étape de cette méthode consiste à transformer un réseau dynamique en un ensemble de réseaux statiques. Si l'on considère un graphe dynamique G défini sur un intervale de temps $[0..T]$, cette transformation consiste à construire un ensemble de graphes statiques $\{G_{[0,\epsilon[}, ..., G_{[T-\epsilon,T]}\}$, où ϵ est le facteur de «discrétisation» et $G_{[t,t+\epsilon]}$ est le graphe statique correspondant à la période de temps $[t, t+\epsilon]$ (i.e. le graphe contenant les sommets et arêtes du graphe dynamique G intervenant dans la période $[t, t+\epsilon]$). L'idée principale de notre approche est que si le graphe évolue peu (et si le facteur de discrétisation est correctement choisi) alors deux graphes sta-

tiques correspondant à deux périodes consécutives ont des structures topologiques proches. Pour comparer les topologies de ces graphes, nous utilisons tout d'abord un algorithme de décomposition puis nous utilisons une mesure de similarité de décompositions de graphes. Afin que notre méthode soit efficace, nous devons garantir la similarité de décompositions obtenues sur deux graphes dont les topologies sont proches.

Dans notre méthode, la qualité du résultat dépend grandement du facteur de discrétisation choisi. Cependant, nous ne nous intéressons pas ici à l'étude de l'impacte du facteur de discrétisation qui est actuellement le sujet d'un autre travail. Dans cet article, nous nous intéressons tout particulièrement à un algorihme de décomposition de graphe peu sensible à de faibles modifications structurelles. Cet algorithme a trois étapes principales : premièrement, nous calculons la métrique *Strength* (de Auber et al. (2003)) sur les arêtes et sommets, puis nous cherchons un ensemble maximal de sommets indépendants (i.e. de sommets à distance au moins 2), enfin nous construisons les groupes «autour» des sommets de l'ensemble indépendant.

3 Algorithme

3.1 Métrique *Strength*

Pour définir la métrique Strength (Auber et al., 2003; Chiricota et al., 2003), nous devons introduire quelques notations. Soient u et v deux sommets du graphe, on note $M_u(v) = N_G(v) \setminus (N_G(u) \cup \{u\})$ l'ensemble des voisins de v (en excluant u) qui ne sont pas voisins de u, et on note $W_{uv} = N_G(u) \cap N_G(v)$ l'ensemble des sommets voisins de u et de v. Soient A et B deux ensembles de sommets, on note $E(A, B)$ l'ensemble des arêtes reliant un sommet de A à un sommet de B. Enfin, $s(A, B) = |E(A, B)|/(|A| \cdot |B|)$ est le ratio entre le nombre d'arêtes reliant A et B et le nombre maximal d'arêtes qu'il pourrait y avoir entre ces deux ensembles [1]. La métrique Strength d'une arête $e = (u, v)$, notée $w_s(e)$ est :

$$w_s(e) = \frac{\gamma_{3,4}(e)}{\gamma_{max}(e)} \tag{1}$$

où :

$$\begin{aligned}\gamma_{3,4}(e) = {} & |W_{uv}| + |E(M_v(u), M_u(v))| + |E(M_v(u), W_{uv})| \\ & + |E(W_{uv}, M_u(v))| + |E(W_{uv})|\end{aligned} \tag{2}$$

$$\begin{aligned}\gamma_{max}(e) = {} & |M_v(u)| + |W(u, v)| + |M_u(v)| + |M_v(u)||M_u(v)| \\ & + |M_v(u)||W_{uv}| + |W_{uv}||M_u(v)| + |W_{uv}|(|W_{uv}| - 1)/2\end{aligned} \tag{3}$$

Cette métrique compte donc pour chaque arête le nombre de cycles de longueur 3 et 4 passant par cette arête et normalise cette valeur par le nombre maximum possible de tels cycles (la figure 2 illustre les 3 ensembles formés par les voisinnage de u et v). Enfin, on peut définir la valeur de Strength d'un sommet comme suit :

$$w_s(u) = \frac{\sum_{e \in adj(u)} w_s(e)}{deg(u)}$$

[1] Lorsque $A = B$, on a $s(A, A) = s(A) = 2 \cdot |E(A)|/(|A| \cdot (|A| - 1))$

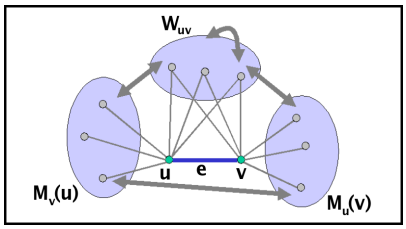

FIG. 2 – *La métrique Strength de l'arête* $e = (u, v)$ *est calculée en comparant le nombre de cycles de tailles* 3 *et* 4 *passant par cette arête au nombre maximal de tels cycles.*

où $adj(u)$ est l'ensemble des arêtes adjacentes à u et $deg(u)$ est le degré de u. La métrique Strength permet donc de quantifier la cohésion du voisinage d'une arête ou d'un sommet. Le temps nécessaire pour calculer cette métrique est $O(|E| \cdot deg_{max}^2)$ où deg_{max} est le degré maximum du graphe.

3.2 Extraction d'un ensemble indépendant maximal

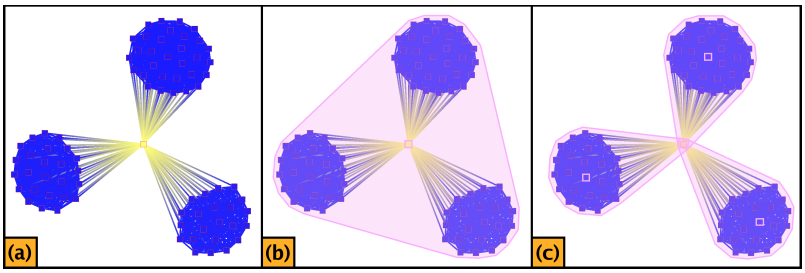

FIG. 3 – *(a) Sous-graphe du « graphe d'Hollywood » où les sommets représentent des acteurs et deux sommets sont reliés si les acteurs correspondant ont participé à un film commun. La couleur de chaque sommet correspond à sa valeur de Strength, de la plus faible en jaune à la plus forte en bleu. (b) Si le sommet entouré en rose est choisi comme « centre » de la communauté, on pourrait alors obtenir un groupe unique contenant tous le réseau. (c) Dans le cas où, les sommets « centres » sont des sommets de fortes valeurs de Strength, on obtient alors trois groupes correspondant au trois films de ce réseau.*

Pour identifier les «centres» des communautés du réseau étudié, nous utilisons une méthode inspirée du *MISF* (pour Maximal Independant Set Filtering) de Gajer et Kobourov (2000). Cette technique consiste à extraire un ensemble maximal $\mathcal{V}$ de sommets du réseau tel que $\forall u, v \in V$, $dist_G(u, v) \geq 2$. Elire un ensemble de sommets à distance au moins 2 dans le graphe permet tout d'abord d'obtenir un bon échantillonnage du réseau. D'autre part, cela permet de déterminer automatiquement le nombre de groupes en fonction de la topologie du graphe. Enfin, cette technique nous garantit l'unicité de chaque groupe trouvé par notre algorithme (i.e. deux groupes obtenus par notre approche ne peuvent être identiques).

Etant donné que les sommets de $\mathcal{V}$ seront les « centres » de nos communautés, ces sommets ne doivent pas être des « pivots » du réseau. Si on utilise l'un de ces sommets pivots comme « centre » d'un groupe, ce groupe pourrait alors contenir plusieurs communautés (cf figure 3).

La métrique Strength calculée sur les sommets nous permet d'identifier ces pivots. En effet, les sommets situés à l'intersection de plusieurs communautés ont des valeurs de Strength relativement faible. Les sommets de fortes valeurs de Strength doivent donc être ajoutés prioritairement à l'ensemble $\mathcal{V}$. Nous utilisons l'algorithme 1 pour extraire un tel ensemble de sommets.

Input : A graph $G = (V, E)$
Output : A maximal set $\mathcal{V}$ of vertices at distance at least 2
vector<node> $sorted_nodes$;
sortNodeWithStrength(G, $sorted_nodes$);
for *unsigned int* i *from* 0 *to (number of vertices in* G*)* **do**
  node u = $sorted_nodes[i]$;
  if u *in* G **then**
    append($\mathcal{V}$,u);
    foreach *node* v *in neighborhood of* u **do**
      remove(G, v);
    end
    remove(G, u);
  end
end

Algorithme 1 : Extraction de l'ensemble $\mathcal{V}$. La fonction sortNodeWithStrength(G, $sorted_nodes$) trie les sommets de G par ordre décroissant de valeur de Strength et stocke le résultat dans $sorted_nodes$.

La complexité de l'algorithme de trie sortNodeWithStrength(Graph, vector<node>) est en temps $O(|V| \cdot log(|V|))$. En ce qui concerne la boucle **for**, on peut trivalement montrer que sa complexité en temps et en espace est $O(|V| + |E|)$. Le coût total de l'extration de l'ensemble $\mathcal{V}$ est réalisé en temps $O(|V| \cdot log(|V|) + |E|)$ et en espace $O(|V| + |E|)$.

3.3 Détection des groupes

L'algorithme 2 permet ensuite d'extraire les groupes en utilisant l'ensemble $\mathcal{V}$. Le principe de cet algorithme est de construire des « boules » de rayon 1 dans le graphe autour des sommets $\mathcal{V}$. Pour chaque sommet u de $\mathcal{V}$, si une arête (u, v) a une valeur de Strength supérieure à une borne ϵ fixée, alors cette arête doit être intra-communautaire et par conséquent le sommet v est ajouté à la communauté de u. La borne ϵ est calculée en fonction du nombre d'arêtes du graphe $G = (V, E)$ et du nombre d'arêtes du graphe complet à $|V|$ sommets. L'idée est que moins le réseau est dense, moins les communautés contenues dans ce réseau sont denses. Dans l'algorithme 2, la borne ϵ a été fixée empiriquement et peut être modifiée afin de construire des groupes plus ou moins tolérant aux « bruits ».

En ce qui concerne la complexité, pour chaque sommet u de $\mathcal{V}$, l'algorithme 2 parcourt toutes les arêtes adjacentes à u et a donc un coût en temps $O(deg(u))$. On obtient donc une

Input : A graph $G = (V, E)$, the Strength of each edge, a maximal set $\mathcal{V}$
Output : A set D of groups of vertices
double $\epsilon = 2 \cdot |E|/(|V| \cdot (|V| - 1))$;
foreach *node* u *in* $\mathcal{V}$ **do**
  Group $curGroup$ = createNewGroup();
  append($curGroup$, u);
  foreach *edge* $e = (u, v)$ *adjacent to* u **do**
    if $Strength(e) > \epsilon$ **then**
      append($curGroup$, v);
    end
  end
  append(D, $curGroup$);
end

Algorithme 2 : Construction des groupes de sommets.

complexité en temps de $O(\sum_{u \in \mathcal{V}} deg_u)$. Etant donné que $\sum_{u \in V} deg(u) = 2 \cdot |E|$, l'algorithme 2 est donc réalisé en temps $O(|E|)$ et en espace $O(|V| + |E|)$.

La complexité totale de notre algorithme de décomposition est donc en temps $O(|E| \cdot deg_{max}^2 + |V| \cdot log(|V|))$ et en espace $O(|V| + |E|)$.

4 Evaluation de l'algorithme

4.1 Cas d'étude

Nous avons choisi comme cas d'étude une sous-partie du jeu de données du concours InfoVis 2007 Contest (2007). Ce jeu de données est issu de la base de données cinématographique IMDb (pour *Internet Movie Database*). Afin de pouvoir réaliser notre étude, nous en avons extrait un sous-réseau de 432 films, 4025 acteurs. Nous avons ensuite contruit un graphe de la manière suivante : un sommet du graphe représente un acteur et deux sommets du graphe sont reliés par une arête si les acteurs correspondant ont participé à (au moins) un film commun. Nous avons ainsi obtenu un réseau de 4025 sommets et 41216 arêtes. La figure 4 montre le graphe ainsi construit. Nous avons choisi ces données puisqu'il est bien connu qu'elles contiennent des structures de communautés, typiquement chaque film est représenté par une clique (sous-graphe complet) et ces cliques peuvent partager des sommets (correspondant aux acteurs ayant participé à plusieurs films).

Afin d'évaluer la qualité de notre algorithme de décomposition, nous utilisons deux critères. Le premier de ces critères est la qualité de la décomposition produite par notre algorithme. Pour évaluer cette qualité, nous utilisons une généralisation de la mesure de qualité MQ introduite par Mancoridis et al. (1998) au cas des décompositions chevauchantes. Le deuxième critère est une mesure de la «sensibilité» de notre algorithme aux modifications structurelles du réseau. Pour cela, nous comparons la décomposition obtenue sur le graphe original à une décomposition obtenue après un certain nombre de modifications structurelles du réseau.

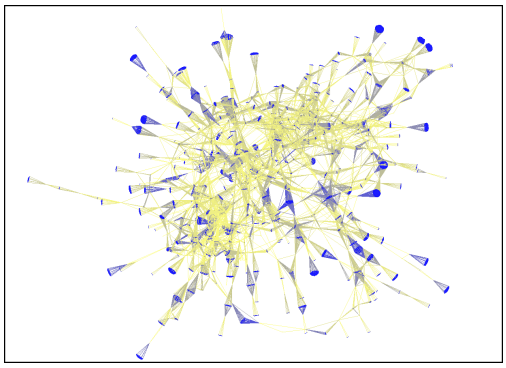

FIG. 4 – *Sous-graphe du « graphe d'Hollywood ». Ce sous-graphe correspond à un ensemble de* 432 *films et contient* 4025 *sommets (acteurs) et* 41216 *arêtes. La couleur de chaque sommet correspond à sa valeur de Strength (de la plus faible en jaune à la plus forte en bleu).*

4.2 Qualité de la décomposition

Le critère le plus largement admis pour qu'un ensemble de groupes forme une « bonne » décomposition du graphe est une densité intra-groupe élevée et une densité inter-groupes faible. Nous utilisons pour évaluer la qualité d'une décomposition une généralisation de la mesure de qualité MQ de Mancoridis et al. (1998) au cas des décompositions chevauchantes. Cette généralisation a été introduite par Bourqui et Auber (2008). Considérons un graphe $G = (V, E)$ et une décomposition $C = \{C_1, C_2, ..., C_k\}$ des sommets de G, MQ_{Over} est définie comme suit :

$$MQ_{Over} = MQ^{+} - MQ^{-}_{Over} \tag{4}$$

avec

$$MQ^{+} = \frac{1}{k} \sum_{i} s(C_i, C_i) \tag{5}$$

et

$$MQ^{-}_{Over} = \frac{1}{k(k-1)} \sum_{i} \sum_{j \neq i} s_{Over}(C_i, C_j) \tag{6}$$

où $s_{Over}(C_i, C_j) = \frac{|E(C_i, C_{j\setminus i})|}{|C_i| \cdot |C_{j\setminus i}|}$ et $C_{j\setminus i} = C_j \setminus (C_j \cap C_i)$. Dans cette équation, MQ^{+} représente la cohésion interne des groupes $C_1, ..., C_k$ tandis que MQ^{-}_{Over} représente la cohésion externe des groupes (ou inter-groupes)

Nous avons appliqué notre algorithme de décomposition au sous-graphe du graphe d'Hollywood (cf figure 4). La figure 5.(b) montre le résultat que nous avons obtenu sur ce graphe. Dans cette figure, chaque groupe trouvé par notre algorithme est entouré par une enveloppe convexe mauve. La valeur de MQ_{Over} de cette décomposition est 0.96 ce qui montre notre algorithme donne un excellent résultat sur ces données (selon la mesure MQ_{Over}). La figure 5.(a) montre la décomposition que nous considérons comme optimale, i.e. la décomposition où chaque groupe correspond à un film. On peut tout d'abord noter la similarité de ces deux décompositions. Lorsque l'on compare en détail les groupes obtenus, il s'avère que notre

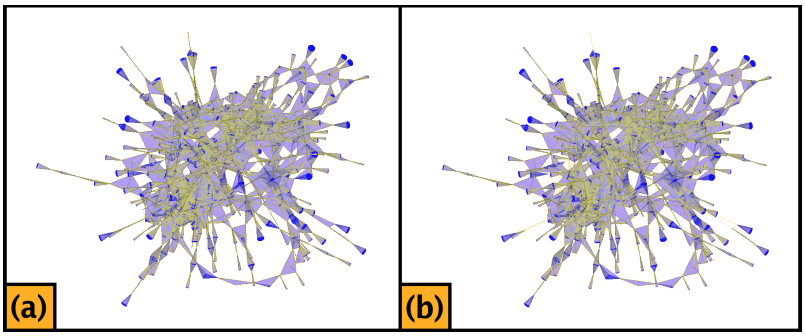

FIG. 5 – *Décompositions optimale (a) et obtenue par notre algorithme (b) du graphe de la figure 4. Chaque groupe est entouré par une enveloppe convexe mauve.*

algorithme trouve 421 groupes dont 404 sont en correspondance parfaite avec des groupes de la décomposition optimale. Notre algorithme permet donc de retrouver plus de 93% des groupes de la décomposition optimale (et près de 96% des groupes trouvés correspondent à des films). En ce qui concerne les 17 groupes de notre décomposition qui ne sont pas en correspondances parfaites, chacun de ces groupes est un sous-groupe de la décomposition optimale.

4.3 Sensibilité aux modifications

Afin d'évaluer la sensibilité de notre algorithme aux changements structuraux du réseau, nous comparons la décomposition de référence (obtenue sur le réseau original) aux décompositions obtenue après un certain nombre d'ajout et/ou de suppressions d'arêtes. Dans cette section, nous expliquons la mesure de similarité utilisée pour comparer deux décompositions, puis nous présentons les résultats que nous avons obtenus.

4.3.1 Mesure de similarité

Pour mesurer la similarité des deux décompositions, nous utilisons une métrique inspirée de celle de Brohée et van Helden (2006). Cette mesure est basée sur la *représentativité*. On dit que deux décompositions sont similaires si et seulement si la première est représentative de la seconde, et inversement.

Pour définir la représentativité de deux décompositions, nous devons tout d'abord définir la représentativité de deux groupes. Soient c_i et c_j deux groupes, on dit que le groupe c_i est représentatif du groupe c_j si et seulement si le groupe c_i est composé en majeure partie d'éléments de c_j. On définit *représentativité dirigée de groupes* (en anglais, *directed cluster representativeness*) comme suit :

$$\rho_{c_i \to c_j} = \frac{c_i \cap c_j}{|c_j|} \qquad \rho_{c_j \to c_i} = \frac{c_i \cap c_j}{|c_i|}$$

On appelle alors la *représentativité non-dirigée de groupes* (en anglais, *undirected cluster representativeness*) ou plus simplement la *représentativité de groupes* :

$$\rho_{c_i c_j} = \sqrt{\rho_{c_i \to c_j} \cdot \rho_{c_j \to c_i}}$$

Cela correspond à la moyenne géométrique (en anglais, *geometrical mean*)) des représentativités dirigées des groupes c_i et c_j.

On peut, de manière analogue, définir le degré de représentativité d'une décomposition par rapport à une autre. Considérons deux décomposition C et C', on dit que C est représentative de C' si et seulement si pour chaque groupe c' de la décomposition C', la décomposition C contient un groupe représentatif de c'. Etant donné que les groupes de « petites » tailles ont tendance à biaiser cette métrique, nous donnons plus d'importance aux représentativités des groupes de « grandes » tailles. On définit alors la *représentativité dirigée de décompositions* (en anglais, *directed clustering representativeness*) comme suit :

$$\sigma_{C \to C'} = \frac{\sum_{c_i \in C'} \max_{c_j \in C} \rho_{c_j c_i} |c_i|}{\prod_{c_i \in C'} |c_i|}$$

Cette formule correspond à la moyenne valuée (et normalisée) des meilleures représentativités de chaque groupe de C' par des groupes de C.

On peut alors définir la *représentativité non-dirigée de décompositions* (en anglais, *undirected clustering representativeness*), ou plus simplement la *représentativité de décompositions* comme suit :

$$\sigma_{CC'} = \sqrt{\sigma_{C \to C'} \cdot \sigma_{C' \to C}}$$

Une variation possible de cette métrique peut être obtenue en utilisant un produit simple en lieu et place de la moyenne géométrique lors du calcul de la représentativité non-dirigée. Cela permet de distinguer plus aisément les décompositions similaires des décomposition « différentes » puisque les « mauvaises » correspondances sont alors plus pénalisées. Nous utilisons dans la suite, cette variation de la métrique de similarité.

4.3.2 Résultats

Pour mesurer la sensibilité de notre algorithme de décomposition, nous commençons par générer un jeu de données à partir du graphe de la figure 4. Pour cela, nous utilisons l'algorithme 3. Cet algorithme permet de générer une collection de 100000 graphes.

Nous avons ensuite comparé la décomposition du graphe original avec celles obtenues sur les graphes de la collection $Collection$ générée par l'algorithme 3. La figure 6 montre les résultats que nous avons obtenus.

Dans la figure 6.(a), la courbe bleue montre le nombre moyen de correspondances parfaites de groupes en fonction du nombre d'arêtes supprimées ou ajoutées au réseau original, et le tracé rouge indique pour chacune de ces moyennes les écarts types observés. On peut remarquer sur cette figure que jusqu'à 2000 opérations, notre algorithme permet de trouver en moyenne entre 250 et 421 (i.e. le nombre de groupes dans la décomposition originale) correspondances parfaites entre la décomposition originale et les décompositions des graphes de la collection générée. D'autre part, on peut aussi remarquer que les écarts types sont relativement faibles, compris entre 0.44 et 10.34. Dans la figure 6.(b), la courbe bleue montre la valeur moyenne de la métrique de similarité en fonction du nombre d'arêtes supprimées ou ajoutées au réseau original, et le tracé rouge indique pour chacune de ces valeurs moyennes les écarts types observés. Les valeurs moyennnes de la métrique de similarité sont comprises sur ce jeu

```
Input : subgraph G = (V, E) of the Hollywood graph
Output : A set Collection of graphs
for unsigned int i = 0 to i == NB_TESTS do
    Graph H = G;
    for unsigned int j = 0 to j == MAX_OPERATIONS do
        Operation op = getOperation();
        if op == 'edge deletion' then
            node src = getRandomNode();
            node tgt = getRandomNode();
            edge e = edge(src, tgt);
            while e is not element of H do
                src = getRandomNode();
                tgt = getRandomNode();
                e = edge(src, tgt);
            end
            deleteEdge(H, e);
        end
        else                                /* op == 'edge addition' */
            node src = getRandomNode();
            node tgt = getRandomNode();
            edge e = edge(src, tgt);
            while e is element of H do
                src = getRandomNode();
                tgt = getRandomNode();
                e = edge(src, tgt);
            end
            addEdge(H, e);
        append(Collection, H);
    end
end
```

Algorithme 3 : Construction du jeu de données utilisé pour évaluer la sensibilité de notre algorithme de décomposition. La fonction getOperation() retourne 'edge addition' avec une probabilité 0.5, 'edge deletion' sinon. Les constantes NB_TESTS et $MAX_OPERATIONS$ ont été respectivement fixées à 50 et 2000.

de données entre 0.9 et 1. On obtient donc de bonnes valeurs de similarités entre la décomposition originale et les décompositions des graphes de la collection générée. En ce qui concerne les écarts types, ils sont compris entre 0.0002 et 0.007.

Si l'on considère une mesure de «stabilité» naïve comme le nombre de correspondances parfaites, notre algorithme permet en moyenne de préserver près de 78% des groupes. D'autre part, les valeurs moyennes des mesures de similarité sont elles aussi élevées montrant ainsi la stabilité de notre algorithme de décomposition.

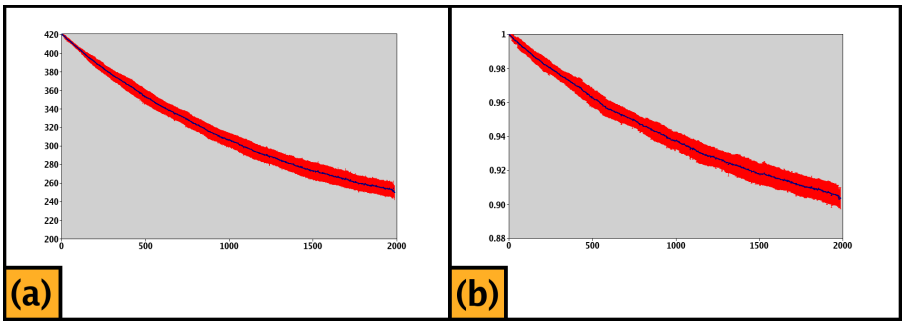

FIG. 6 – *(a) Nombre moyen de correspondances parfaites (en bleu) en fonction du nombre d'opérations d'addition et de suppression effectuées ainsi que l'écart type (en rouge) ; (b) Valeur moyenne de la métrique de similarité (en bleu) en fonction du nombre d'opérations d'addition et de suppression effectuées ainsi que l'écart type (en rouge).*

5 Conclusion

Dans cet article, nous avons présenté une nouvelle approche pour l'analyse de réseau dynamiques. Cette technique est basée sur la discrétisation du graphe dynamique en un ensemble de graphes statiques et sur la décomposition de graphe en groupes chevauchants de sommets. L'idée principale de notre approche est que si la structure du réseau évolue peu dans le temps alors les décompositions obtenues sur deux graphes correspondants à deux périodes de temps successives doivent contenir des structures de communautés similaires.

Afin de décomposer les graphes correspondant à chaque période de temps, nous donnons dans cet article un nouvel algorithme de décomposition de graphe dont la complexité en temps est $O(|E| \cdot deg^2_{max} + |V| \cdot log(|V|)))$. Etant donné que dans notre approche, deux graphes similaires doivent avoir des décompositions similaires, nous avons montré la faible sensibilité de notre algorithme aux modifications structurelles du réseau.

Enfin, nous donnons une généralisation de la mesure de similarité de Brohée et van Helden (2006) au cas des décompositions chevauchantes. Cela nous permet de comparer les décompositions obtenues sur deux graphes correspondants à deux périodes de temps successives, et par conséquent de détecter d'éventuels changements structuraux du réseau.

Références

Auber, D., Y. Chiricota, F. Jourdan, et G. Melançon (2003). Multiscale Visualization of Small-World Networks. In *Proc. of IEEE Information Visualization Symposium*, pp. 75–81.

Bader, G. D. et C. W. Hogue (2003). An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics 4*.

Bourqui, R. et D. Auber (2008). Analysis of 4-connected components decomposition for graph visualization. Technical report, LaBRI (http://www.labri.fr/).

Brohée, S. et J. van Helden (2006). Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics 7*(488). http://www.biomedcentral.com/1471-2105/7/488.

Chakrabarti, D., R. Kumar, et A. Tomkins (2006). Evolutionary clustering. In *Proc. of the 12th ACM SIGKDD int. conference on Knowledge Discovery and Data mining*, pp. 554–560.

Chiricota, Y., F. Jourdan, et G. Melançon (2003). Software Components Capture using Graph Clustering. In *11th IEEE Int. Workshop on Program Comprehension*.

Gaertler, M., R. Görke, D. Wagner, et S. Wagner (2006). How to Cluster Evolving Graphs. In *Proc. of the European Conference of Complex Systems (ECCS'06)*.

Gajer, P. et S. G. Kobourov (2000). GRIP: Graph dRawing with Intelligent Placement. In *Proc. Graph Drawing 2000 (GD'00)*, pp. 222–228.

Hübner, F. (2008). *The Dynamic Graph Clustering Problem -ILP-Based Approaches Balancing Optimality and the Mental Map*. Ph. D. thesis, Institut für Theoritische Informatik, Universität Karlruhe (TH).

InfoVis 2007 Contest (2007). IEEE InfoVis 2007 Contest: InfoVis goes to the movies. http://www.apl.jhu.edu/Misc/Visualization/.

Li, C. et J. Yoo (2005). A study of the effects of bias in criterion functions for temporal data clustering. In *Proc. of the 43rd annual Southeast regional conference*, pp. 85–89.

Mancoridis, S., B. S. Mitchell, C. Rorres, Y. Chen, et E. R. Gansner (1998). Using Automatic Clustering to Produce High-Level System Organizations of Source Code. In *IEEE Proc. Int. Workshop on Program Understanding (IWPC'98)*, pp. 45–53.

Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E 69*, 066133.

Newman, M. E. J. et M. Girvan (2004). Finding and evaluating community structure in networks. *Physical Review E 69*, 026113.

Palla, G., A.-L. Barabasi, et T. Vicsek (2007). Quantifying social group evolution. *Nature 446*, 664–667.

Popescul, A., L. H. Ungar, G. W. Flake, S. Lawrence, et C. L. Giles (2000). Clustering and Identifying Temporal Trends in Document Databases. In *Proc. of the IEEE Advances in Digital Libraries 2000*, pp. 173–182.

Suderman, M. et M. Hallett (2007). Tools for visually exploring biological networks. *Bioinformatics Advanced Access*.

Summary

Dynamical networks raise new analysis problems. An efficient analysis tool has to not only allow to split the network into groups of "similar" elements but also allow to detect changes in the network structure. In this article, we describe a new method for analyzing such dynamical networks. This technique is based on an algorithm of decomposition of graph into overlapping clusters. Time complexity of this algorithm is $O(|E| \cdot deg^2_{max} + |V| \cdot log(|V|)))$. The stability of that algorithm allows to detect the changes of the studied network over the time.