l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

Université du Maine

Thèse de Doctorat

Olivier GACH

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université du Maine
sous le label de l'Université de Nantes Angers Le Mans

École doctorale : Sciences et technologies de l'information et mathématiques (STIM)

Discipline : Informatique, section CNU 27
Unité de recherche : Laboratoire d'Informatique de l'Université du Maine (LIUM)
Laboratoire d'Étude et de Recherche en Informatique d'Angers (LERIA)

Soutenue le 3 décembre 2013

Algorithmes mémétiques de détection de communautés dans les réseaux complexes

Techniques palliatives de la limite de résolution

JURY

Président :	**M. Patrick GALLINARI**, Professeur, Université Pierre & Marie Curie
Rapporteurs :	**M. Gilles VENTURINI**, Professeur, Université de Tours **M. Jean-Loup GUILLAUME**, Maître de conférences HDR, Université Pierre & Marie Curie
Directrice de thèse :	**M^{me} Dominique PY**, Professeur, Université du Maine
Co-directeur de thèse :	**M. Jin-Kao HAO**, Professeur, Université d'Angers

Remerciements

Mes premières pensées vont aux opiniâtres et pétillants chercheurs de l'INRA, l'Institut Nationale de la Recherche Agronomique, tout particulièrement Thomas Schiex et Christine Gaspin qui m'ont définitivement conforté dans la vocation de chercheurs lors de mon stage en D.E.A. Bien que l'aboutissement en est lointain, cette thèse n'aurait peut être pas vu le jour sans eux.

Fort de son soutien sans faille et de ses encouragements, Jin-Kao, mon directeur de thèse disciplinaire, mérite d'être salué et remercié. Sa disponibilité, ses compétences et sa rigueur scientifique n'ont jamais été mises en défaut durant ces trois années de thèse. Je tiens aussi à saluer Dominique Py, la codirectrice de thèse, pour ses précieux conseils d'organisation et de rédaction.

Je tiens tout spécialement à saluer Patrick Donnet et Pierre Le Louarn, respectivement Directeur de l'IUT de l'Université du Maine et Chef du département GEA, pour leur appui indéfectible sans lequel le projet de cette thèse n'aurait jamais abouti, faute d'un dispositif organisationnel et financier me permettant de concilier le travail de recherche et une charge d'enseignement.

Enfin, mes dernières pensées vont à mon entourage, ma famille et ma compagne Christine à qui je dédie ce mémoire.

Table des matières

Introduction générale

Contexte

Les réseaux formés par l'observation d'un phénomène d'interaction, de natures très diverses aussi bien en biologie, en science de l'information qu'en sociologie, présentent des caractéristiques communes qui en font des objets d'étude à part entière, indépendamment de leur origine. Tout d'abord, certains éléments de réseau sont centraux, jouant un rôle particulier et agrégeant autour d'eux un grand nombre d'autres éléments. Ensuite, deux éléments quelconques ont une distance entre eux courte et il existe plusieurs chemins pour naviguer de l'un à l'autre, de sorte que cette propriété est préservée même si un élément non central est retiré du réseau. Enfin, le réseau possède des zones de forte densité constituées d'éléments fortement connectés entre eux. Les réseaux vérifiant ces propriétés remarquables, nommés graphes de terrain ou réseaux complexes, ont émergé récemment comme objets d'étude et sont modélisés sous la forme de graphe, de façon à étudier leurs propriétés structurelles et dynamiques.

La dernière propriété, qui se traduit par l'existence de sous-réseaux nommés communautés, modules ou simplement groupes, est importante car elle permet de mieux appréhender de très grands réseaux en identifiant des sous-parties dont les éléments sont semblables ou interagissent fortement entre eux. L'identification de ces groupes est très délicate et fait l'objet de nombreuses études depuis le début des années 2000. Ce problème, que nous nommons détection de communautés pour le distinguer du problème de partitionnement de graphe classique, est d'abord délicat parce qu'il n'y a pas de définition unique et pleinement opérationnelle d'une communauté. Ensuite, parce que les possibilités de découpage en communautés sont innombrables rendant l'examen de toutes ces partitions impossibles avec un réseau de seulement une centaine d'éléments. Enfin, ce problème est difficile, car même en présence d'une définition opérationnelle des communautés, les partitions en communautés y répondant pour un réseau donné sont très nombreuses, obligeant les experts du domaine d'application à étudier ces solutions selon la signification qu'ils donnent aux éléments et aux liens du réseau.

Problème et objectif

Notre étude se limite à la recherche de partitions en communautés, dans l'hypothèse où un élément du réseau appartient à une et une seule communauté. Les méthodes de résolution de ce problème sont variées, se partageant entre les approches locales définissant une communauté par rapport aux voisins, et les approches globales qui mesurent la pertinence d'une partition par une fonction donnant un score. La modularité est sans conteste la plus utilisée, mais aussi la plus contestée de ces mesures globales. À partir d'un des algorithmes parmi les plus efficaces pour optimiser la modularité, l'algorithme de Louvain, nous proposons de mesurer les défauts associés à cette mesure et de mettre au point des dispositifs qui en atténue les effets. Les méthodes de détection de communautés fondées sur une mesure globale souffrent d'un défaut majeur nommé limite de résolution, mais aussi d'une inconsistance de solution. Le premier rend difficile la capture à la fois des petites et des grandes communautés, car une mesure globale re-

présente un arbitrage numérique, parfois ténu, entre un grand nombre de petites communautés et un petit nombre de grandes communautés. Le second défaut concerne la diversité des solutions proposées par l'algorithme exécuté plusieurs fois avec un ordre différent d'éléments, pour un même réseau. Les solutions obtenues peuvent être très éloignées alors que fondamentalement le graphe représentant le réseau est le même. Notre objectif est de préserver la modularité dans une forme qui en atténue, voir supprime, ces deux défauts.

Démarche et contributions

Notre démarche est articulée en deux grandes étapes. D'abord, nous proposons de pousser très loin l'optimisation de la modularité pour observer, confirmer ou infirmer les défauts de celle-ci lorsque des valeurs optimales très difficiles à atteindre sont trouvées. Cet objectif passe par une amélioration de l'algorithme de Louvain, puis par le développement d'un algorithme mémétique, de nature génétique et utilisant une optimisation locale, seul capable d'atteindre ces optimums. Ensuite, nous proposons une condition de regroupement de communautés originale, le regroupement étant une opération essentielle dans l'algorithme de Louvain, qui réduit fortement la limite de résolution. Cette amélioration n'étant pas suffisante, nous présentons une technique ad hoc qui complète la condition de fusion. Enfin, l'algorithme mémétique de la première partie est associé à la condition de fusion pour produire de très bons résultats, particulièrement sur des réseaux artificiels générés avec leur solution selon un modèle réaliste qui les rapprochent des réseaux observés dans la réalité.

Organisation du manuscrit

Dans une partie liminaire, notre objectif est de brosser le cadre général de l'étude présentée, en posant les définitions communément admises et en décrivant les principales méthodes de détection de communautés, nécessaires à une bonne compréhension du développement de notre contribution. Nous nous attachons également dans cette partie à poser le cadre de l'expérimentation qui servira à évaluer nos algorithmes et à mesurer leur impact sur les défauts de la modularité. Notre contribution est décrite dans les deux parties suivantes qui, selon notre démarche, sont consacrées à l'optimisation pure de la modularité pour la première et aux palliatifs de la limite de résolution pour la seconde.

Notations

TABLE 1 – Notations pour les graphes

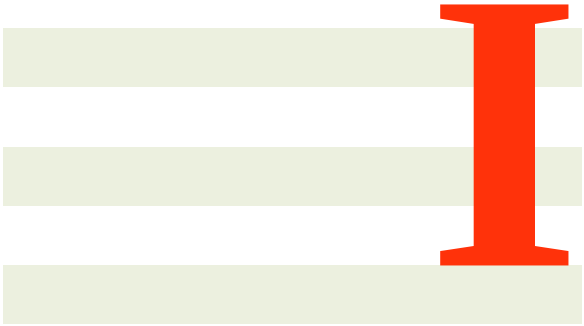
Symbole	Description
$G = (V, E)$	Graphe non orienté G formé d'un ensemble de nœuds V et d'un ensemble d'arêtes E
$G = (V, E, w)$	Graphe pondéré par la fonction de poids w qui associe un nombre réel à chaque arête
n	Nombre de nœuds du graphe G
m	Nombre d'arêtes du graphe G
$d(v)$	Degré du nœud v, c'est-à-dire nombre d'arêtes incidentes à v ou somme des poids de ces arêtes pour un graphe pondéré

TABLE 2 – Notations pour les partitions

Symbole	Description
$\mu(v)$ ou $\mu_{\mathcal{P}}(v)$	Communauté d'appartenance du nœud v dans une partition $\mathcal{P}$
k ou $k_{\mathcal{P}}$	Nombre de communautés d'une partition $\mathcal{P}$
Q ou $Q(\mathcal{P})$	Modularité de la partition $\mathcal{P}$

TABLE 3 – Notations pour les communautés

Symbole	**Description**
$q(C)$ ou $q(\mathcal{P}, C)$	Modularité de la communauté C dans la partition $\mathcal{P}$
$\delta_{int}(C)$	Densité du sous-graphe engendré par la communauté C
$d_{in}(v)$	Degré interne du nœud v défini comme le nombre ou la somme des poids des arêtes incidentes à v ayant leurs deux extrémités dans la communauté de v
$d_{out}(v)$	Degré externe du nœud v défini comme le nombre ou la somme des poids des arêtes incidentes à v ayant une extrémité en dehors de la communauté de v. On a $d(v) = d_{in}(v) + d_{out}(v)$
$d_{in}(C)$ et $d_{out}(C)$	Degré interne (resp. externe) d'une communauté C défini comme la somme des degrés internes (resp. externes) des nœuds la composant
$d(C)$	Degré d'une communauté défini comme la somme des degrés des nœuds la composant soit $d(C) = d_{out}(C) + d_{in}(C)$
$l(C)$	Nombre ou somme des poids des arêtes internes à la communauté C (ici chaque arête ne compte qu'une fois), soit $\frac{1}{2}d_{in}(C)$
$l(C, C')$	Nombre ou somme des poids des arêtes ayant une extrémité dans la communauté C et l'autre extrémité dans la communauté C'



Introduction

1

État de l'art

Sommaire

Dans ce chapitre, nous présentons le contexte des réseaux complexes modélisés par des graphes et le problème de la détection de communauté, dans l'état actuel des travaux scientifiques sur le sujet. Nous définissons la modularité, principale mesure globale de qualité de partitionnement, ainsi que les défauts qui lui sont associés-. Enfin, nous brossons un panorama des méthodes de détection de communautés non chevauchées, en terminant par le principal algorithme sur lequel s'appuie cette thèse, l'algorithme de Louvain.

Introduction

Certains réseaux formés par l'activité humaine ou observés dans la nature présentent des caractéristiques topologiques non triviales qui les distinguent fondamentalement de réseaux plus simples, en particulier ceux constitués aléatoirement par des modèles mathématiques. Ces réseaux, formés d'éléments en lien ou en interaction entre eux, sont généralement auto-organisés, si bien que chaque élément décide des liens ou interactions qu'il développe avec les autres membres.

Ces réseaux sont étudiés depuis les années 1920 dans trois principales disciplines : la sociologie, la biologie et les technologies de l'information. De par leurs propriétés communes, ils font l'objet d'études indépendamment de leur origine depuis une dizaine d'années, sous la dénomination de *réseaux complexes*. De manière informelle, leurs propriétés remarquables sont les suivantes :

- peu d'éléments, qui jouent un rôle important, sont fortement connectés alors qu'un grand nombre d'éléments ne le sont presque pas ;
- la densité globale du réseau est faible alors que certaines zones sont très denses, c'est-à-dire formées d'éléments très fortement connectés entre eux ;
- le réseau présente une structure communautaire, résultant de la deuxième propriété, et parfois hiérarchique, avec des groupes très denses formés de sous groupes.

L'étude de la structure communautaire a des applications dans toutes les disciplines d'origine des réseaux complexes. Elle permet de comprendre les interactions entre la structure topologique et la dynamique d'un réseau, en établissant par exemple pourquoi les liens se forment et comment ils évoluent. Elle permet aussi de représenter le réseau à différentes échelles.

Les méthodes algorithmiques de découverte de la structure communautaire sont très variées, en se fondant soit sur une approche locale telle que chaque élément n'a d'horizon que sa communauté d'appartenance, soit sur une approche globale où une fonction mesure la qualité de la partition en communautés, d'après la topologie du réseau. Parmi ces mesures, la modularité est certainement la plus utilisée. Elle se calcule facilement et donne de bons résultats, sous réserve des limitations que nous allons exposer.

Nous présentons dans ce chapitre les définitions générales sur les réseaux complexes et le problème de détection de communautés, puis traitons des mesures de qualité de partitionnement, dont la modularité, pour terminer par un exposé des algorithmes de partitionnement, d'abord très général puis détaillé pour les méthodes sur lesquelles s'appuie notre étude.

1.1 Réseaux complexes

1.1.1 Présentation générale et applications

D'abord étudiés sous la forme de réseaux, d'individus en sociologie, d'informations ou d'ordinateurs en technologie et de protéines en biologie, les réseaux d'interactions issus de la réalité émergent par généralisation à la fin des années 1990 avec les ouvrages de Barabási [1] et de Watts [2] qui consacrent une nouvelle discipline, au croisement d'autres disciplines plus anciennes, sous les dénominations de *Complex Networks* et de *New Science of Networks*. La traduction française de *Complex Networks* ne fait pas l'unanimité, allant de réseaux d'interaction à graphes de terrain en passant par la traduction littérale de "réseaux complexes". Cette dernière expression est certes vague, faisant aussi référence aux systèmes complexes liés à la théorie du chaos, mais nous la privilégions, car elle conserve l'idée générale de réseau, le graphe en

étant la modélisation mathématique. L'expression "graphes de terrain" évoque bien l'origine et la particularité de ces réseaux issus d'une réalité observée, mais perd en généralité.

L'émergence du concept général de réseaux complexes s'appuie historiquement sur deux modèles de réseaux clairement identifiés comme ayant des propriétés remarquables par rapport aux graphes aléatoires : les réseaux "petit-mondes" (*small-world nerworks*) et les réseaux sans échelles (*scale-free networks*). Nous présentons ces deux types de réseaux à travers leurs disciplines d'application.

Sociologie

Les premières études de réseaux complexes concernent la sociologie avec le travail de Jacob Moreno dans les années 1920 sur des petits groupes d'amis [3]. Les réseaux sociaux sont en général formés d'entités sociales ou d'individus liés entre eux. Il peut s'agir aussi bien de groupes de pouvoir, par exemple au sein des Medicis [4], d'entreprises [5] ou d'animaux comme les dauphins [6].

L'expérience de Milgram [7], décrite en 1967, a popularisé le concept de "six degrés de séparation" et a surtout mis en évidence l'effet "petit-monde", selon lequel deux personnes dans un réseau de ce type ne sont distantes que de six degrés au plus. En sautant de relation en relation, les deux personnes ne sont séparées que de six pas au maximum. Cette propriété a par la suite été définie de façon plus rigoureuse (voir la section 1.1.3). Elle est possédée par de nombreux réseaux aussi bien artificiels, comme certains réseaux aléatoires, que naturels. Un réseau petit-monde sera complexe et donc déviant par rapport aux graphes aléatoires, s'il possède aussi la propriété de transitivité ou de *clustering* décrite plus bas.

Information et technologie

Nous pouvons ici distinguer les réseaux d'informations, comme les réseaux de collaboration entre scientifiques [8], les réseaux sémantiques [9] ou le *World Wide Web* [10], et les réseaux technologiques, de transports [11] ou de routeurs [12] par exemple. Ils présentent généralement les caractéristiques des réseaux complexes dont celle de réseau sans échelles, vérifiée si la distribution des degrés de nœuds suit une loi particulière appelée loi de puissance (voir la définition exacte dans la section 1.1.3). Pour être précis, ce type de réseau est dénommé "sans échelles de distribution de degré"[1]. Le degré d'un élément du réseau, c'est-à-dire son nombre de liens avec les autres éléments, peut varier considérablement par rapport à la moyenne, de sorte qu'il y a de très nombreux éléments avec un degré faible et que le nombre d'éléments du réseaux diminue fortement d'autant plus que le degré augmente.

Biologie

Les biologistes rencontrent des réseaux métaboliques, modélisant les processus de génération et de dégradation des matériaux et de l'énergie au sein d'organismes vivants [13], des réseaux d'interaction entre protéines [14] ou encore des réseaux de régulation génétique [15]. Nous pouvons citer également les réseaux de neurones et les réseaux alimentaires. Ces réseaux exhibent des propriétés de réseaux complexes, en particulier la distribution sans échelles, d'après les études topologiques dont ils font l'objet (voir [16, 17] pour avoir un large aperçu de ces études).

[1] La loi de puissance est la seule fonction mathématique invariante au changement d'échelle, d'où l'expression de réseau sans échelles

Autres domaines

Les réseaux complexes intéressent également les physiciens statisticiens. D'une part en raison des connexions avec l'étude des systèmes complexes, la topologie et la dynamique du réseau ayant des déterminants locaux alors qu'il en résulte des comportements macroscopiques difficiles à prévoir et à modéliser et que les caractéristiques statistiques globales qui en émergent sont celles des réseaux complexes. D'autre part, parce que ces réseaux apparaissent en physiques, pas directement comme objet d'étude issu d'une réalité physique, mais plutôt dans des analyses de systèmes dynamiques comme les oscillateurs couplés [18], les interactions ferromagnétiques entre spins [19], les surfaces d'énergie potentielle [20], et bien d'autres (voir les articles de synthèse sur les réseaux complexes de Marc Newman [21, 22], d'Albert-László Barabási [1], d'Annick Lesne [23] et d'Alain Barrat [24]).

Notons enfin que les réseaux de transports et les réseaux sociaux permettent, à travers l'étude de leur dynamique, la simulation de la propagation d'une épidémie [25].

1.1.2 Modélisation par les graphes

La théorie des graphes fournit un support de modélisation des réseaux complexes en généralisant leur structure quelque soit leur origine :

- un élément constitutif du réseau (individu, ordinateur, protéines,...) est représenté par un sommet ou nœud de graphe ;
- une relation ou un lien entre deux éléments est représenté par une arête ou un arc du graphe.

Cette modélisation permet d'exprimer les propriétés distinctives des réseaux complexes, et d'y appliquer des algorithmes pour résoudre les problèmes que ceux-ci soulèvent.

Définitions générales

Un **graphe** non orienté $G = (V, E)$ est composé d'un ensemble de **nœuds** V et d'un ensemble d'**arêtes** E, une arête étant un couple (u, v) de nœuds désignant ces extrémités. Pour un graphe pondéré, la notation devient $G = (V, E, w)$ en ajoutant une fonction de poids $w : E \longmapsto \mathbb{R}$ telle que $w((u, v))$ désigne le poids de l'arête (u, v). L'arête (u, v) est dite incidente à ses extrémités u et v.

L'**ordre** du graphe est son nombre de nœuds, noté $n = |V|$. Le nombre d'arêtes est noté $m = |E|$. Pour une arête (u, v), on dit que les nœuds u et v sont adjacents ou voisins. L'ensemble des nœuds adjacents à u est noté $\Gamma(u)$. Le **degré** d'un nœud est la cardinal de cet ensemble, soit $d(u) = |\Gamma(u)|$. Dans un graphe pondéré, le degré pondéré du nœud u est $d_w(u) = \sum_{v \in \Gamma(u)} w(u, v)$, c'est-à-dire la somme des poids des arêtes incidentes.

Une **boucle** est une arête (u, u) ayant les mêmes extrémités. En général, il est possible qu'un graphe contienne plusieurs arêtes entre deux nœuds donnés. Un graphe est dit **simple** si, pour deux nœuds quelconques, il n'existe qu'une arête au plus les reliant et si le graphe ne contient aucune boucle. Dans la suite de notre exposé, sauf exception mentionnée, nous ne considérons que les graphes simples. Sous cette condition, la fonction de poids d'un graphe pondéré peut se définir simplement par $w : V \times V \longmapsto \mathbb{R}$ telle que $w(u, v)$ désigne le poids réel non nul associé à l'arête (u, v) ou vaut zéro en l'absence d'arête entre les nœuds u et v. Le graphe est complètement défini par (V, w) mais nous continuerons à le noter (V, E, w), car l'ensemble des arêtes est utile dans certaines formules.

Sous-graphes et cliques

À partir d'un sous-ensemble de nœuds $V' \subset V$, le **sous-graphe** induit par V' est le graphe (V', E') où $E' = \{(u, v) \in E, u \in V' \text{ et } v \in V'\}$. Le sous-graphe ne contient donc que les arêtes ayant leurs extrémités dans le sous-ensemble de nœuds considérés.

Un graphe est dit **complet** si pour toute paire de nœuds, il existe une et une seule arête les reliant. Dans ce cas, l'ordre du graphe est $n(n-1)$. Un sous-graphe complet d'un graphe donné est appelé une **clique** de ce graphe. La **densité** d'un graphe est définie comme $\frac{2m}{n(n-1)}$ soit le rapport entre le nombre d'arêtes et le nombre maximum d'arêtes possibles compte tenu du nombre de nœuds du graphe.

Chemin et diamètre

Le **chemin** entre les nœuds u et v est une séquence $\langle v_0, \ldots, v_k \rangle$ de nœuds telle que $v_0 = u, v_k = v$ et $\forall i, 0 \leq i \leq k-1, (v_i, v_{i+1}) \in E$. C'est une succession d'arêtes qui permettent de naviguer d'un nœud à un autre. La **distance** entre deux nœuds est la longueur du plus court chemin les reliant, c'est-à-dire son nombre d'arêtes. Le **diamètre** d'un graphe est la plus grande distance qui puisse exister entre deux de ses nœuds. Enfin, la **longueur moyenne de chemin** est la moyenne des longueurs du plus court chemin entre toutes les paires possibles de nœuds.

Dans la suite de l'exposé, nous ne considérons que les graphes simples non orientés, éventuellement pondérés. Ainsi une arête peut être désignée par un ensemble de deux nœuds $\{u, v\}$ puisqu'il n'y a pas de sens entre u et v et que $u \neq v$.

1.1.3 Définition d'un réseau complexe par ses propriétés

Propriétés des réseaux en général

Pour caractériser les réseaux complexes, nous nous intéressons à trois propriétés essentielles. Tout d'abord le nombre de triangles, dont l'importance est relative à ce que l'on s'attend à trouver dans un graphe aléatoire. Cela traduit le fait que si les nœuds u et u' sont reliés, de même que u' et u", alors il est très probable que u et u" le soient aussi. Cette propriété est mesurée par le coefficient de *clustering* global défini par le rapport du nombre de triplets fermés (formant un triangle) et du nombre total de triplets [26].

Ensuite, la longueur moyenne de chemin permet de caractériser le phénomène de petit-monde. Elle est faible dans un réseau complexe, toujours relativement à un graphe aléatoire.

Enfin, la distribution des degrés de nœuds qui, typiquement dans un modèle de réseau sans échelles, suit une loi de puissance. Cette distribution est définie par $P(\delta) = \frac{|\{v \in V, d(v) = \delta\}|}{n}$ qui désigne la proportion de nœuds ayant un degré δ. En suivant une loi de puissance, cette distribution s'écrit $P(\delta) \sim \delta^{-\gamma}$ avec $\gamma > 2$. Dans cette distribution, il y a un grand nombre de nœuds de degré faible (d'autant plus que δ est grand) et très peu de nœuds ayant un degré élevé.

Ces propriétés, réunies dans un même graphe, engendrent d'autres propriétés dont une nous intéresse particulièrement. La première est la résilience du réseau qui fait que la navigation à travers le réseau, d'un nœud quelconque à un autre, a une probabilité très faible d'être affectée si un nœud est supprimé. La deuxième est la faible densité du graphe, qui croit linéairement avec son ordre, qui tend donc vers 0 lorsque celui-ci tend vers l'infini. Enfin la troisième est l'existence de zones de densité très forte, à différentes échelles. C'est précisément la propriété de structure communautaire, qui fait l'objet de notre étude.

Modèles de graphe

Un réseau complexe a des caractéristiques topologiques qui le distinguent clairement d'un graphe parfaitement régulier, mais aussi d'un graphe parfaitement aléatoire de même taille.

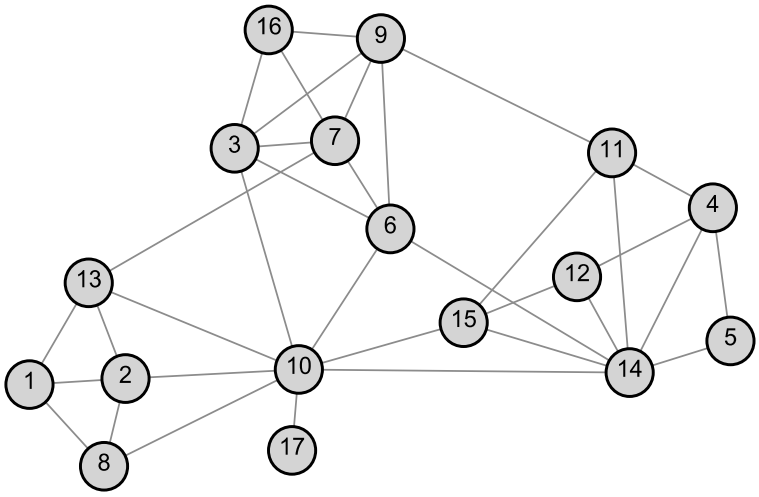


FIGURE 1.1 – Le graphe suivant contient 17 nœuds et 35 arêtes. Il se distingue essentiellement d'un graphe aléatoire par le coefficient de clustering valant 0.559 (moins de 0.4 dans un graphe aléatoire de même degré et même nombre d'arêtes). Sa longueur moyenne de chemin est de 2.074. Il est impossible d'estimer la distribution des degrés de nœuds avec un graphe si petit, mais la présence de nœuds centraux de fort degré est manifeste (n° 10 et 14). Enfin la modularité présentée plus loin vaut ici 0.467 en identifiant trois communautés, alors qu'elle ne dépasse par 0.3 dans les graphes aléatoires.

Les modèles de graphes utiles, par comparaison, à l'étude des réseaux complexes sont :

- le modèle de graphe aléatoire de Erdös-Rényi (ER) [27] dont la distribution de degré suit une loi de Poisson ;
- le modèle petit-monde présenté par Watts et Strogatz [26] qui, en jouant sur un paramètre de recâblage de nœuds, donne des graphes ayant une longueur moyenne de chemin faible (le diamètre du graphe croit au plus comme le logarithme de la taille du graphe) et un fort coefficient de clustering (tendant vers $\frac{3}{4}$ lorsque la taille du graphe croit infiniment) ; cependant, la distribution de degré suit une loi binomiale ;
- le modèle de réseau sans échelles de Barabasi-Albert qui exhibe une distribution de degré de nœuds en loi de puissance, vérifie la propriété de longueur moyenne de chemin faible, mais ne se différencie pas d'un graphe aléatoire par le coefficient de clustering.

Définition d'un réseau complexe

Ces modèles nous permettent de définir plus précisément un réseau complexe qui possède à la fois la propriété de clustering, la distribution en loi de puissance (de façon approchée bien sûr, les données étant partielles et issues de la réalité) et la longueur moyenne de chemin faible, qui n'évolue pas plus vite que le logarithme de l'ordre du graphe (voir une illustration en figure 1.1). Il en résulte une propriété remarquable, non observée dans les graphes aléatoires, d'existence de groupes de forte densité, que nous allons définir. Il peut en résulter aussi une structure hiérarchique dans laquelle des groupes très denses sont eux-mêmes composés de sous-groupes.

1.2 Problème de la détection de communautés

Les communautés dans un réseau complexe ne sont pas définies de façon unanime, mais l'idée de connexions internes fortes et de connexions externes faibles est partagée par tous. Les applications de ce concept sont variées selon les disciplines d'origine des graphes. Là encore, le cadre général des réseaux complexes, fourni par la théorie des graphes, permet d'identifier des communautés et de développer des méthodes de partition en communautés. Il incombe aux spécialistes du domaine d'étude d'interpréter et d'exploiter ce découpage.

À partir des définitions locales qui concernent les nœuds ou les communautés et leurs voisins, des mesures globales sont définies pour évaluer la pertinence d'un découpage sur une échelle de valeurs. La plus connue et utilisée est la modularité sur laquelle s'appuie notre étude.

1.2.1 Concept de communauté dans un réseau complexe

Une communauté, que nous appelons aussi groupe ou module, est définie de façon la plus universelle et minimaliste comme un sous-ensemble de nœuds possédant plus d'arêtes internes, c'est-à-dire entre nœuds du groupe que d'arêtes externes, c'est-à-dire avec une extrémité dans le groupe et l'autre en dehors.

Radicchi et al. en 2004 [28] ont défini une condition forte et une condition faible, proches mais un peu plus strictes que la définition intuitive précédente. Le degré interne d'un nœud v par rapport à sa communauté C, noté $d_{in}(v)$ est défini comme le nombre d'arêtes incidentes à v ayant leurs deux extrémités dans C. Par opposition, le degré externe $d_{ext}(v)$ est le nombre d'arêtes incidentes à v ayant une extrémité en dehors de C. Pour un graphe pondéré, ces degrés s'entendent comme la somme des poids des arêtes concernées. Par extension, le degré interne $d_{in}(C)$ d'une communauté C est la somme des degrés internes de ses nœuds, ce qui revient au double du nombre d'arêtes internes, chacune étant comptée deux fois pour chacun des deux nœuds extrémités. Le degré externe $d_{out}(C)$ d'une communauté C est la somme des degrés externes de ses nœuds. À partir de ces définitions, une communauté C est :

- forte si $\forall v \in C, d_{in}(v) > d_{out}(v)$;
- faible si $\sum_{v \in C} d_{in}(v) > \sum_{v \in C} d_{out}(v)$, soit si $d_{in}(C) > d_{out}(C)$.

Hu et al. [29] ont donné une définition encore plus faible des communautés, que nous utiliserons par la suite. Une communauté, au sens le plus faible, est telle que son degré interne est supérieur au nombre d'arêtes partagées par celle-ci avec *n'importe quelle* autre communauté voisine, au lieu du nombre d'arêtes partagées avec *l'ensemble* des communautés, au sens faible de Radicchi (voir la figure 1.2 pour une illustration des trois définitions).

D'autres définitions existent, comme celle faisant intervenir la densité interne d'une communauté comparée à sa densité externe définie comme le rapport du degré externe et du nombre total d'arêtes possibles avec une extrémité dans la communauté et l'autre en dehors. Dans la plupart des définitions, l'idée directrice est l'opposition entre une cohésion interne forte, selon l'idée de communauté au sens social, et une connexion avec l'extérieur de la communauté relativement faible. Pourtant, ces définitions sont insuffisantes pour définir et qualifier une façon de partitionner un graphe en plusieurs communautés, ne serait-ce qu'en raison d'une partition triviale constituée d'une seule communauté englobant tous les nœuds. Les méthodes qui résolvent le problème de détection de communautés n'utilisent pas directement ces définitions, si ce n'est pour vérifier leurs résultats, mais emploient plutôt des techniques qui capturent ces définitions et permettent de décider si une partition en communautés donnée est meilleure qu'une autre.

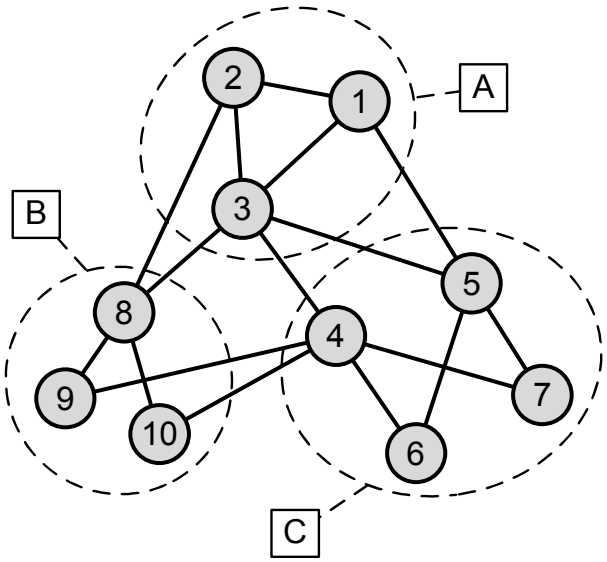


FIGURE 1.2 – Dans cette partition en trois communautés A, B et C, la communauté A n'est pas définie fortement à cause du nœud 3 dont le degré externe valant 3 est supérieur au degré interne valant 2. En revanche, elle est faiblement définie, car $d_{in}(1) + d_{in}(2) + d_{in}(3) = 2 + 2 + 2 = 6$ est supérieur au degré externe de communauté $d_{out}(1) + d_{out}(2) + d_{out}(3) = 1 + 1 + 3 = 5$. La communauté B n'est définie que très faiblement, car son degré interne valant 4 est supérieur au degré externe entre B et C (2) et au degré externe entre B et A (2).

1.2.2 Présentation et définition du problème

Applications

La détection de communautés dans un réseau complexe recouvre deux réalités selon qu'un nœud du réseau peut ou non appartenir à plusieurs communautés. Ce problème se pose de différentes manières selon les disciplines. Les sociologues voient la communauté dans son acception générale de groupe d'individus partageant des valeurs, une culture, des comportements communs ou tout simplement des affinités. Dans ce contexte, il est fréquent que les communautés se chevauchent, c'est-à-dire qu'un individu se revendique de plusieurs groupes. Dans une autre application sociale, un hôpital par exemple, les communautés sont de différentes natures et permettent d'étudier la propagation des bactéries au sein de l'hôpital. Dans les réseaux d'interaction entre protéines, celles-ci interagissant fortement entre elles au sein d'un module (un groupe de protéines) ont des fonctions similaires voir identiques dans les cellules de l'organisme étudié et participent ensemble à un même processus biologique.

Les communautés permettent ainsi aux chercheurs de schématiser les réseaux qu'ils étudient en identifiant des sous-parties homogènes. De la même manière, le découpage en communautés favorise la visualisation d'un graphe puisqu'une communauté peut être perçue comme un constituant "boîte noire" du réseau, avec la possibilité de plonger à l'intérieur pour visualiser et étudier ses composants, des nœuds simples ou des sous-communautés si le réseau est hiérarchique. Ce découpage permet également, comme pour le problème du partitionnement de graphe, d'obtenir un graphe grossier, une communauté étant vue comme un nœud, qui peut être traité plus facilement. Un traitement à appliquer à chaque communauté peut aussi être parallélisé grâce à ce découpage. Le principe général est d'offrir une ou plusieurs strates intermédiaires entre le niveau microscopique des nœuds et le niveau macroscopique du réseau tout entier. Les comportements qui émergent au niveau macroscopique sont rattachés aux communautés pour être étudiés.

Partitionnement d'un graphe en communautés

Dans l'hypothèse où un nœud appartient au plus à une communauté, le problème consiste à définir une partition de l'ensemble des nœuds, au sens mathématique, soit qui satisfait un critère de validation des communautés, soit qui maximise une fonction d'évaluation du partitionnement. Le partitionnement en communauté d'un graphe $G = (V, E)$ est une partition $\{C_1, C_2, ..., C_k\}$ telle que :

1. $\forall i \in \{1, 2, ..., k\}, C_i \subset V$ et $C_i \neq \varnothing$;
2. $\cup_{i=1}^{k} C_i = V$;
3. $\forall i, j \in \{1, 2, ..., k\}, C_i \cap C_j = \varnothing$.

Chaque sous-ensemble disjoint C_i est une communauté dans cette partition. Par ailleurs, nous notons pour la suite $\mu(v)$ la communauté d'appartenance du nœud v, ou $\mu^{\mathcal{P}}(v)$ s'il est besoin de préciser la partition de référence $\mathcal{P}$. Le degré interne de communauté $d_{in}(C)$ et le degré externe de communauté $d_{out}(C)$ ont été définis plus haut. Le degré de communauté $d(C)$ est défini comme la somme des degrés des nœuds la composant, soit $d(C) = \sum_{v \in C} d(v)$ soit encore $d(C) = d_{out}(C) + d_{in}(C)$.

Définition locale Au niveau des communautés, un critère de décision Γ valide chaque communauté : $\forall i \in \{1, 2, ..., k\}, \Gamma(C_i)$. Cette condition locale peut guider un algorithme de partitionnement, mais peut aussi servir à la validation d'une partition fournie par un algorithme quelconque. À titre d'exemple, un seuil minimal de densité interne de communauté peut constituer un tel critère. La densité interne d'une communauté C, notée $\delta_{int}(C)$, est la densité du sous-graphe C soit $\frac{d_{in}C}{|C|(|C|-1)}$. La condition est $\Gamma(C) \equiv \delta_{int}(C) \geq \xi$, en notant ξ le seuil compris entre 0 et 1. Dans le cas particulier où $\xi = 1$, C est une clique.

Définition globale Une fonction de qualité de partitionnement Q associe à toute partition $\mathcal{P}$ un nombre réel $Q(\mathcal{P})$ qui permet de déterminer entre deux partitions quelconques, laquelle est la meilleure selon ce critère. Le problème de partitionnement en communautés devient alors un problème d'optimisation. Les mesures de qualité les plus courantes sont présentées dans la section suivante.

Détection de communautés chevauchées

Dans sa forme la plus générale et difficile à résoudre, lorsqu'un nœud peut appartenir à plusieurs communautés, le problème consiste à trouver un recouvrement de communautés c'est-à-dire un ensemble $\{C_1, C_2, ..., C_k\}$ de parties de V tel que :

1. $\forall i \in \{1, 2, ..., k\}, C_i \subset V$ et $C_i \neq \varnothing$;
2. $\cup_{i=1}^{k} C_i = V$.

Il est également possible de pondérer l'appartenance de chaque nœud aux communautés par un nombre. Le découpage en communautés chevauchées, noté $\mathcal{C} = \{C_1, C_2, ..., C_k\}$, est alors complètement défini par la fonction d'appartenance $\phi : V \times \mathcal{C} \longmapsto \mathbb{R}$ telle que :

1. $\forall v \in V$ et $C \in \mathcal{C}, 0 \leq \phi(v, C) \leq 1$;
2. $\phi(v, C) = 0$ signifie que le nœud v n'appartient pas à la communauté C ;
3. $\forall v \in V, \sum_{C \in \mathcal{C}} \phi(v, C) = 1$.

Notre étude, y compris l'état de l'art des méthodes de résolution, se limite au partitionnement en communautés non chevauchées.

1.2.3 Mesures de qualité d'une partition

Dans une première approche, la mesure de qualité d'une partition permet de comparer deux solutions fournies par un algorithme de détection de communautés et de décider laquelle de deux est la plus pertinente. L'idée d'utiliser cette mesure en amont dans l'algorithme de partitionnement comme objectif d'optimisation vient naturellement, puisqu'on recherche précisément la partition qui donne le meilleur score. Le problème se transforme en problème d'optimisation, la fonction de mesure devant être minimisée ou maximisée selon sa formulation.

Il existe bon nombre de mesures de qualité, la plupart étant additives au sens où, en notant $Q(\mathcal{P})$ la mesure de qualité de la partition $\mathcal{P}$, nous avons $Q(\mathcal{P}) = \sum_{C \in \mathcal{P}} q(C)$ à partir d'une fonction q évaluant une communauté. L'additivité, très pratique pour les algorithmes, est en général la cause de l'un des défauts de cette approche, comme nous le verrons dans la section suivante. Nous pouvons citer, comme exemple de mesure de qualité, la performance ou la "couverture" (voir l'article de synthèse de Fortunato [30] pour les deux) ou encore le facteur de mérite [31], l'influence globale basée sur la modularité [32], la mesure de "surprise" (improbabilité) de trouver une structure communautaire identique dans un graphe généré [33]. Cependant, la plus connue, étudiée et utilisée est sans conteste la modularité.

Modularité

L'idée générale de la modularité, introduite par Newman en 2004 [34] est de gratifier la présence d'arêtes dans les communautés et de pénaliser l'existence d'arêtes entre communautés, non dans l'absolue comme le prétend la définition d'une communauté, mais par rapport à la situation des arêtes dans un graphe aléatoire possédant la même distribution de degrés. Pour une communauté C, la modularité $q(C)$ est la différence entre la part des arêtes effectivement dans la communauté, notée $e(C)$, et la part que l'on s'attend à trouver dans un graphe aléatoire, pourvu des mêmes nœuds avec les mêmes degrés, notée $a(C)$. En notant $l(C_i)$ le nombre d'arêtes internes à la communauté C_i (arêtes ayant leurs deux extrémités dans la communauté), le premier terme s'écrit $e(C_i) = \frac{l(C_i)}{m}$. Le deuxième terme $a(C)$ s'interprète comme la probabilité qu'une arête choisie au hasard dans le graphe aléatoire correspondant soit dans C_i. Une telle arête doit s'arrimer à un des nœuds de C_i, soit $\sum_{v \in C_i} d(v) = d(C_i)$ possibilités, sur un total d'arrimages possibles de $2m$, la somme des degrés de tous les nœuds. Par son autre extrémité, elle doit aussi s'arrimer à un noeud de C_i. L'arrimage des deux extrémités étant indépendants, la part d'arêtes recherchée est $a(C_i) = \frac{d(C_i)}{2m}\frac{d(C_i)}{2m}$. Ainsi, la modularité de communauté est :

$$q(C_i) = \frac{l(C_i)}{m} - \left(\frac{d(C_i)}{2m}\right)^2 \tag{1.1}$$

La modularité $Q(\mathcal{P})$ est la somme des modularités de communauté, soit :

$$Q(\mathcal{P}) = \sum_{C \in \mathcal{P}} \left(\frac{l(C)}{m} - \left(\frac{d(C)}{2m}\right)^2\right) \tag{1.2}$$

Dans un graphe pondéré, la formule devient :

$$Q(\mathcal{P}) = \sum_{C \in \mathcal{P}} \left(\frac{d_{in}(C)}{d(V)} - \left(\frac{d(C)}{d(V)}\right)^2\right) \tag{1.3}$$

La notation $d(C)$, vue plus haut, désigne la somme des degrés pondérés des nœuds de C. Comme cas particulier, $d(V)$ est la somme des degrés pondérés de tous les nœuds du graphe. A noter que dans un graphe non pondéré, $d(V) = 2m$.

Dans le cas d'une partition ayant une seule communauté C couvrant la totalité du graphe, $l(C) = m$ et $d(C) = 2m$, soit une modularité nulle. Dans le meilleur des cas, la partition est formée de k-cliques non connectées entre elles. Ceci implique que $l(C) = k(k-1)/2$ et $d(C) = k(k-1)$ pour tout C, soit une modularité égale à $1/m - (1/m)^2$. A l'extrême, si m est nul, la modularité atteint la valeur maximale de 1. Dans le cas le plus défavorable de la partition singleton, avec un seul nœud par communauté, $l(C)$=0 pour toute communauté C, ce qui conduit à une modularité négative. Généralement, cette modularité est tout de même proche de zéro si les $d(C)$ sont faibles par rapport à un grand m. Au pire avec un graphe de deux nœuds reliés par une arête ($m = 1$), chaque nœud étant dans une communauté, la modularité est minimale et vaut $-1/2$.

Il existe une forme plus générale de la modularité exprimée au niveau des arêtes, qui se déduit facilement de l'écriture précédente :

$$\begin{aligned} q(\mathcal{P}) &= \frac{1}{2m} \sum_{u,v \in V \text{ et } \mu(u)=\mu(v)} \left(A_{u,v} - \frac{d(u)d(v)}{2m} \right) \\ &= \sum_{u,v \in V \text{ et } \mu(u)=\mu(v)} \left(\frac{A_{u,v}}{2m} - p(u,v) \right) \end{aligned} \tag{1.4}$$

La matrice A est la matrice d'adjacence du graphe de sorte que $A_{u,v} = 1$ si $\{u, v\} \in E$ et $A_{u,v} = 0$ sinon. La condition $\mu(u) = \mu(v)$ exprime le fait que les nœuds u et v sont dans la même communauté dans la partition $\mathcal{P}$.

Dans la deuxième écriture, $p(u, v)$ désigne la probabilité que l'arête $\{u, v\}$ existe dans le graphe aléatoire de comparaison. Dans cette forme généralisée, le graphe aléatoire est nommé "modèle nul" et reprend certaines caractéristiques du graphe étudié sans la structure communautaire. Il existe plusieurs types de modèle nul, selon les caractéristiques que l'on souhaite préserver dans le graphe de comparaison. Le modèle employé dans la modularité est proche du modèle de configuration, introduit et amélioré par plusieurs auteurs (voir Molloy et Reed [35] qui ont établi une procédure de construction sur ce modèle). Ce modèle s'attache à préserver la distribution de degré de sorte que $p(u, v) = \frac{d(u)d(v)}{2m}$.

1.2.4 Défauts de l'optimisation d'une mesure globale

La première difficulté mise en évidence avec la modularité est son interprétation dans l'absolu pour déterminer si un graphe a une structure modulaire ou non. En effet, la valeur de modularité est significativement supérieure à zéro pour certains graphes aléatoires [36], comme les graphes ER [37, 38]. Cette particularité peut obliger les utilisateurs d'algorithmes de détection de communautés à valider les résultats de ceux-ci par des méthodes statistiques pour écarter les phénomènes qui relèvent du hasard. Dans la pratique, de nombreux auteurs considèrent qu'une modularité supérieure ou égale à 0.3 est significative d'un réseau complexe. Par la suite, deux problèmes plus graves ont été mis au jour, la limite de résolution, qui fait "disparaitre" les très petites communautés et la dégénérescence, qui rend délicate l'interprétation d'une partition dont la modularité n'est pas suffisamment maximisée.

Limite de résolution

Fortunato et Barthemely[39] ont établi que la modularité souffre d'une limite de résolution qui conduit, en optimisant cette fonction et dans certaines conditions, à faire disparaître les petites communautés qui sont regroupées avec d'autres communautés de tailles similaires ou absorbées par des communautés relativement beaucoup plus grandes. Pour être précis, leur démonstration porte sur des cas extrêmes de cliques connectées entre elles de façon minimale, par exemple

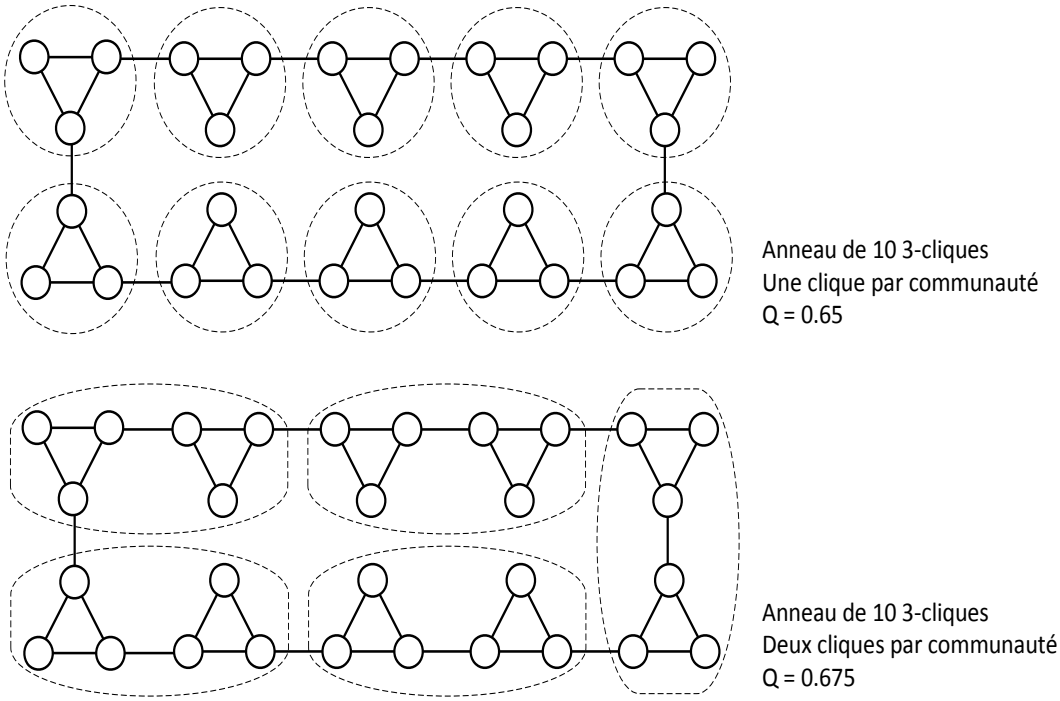


FIGURE 1.3 – Illustration de la limite de résolution. À partir d'un anneau de dix 3-cliques reliées deux à deux par un seul lien, la partition naturelle associe une communauté à chaque clique avec une modularité d'environ 0.65. La modularité peut être améliorée en regroupant les cliques par deux, ce qui pourtant constitue manifestement une partition erronée.

un anneau de cliques reliées par paires par une seule arête (voir figure 1.3). Dans ce cas, si le graphe est suffisamment grand, les cliques de taille inférieure ou égale à $\sqrt{m}$ sont fusionnées avec d'autres cliques alors qu'elles devraient être identifiées comme des communautés. Ce phénomène est aussi observé dans des situations plus réalistes, par exemple avec des graphes et des partitions générés aléatoirement [40, 41].

Des remèdes variés ont été proposés, comme celui de Fortunato lui-même, qui consiste à réexaminer les grandes communautés comme des graphes autonomes pour vérifier si elles ne contiennent pas des sous-communautés non détectées. Cette solution est discutable, car elle fait l'impasse sur les liens entre communautés lorsqu'une communauté est prise comme un graphe à part entière. Berry et al. ont proposé une transformation des poids des arêtes qui réduit les effets de la limite de résolution [42]. D'autres auteurs se sont orientés vers la multirésolution que nous abordons dans la section 1.3. Il est possible également d'utiliser d'autres fonctions que la modularité, comme celles que nous avons citées plus haut, ou comme la modularité-densité [43] qui souffre tout de même d'une limite de résolution atténuée. Fortunato conjecture que la plupart des fonctions de qualité de partitionnement, du fait de leur caractère additif, présentent une forme de limite de résolution, car l'optimisation traduit un compromis entre un grand nombre de petites communautés qui peut avoir une qualité légèrement inférieure à un petit nombre de grandes communautés.

Inconsistance de solution

Good et al. [44] ont mis en évidence un phénomène de "dégénérescence" de la fonction de modularité qui se traduit par un nombre très important de valeurs sous-optimales, en croissance exponentielle avec la taille du graphe. Ces optimums locaux forment un plateau de valeurs très proches de l'optimum global, correspondant à des partitions structurellement assez différentes

entre-elles et différentes de la partition optimale. Ce phénomène peut être mesuré expérimentalement, comme nous le ferons dans la suite, en calculant une distance structurelle entre partitions. Cela se traduit par une forme d'inconsistance des solutions proposées par un algorithme de détection de communauté optimisant la modularité. Le problème est d'autant plus accru que l'algorithme utilisé ne parvient pas à optimiser efficacement la modularité.

1.3 Méthodes de résolution hors optimisation globale

Nous allons brosser un panorama des principaux types de méthodes de détection de communautés qui ne s'appuient pas sur l'optimisation d'une fonction globale. Le principe général de ces méthodes est abordé sans entrer dans les détails, en renvoyant aux auteurs qui les ont développées et testées.

1.3.1 Méthodes et problèmes voisins

Historiquement, au moment de sa formulation explicite à la fin des années 1990, le problème de la détection de communautés a été naturellement rapproché de problèmes voisins déjà bien connus comme celui du partitionnement de graphe. La spécificité de la détection de communautés dans les réseaux complexes est la méconnaissance *a priori* du nombre de communautés et donc de leurs tailles. Pour cette raison, les rapprochements avec les problèmes voisins décrits ci-dessous ne permettent pas une utilisation directe des méthodes de résolution de ces problèmes, mais peuvent offrir des techniques transposables à la détection de communautés.

Partitionnement de graphe

Le problème du partitionnement de graphe consiste à partager l'ensemble des nœuds d'un graphe en k groupes, k étant préalablement fixé, de manière à minimiser le nombre d'arêtes entre les groupes. La taille des groupes est également contrainte pour éviter une partition triviale sans intérêt. Ce problème apparaît par exemple dans la conception de circuits imprimés et l'ordonnancement de tâches exécutées sur plusieurs processeurs. Une des méthodes locales parmi les plus connues pour le résoudre est l'algorithme de Kernighan-Lin [45] qui, à partir de deux groupes de même taille, échange des nœuds entre eux pour optimiser une fonction de coût. Il existe aussi une méthode multiniveaux qui permet de traiter des graphes de grandes tailles, dont les principes sont repris pour la détection de communautés (voir la section 1.4.3). Citons enfin la bissection spectrale [46] qui cherche à établir une coupe, minimale en nombre d'arêtes, qui sépare un graphe en deux groupes égaux en taille. Cette coupe peut s'écrire en utilisant la matrice Laplacienne, définie comme la différence entre la matrice de degré et la matrice d'adjacence, ce qui transforme le problème en recherche de vecteurs propres d'une matrice. Ces méthodes ne sont pas directement exploitables pour la détection de communautés, mais elles fournissent des principes s'appliquant à ce problème, par exemple le déplacement de nœud en optimisant une fonction avec l'algorithme de Kernighan-Lin ou la coupe minimale et la matrice Laplacienne avec la bissection spectrale.

Analyse de cluster

Un *cluster* est un regroupement de nœuds d'un réseau présentant des similarités, selon une grande variété de définitions possibles faisant appel en général à une mesure de distance. La notion est imprécise, car elle apparaît sous différente forme dans le vaste domaine de l'analyse de

cluster, nommée *clustering*[2] et utile à l'analyse et au forage de données (*data mining*). Parmi les modèles d'algorithme de clustering, nous pouvons citer le clustering de partition et le clustering hiérarchique, le seul fournissant des méthodes transposables pour la détection de communautés.

Clustering de partition Dans ce modèle d'analyse de cluster, les clusters sont formés de points entre lesquels une distance est définie. L'objectif est de former des groupes de k points (k est fixé à priori) proches entre eux ou proches d'un point centre de cluster, par la mesure de distance. Le plus connu des algorithmes est *k-means clustering* [47] mais, encore une fois, l'obligation de connaître au préalable le nombre de groupes ne rend pas les méthodes de résolution de ce problème transposables aux réseaux complexes.

Clustering hiérarchique Dans ce modèle, les groupes sont formés de sous-groupes, l'ensemble constituant une vision hiérarchique d'un graphe que l'on peut représenter par un dendrogramme (figure 1.4). Une mesure de similarité entre nœuds est définie puis calculée pour chaque paire de noeuds, qu'ils soient connectés ou non. Ensuite, un algorithme exploite cette matrice pour construire la hiérarchie selon une des deux techniques suivantes :

- agglomération : deux groupes sont fusionnés si leur similarité est suffisamment haute ;
- division : un groupe est scindé en deux en supprimant les arêtes entre nœuds de faible similarité.

Ces techniques déterminent deux classes d'algorithmes, les uns partant d'une partition où chaque nœud est seul dans une communauté (méthodes par agglomération), les autres partant du graphe formé d'un seul groupe (méthodes par division). La similarité de nœud peut être une distance dans un espace euclidien, une mesure fondée sur le nombre de nœuds voisins en commun ou encore le nombre de chemins entre deux nœuds. Une marche aléatoire peut aussi être utilisée (voir section 1.3.3). Les méthodes de clustering hiérarchique peuvent être adaptées à la détection de communautés à condition de trouver une mesure de similarité pertinente. Cependant, plusieurs difficultés ont été observées rendant délicate l'utilisation de ces méthodes dans tous les cas de réseaux complexes. Tout d'abord, des nœuds peuvent être mal placés dans un groupe, certains en dehors du groupe dans lequel ils jouent un rôle central, d'autres connectés à un seul voisin se retrouvent seuls dans une communauté. Ensuite, la structure hiérarchique n'est pas nécessairement pertinente si le graphe ne possède pas cette structure intrinsèquement. Enfin, les méthodes fondées sur une distance ne permettent pas le traitement de très grands réseaux complexes du fait de leur complexité en temps qui peut atteindre $O(n^2 \log(n))$.

Analyse spectrale L'analyse spectrale trouve son utilité dans l'analyse de cluster en faisant intervenir la matrice de similarité (voir le tutoriel de Von Luxburg [48]) qui représente les distances entre chaque paire de nœuds. Ici aussi, la difficulté liée à la complexité introduite par le calcul des distances est rédhibitoire pour les grands réseaux complexes. L'algorithme le plus connu utilisant l'analyse spectrale pour la détection de communautés est sans doute celui de Donetti et Muñoz [49].

1.3.2 Méthodes séparatives

Les méthodes séparatives s'inspirent du clustering hiérarchique, mais sans utiliser de mesure de similarité entre nœuds. Les arêtes inter-communautés sont supprimées pour faire apparaître

[2] Nous préservons ces termes anglais de *cluster* et *clustering* par opposition au simple partitionnement (*partitioning*)

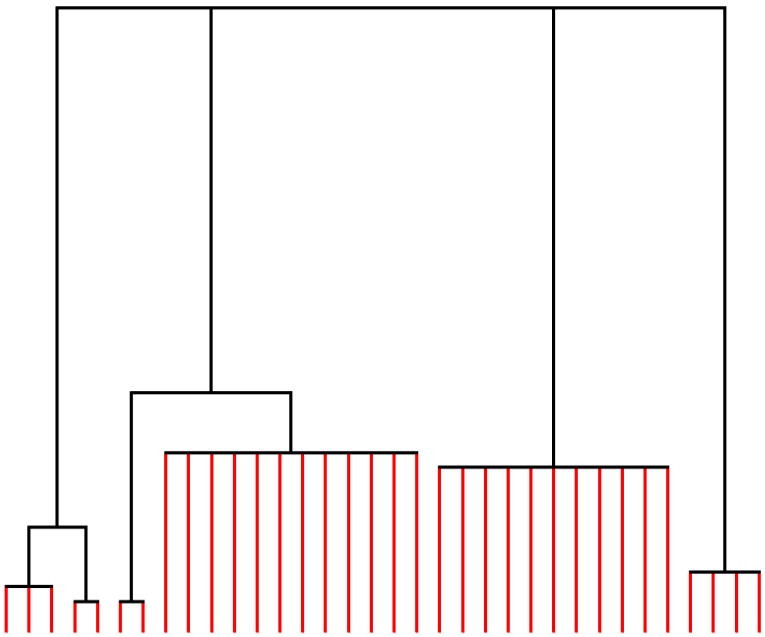


FIGURE 1.4 – Exemple de représentation d'un clustering hierarchique sous la forme d'un dendrogramme. Chaque fourche représente un groupe constitué de sous-groupes ou de nœuds du graphe.

les communautés, lorsqu'ils ne restent que les arêtes intra-communautaires. La difficulté est de définir un critère de distinction entre ces deux types d'arêtes. Un exemple de critère, le coefficient d'arête-clustering, a été développé par Radicchi et al [28]. Il s'appuie sur l'idée que les arêtes inter-communautaires participent peu aux cycles, qui en revanche sont fortement présents à l'intérieur des communautés. Il existe d'autres critères dans le sillage de l'algorithme qui a ouvert la voie aux méthodes divisionnaires et plus généralement à toute l'effervescence sur la recherche de communautés, l'algorithme GN décrit par Girvan et Newman en 2002 [50, 34].

Algorithme GN

Cet algorithme est fondé sur une mesure de centralité d'arête. Toutes les valeurs de centralité sont calculées et l'arête de plus faible centralité est supprimée (les ex æquo sont départagées aléatoirement). Ce processus est itéré jusqu'à retirer la dernière arête. Dans une seconde phase, à partir du graphe sans arête, les arêtes sont réintroduites dans l'ordre inverse, ce qui fournit une hiérarchie très fine, car en ajoutant une arête reliant deux communautés, on ajoute un niveau hiérarchique, les deux communautés étant regroupées dans une super-communauté. La mesure utilisée est la centralité d'intermédiarité (*edge-betweeness-centrality*) d'une arête, définie comme le nombre de plus courts chemins entre deux nœuds qui passent par cette arête. Dans une seconde version, les auteurs traitent deux autres mesures, dont une fondée sur une marche aléatoire. De plus, ils proposent une mesure pour distinguer les meilleures solutions parmi toutes les partitions possibles à partir du dendrogramme hiérarchique. Cette mesure est la modularité, mainte fois utilisée par la suite comme objectif d'optimisation. L'algorithme a une complexité en général de $O(mn^2)$ et $O(n^3)$ pour un graphe creux (très peu dense), le réservant à des réseaux d'une dizaine de milliers de nœuds. À noter que l'algorithme de Radicchi et al cité plus haut a une complexité au pire de $O(n^2)$ pour les graphes creux.

1.3.3 Autres méthodes

Les méthodes de détection de communautés, hors méthodes d'optimisation de la modularité ou de toute autre mesure globale, sont pléthoriques. Nous en présentons quelques-unes parmi les plus connues et efficaces pour la détection de communautés (voir les tests de Danon et al [51] et de Lancichinetti et al [41]).

Le principe de marche aléatoire détermine un modèle pour de nombreux algorithmes, sur l'idée qu'un promeneur qui se déplace de nœud en nœud dans le graphe passe la plupart du temps dans la même communauté du fait de la haute densité de celle-ci ou, autrement dit, a une probabilité faible de sortir de sa communauté de départ. Ce principe est à la base d'une mesure de distance utilisable dans les algorithmes de clustering hiérarchique. Nous pouvons citer comme réalisations reconnues les algorithmes NetWalk de Zhou et Lipowsky [52], Walktrap de Latapy et Pons [53] qui sont tous deux, sans optimisation additionnelle possible, en $O(n^3)$.

Rosvall et Bergstrom ont proposé une méthode, nommée depuis Infomap, s'appuyant sur la quantité d'information nécessaire pour coder un message représentant la circulation d'information dans le graphe. Le codage qui utilise un label pour chaque communauté et un label court pour les nœuds, relatif à la communauté d'appartenance, est en principe plus court que si l'on doit nommer chaque nœud de façon unique dans tout le graphe. La circulation d'information à mesurer est évaluée par une marche aléatoire qui selon le partitionnement choisi donne un message d'encodage plus ou moins long. Finalement, l'algorithme cherche à minimiser la longueur de ce message par un hybride entre le recuit similé et une descente rapide. L'algorithme donne de bons résultats en $O(m)$, mais est plus adapté, selon les auteurs, aux graphes dirigés du fait de la mesure des flux d'information dans un graphe.

Enfin, une idée simple de propagation d'étiquette de nœud, l'étiquette désignant la communauté d'appartenance, fournit des résultats intéressants si elle est combinée avec la modularité. Chaque nœud au départ est un singleton dans sa communauté et prend ensuite la communauté d'un de ses voisins selon un critère de fréquence (la communauté la plus présente parmi les voisins est adoptée, version d'origine de LPA [54], *Label Propation Algorithm*) ou selon l'optimisation de la modularité (version améliorée LPAm [55]).

1.4 Méthodes de résolution par une optimisation globale

Nous abordons dans cette partie les méthodes approchées d'optimisation de la modularité. En effet, ce problème étant NP-difficile [56], il est nécessaire, lorsque la taille du graphe à traiter dépasse quelques centaines de nœuds, de recourir à des heuristiques pour s'approcher de l'optimum global sans garantie de l'atteindre. Ces méthodes s'appuient sur des techniques de recherche, locale ou non, par transformation de partition avec pour objectif l'optimisation de Q. Elles présentent l'avantage d'être adaptables à toute autre mesure globale de qualité de partitionnement, à condition que l'évaluation de ΔQ pour les opérateurs de transformation soit peu coûteuse en temps de calcul, comme pour la modularité.

1.4.1 Algorithmes classiques : Greedy et VM

Greedy

A la suite de ses travaux sur l'algorithme GN qui utilise la modularité pour distinguer la meilleure partition, Newman a imaginé une méthode [57] qui optimise directement la modularité, en s'inspirant des techniques agglomératives du clustering hiérarchique. Cette méthode est dite "gloutonne" (*greedy*), car elle procède par une succession de fusions de paires de communautés qui augmentent la modularité. La différence de modularité engendrée par la fusion de

deux communautés C et C' s'écrit comme suit, puisque la communauté $C \cup C'$ est ajoutée à la partition alors que C et C' disparaissent :

$$\begin{aligned}\Delta_f Q(\mathcal{P}, C, C') &= \frac{l(C \cup C')}{m} - \left(\frac{d(C \cup C')}{2m}\right)^2 \\ &\quad - \left[\frac{l(C)}{m} - \left(\frac{d(C)}{2m}\right)^2\right] - \left[\frac{l(C')}{m} - \left(\frac{d(C')}{2m}\right)^2\right]\end{aligned}$$

Pour évaluer l'expression $l(C \cup C')$, nous devons introduire une notation importante pour la suite de notre exposé, le nombre $l(C, C')$ d'arêtes entre C et C' soit $|\{\{u, v\} \in E, u \in C \text{ et } v \in C'\}|$. Ceci étant posé, $l(C \cup C')$ s'écrit $l(C) + l(C') + l(C, C')$. Le degré de communauté étant défini comme la somme des degrés des nœuds la composant, nous avons $d(C \cup C') = d(C) + d(C')$, soit pour la variation de modularité :

$$\begin{aligned}\Delta_f Q(\mathcal{P}, C, C') &= \frac{l(C) + l(C') + l(C, C')}{m} - \left(\frac{d(C) + d(C')}{2m}\right)^2 \\ &\quad - \left[\frac{l(C)}{m} - \left(\frac{d(C)}{2m}\right)^2\right] - \left[\frac{l(C')}{m} - \left(\frac{d(C')}{2m}\right)^2\right] \\ &= \frac{l(C, C')}{m} - \frac{d(C)d(C')}{2m^2} \qquad (1.5)\end{aligned}$$

Dans un graphe pondéré, la formule devient :

$$\Delta_f Q(\mathcal{P}, C, C') = 2\frac{d(C, C')}{d(V)} - 2\frac{d(C)d(C')}{d(V)^2} \qquad (1.6)$$

Le degré $d(C, C')$ entre deux communautés est égal à la somme des poids des arêtes reliant les deux communautés soit $d(C, C') = \sum_{u \in C \text{ et } v \in C'} w(u, v)$.

Dans l'algorithme Greedy, Newman stocke un vecteur $d(C)$ et une matrice $l(C, C')$ (ou $d(C, C')$ si le graphe est pondéré), ces données étant mises à jour incrémentalement après chaque fusion, pour réaliser le calcul de ΔQ en temps constant. À chaque étape, l'algorithme choisit la paire de communautés dont la fusion produit la plus grande variation de Q positive. Il en résulte une complexité de temps en $O((m + n)n)$ dans le cas général et $O(n^2)$ pour un graphe creux, et tout de même une complexité en espace pour enregistrer la matrice des degrés de communautés en $O(n^2)$ au pire. Nous verrons plus loin une façon moins coûteuse de stocker ces données avec le graphe des communautés.

L'algorithme a été amélioré par la suite sous le nom de *FastGreedy* par Clauset et al [58] en classant les valeurs de $l(C, C')$ dans un arbre binaire. La complexité au pire pour un graphe creux devient $O(n \log(n)^2)$ permettant de traiter des graphes de plus d'un million de nœuds. L'algorithme offre une optimisation rapide mais grossière de Q. L'opérateur que nous présentons dans la section suivante permet un raffinement de l'optimisation, en complément de l'algorithme glouton.

Vertex-Mover

En 2008, Schuetz, et Caflisch [59] ont publié une technique nommée MSG (*Multi Step Greedy*) qui étend la procédure *Greedy* de Newman en promouvant plusieurs fusions à chaque étape. Ils y ont ajouté un raffinement, appelé *vertex mover*, qui s'inspire de l'opérateur de déplacement de nœud de l'algorithme de Kernighan-Lin.

Chaque nœud est examiné à tour de rôle dans un ordre prédéterminé qui peut être aléatoire ou par exemple fixé par le degré croissant ou décroissant. L'examen d'un nœud consiste à évaluer son déplacement vers chaque communauté voisine, telle qu'il existe au moins un nœud de

cette communauté partageant une arête avec le nœud examiné. Parmi tous ces déplacements possibles, celui qui engendre la plus grande augmentation de la modularité est exécuté. Si aucun déplacement n'engendre une variation positive de Q, le nœud reste dans sa communauté et l'algorithme passe au suivant.

Pour obtenir un optimum local selon cet opérateur, l'algorithme doit s'arrêter lorsque, en ayant examiné tous les nœuds, aucun n'est déplacé. Un critère d'arrêt plus relâché consiste à s'arrêter quand la variation de modularité générée par un examen complet des nœuds est inférieure à un seuil fixé par avance, qui mesure donc une précision décimale souhaitée pour la modularité.

La différence de modularité engendrée par le déplacement du nœud v vers la communauté C s'écrit comme suit, sachant que les deux termes dans l'expression de Q qui changent concernent les communautés C et $\mu(v)$, la première gagnant le nœud v, et la deuxième le perdant :

$$\begin{aligned}
\Delta_d Q(\mathcal{P}, v, C) &= \frac{l(C \cup \{v\})}{m} - \left(\frac{d(C \cup \{v\})}{2m}\right)^2 - \left[\frac{l(C)}{m} - \left(\frac{d(C)}{2m}\right)^2\right] \\
&\quad + \frac{l(\mu(v) \setminus \{v\})}{m} - \left(\frac{d(\mu(v) \setminus \{v\})}{2m}\right)^2 - \left[\frac{l(\mu(v))}{m} - \left(\frac{d(\mu(v))}{2m}\right)^2\right] \\
&= \frac{l(C) + l(C, \{v\})}{m} - \left(\frac{d(C) + d(v)}{2m}\right)^2 - \frac{l(C)}{m} + \left(\frac{d(C)}{2m}\right)^2 \\
&\quad + \frac{l(\mu(v)) - l(\mu(v), \{v\})}{m} - \left(\frac{d(\mu(v)) - d(v)}{2m}\right)^2 - \frac{l(\mu(v))}{m} + \left(\frac{d(\mu(v))}{2m}\right)^2 \\
&= \frac{l(C, \{v\}) - l(\mu(v), \{v\})}{m} + d(v)\left(\frac{d(\mu(v)) - d(C) - d(v)}{2m^2}\right) \qquad (1.7)
\end{aligned}$$

L'expression $l(C, \{v\})$ désigne le nombre d'arêtes ayant le nœud v pour extrémité et l'autre extrémité dans la communauté C. Pour un graphe pondéré, la formule devient :

$$\Delta_d Q(\mathcal{P}, v, C) = 2\left(\frac{d(C, \{v\}) - d(\mu(v), \{v\})}{d(V)}\right) - 2d(v)\left(\frac{d(C) - d(\mu(v)) - d(v)}{d(V)^2}\right) \qquad (1.8)$$

L'évaluation de cette formule nécessite de parcourir tous les nœuds adjacents à v pour calculer $l(C, \{v\})$ et $l(\mu(v), \{v\})$. Sa complexité moyenne, pour un nœud examiné et une communauté, est donc en $O(\bar{d})$, $\bar{d}$ étant le degré moyen dans le graphe G. La complexité augmente pour l'évaluation complète d'un nœud, car il faut comparer toutes les communautés voisines. Il est préférable de réaliser l'examen des voisins en une seule fois et d'évaluer incrémentalement $l(C, \{v\})$ et $l(\mu(v), \{v\})$ en les stockant dans un vecteur de communautés. Ainsi, la complexité pour l'examen d'un nœud reste en $O(\bar{d})$. Une itération principale de l'algorithme (un tour) consiste en l'examen de tous les nœuds soit une complexité en temps pour un tour de $O(n\bar{d}) = O(n\frac{2m}{n}) = O(m)$. Le nombre de tours nécessaires à l'optimisation est difficile à estimer, car il dépend de la structure communautaire du graphe et de sa taille. En pratique, avec les graphes de tests présentés dans le chapitre suivant qui sont très peu denses ($m \ll n^2$), on constate une complexité en temps de cette procédure VM (*vertex mover*) en $O(m)$.

1.4.2 Métaheuristiques d'optimisation

Nous devons la première application d'une méta-heuristique à Guimerà et al qui ont utilisé un recuit simulé [60] pour optimiser la modularité à partir de déplacements de nœud pris au hasard qui conduisent à des optimums locaux. Pour en sortir, les auteurs emploient des fusions et des séparations de communautés. La séparation d'une communauté consiste à découper le

sous-graphe de la communauté en deux parts égales (algorithme de bipartition) en optimisant Q. Cette méthode est très lente (la complexité exacte est délicate à évaluer), mais est réputée pour fournir d'excellents résultats.

Les algorithmes génétiques [61] ont aussi été utilisés pour l'optimisation de la modularité par Clara Pizzuti [62, 63], Tasgin et al [64] et Shi et al [65]. La majorité de ces expériences utilisent une représentation de partition issue d'une application d'algorithme génétique au partitionnement classique de graphe [66] qui favorise les croisements. Ces algorithmes, quoiqu'instructifs sur leur fonctionnement, ont une performance d'optimisation voisine de celle des algorithmes LPA ou Louvain [67], mais ne rivalisent pas avec les meilleurs que sont LPAm et l'algorithme multiniveaux de Noack et Rotta [68].

Citons aussi une expérience de recherche tabu [69] qui s'attache à optimiser la modularité par des déplacements de nœuds et une opération de perturbation pour s'échapper d'un optimum local. Son objectif majeur porte sur l'optimisation de la modularité sans atteindre les bons résultats de la méthode de Louvain, par défaut de fusions de communautés qui apportent un saut quantitatif important pour l'optimisation. Enfin, signalons l'une des premières expériences d'algorithme mémétique [70] concomitante avec la nôtre qui introduit efficacement sur quelques petits graphes réels testés la notion de densité de modularité qui agrège deux objectifs essentiels pour minimiser la limite de résolution, la maximisation à la fois la modularité et de la densité de communauté. Cette approche rejoint et conforte les résultats que nous présentons dans les deux derniers chapitres. Elle ouvre aussi la voie aux méthodes multi-objectifs.

1.4.3 Méthodes multirésolutions et multiniveaux

Les réseaux complexes, en particulier les réseaux hiérarchiques, ont typiquement plusieurs structures communautaires à différentes échelles. Les algorithmes fondés sur une mesure globale se heurtent à la difficulté de capturer cette structure stratifiée en optimisant continûment la même fonction. La limite de résolution traduit cette difficulté. Les approches multirésolutions, où l'on introduit un paramètre de résolution dans la modularité, et multiniveaux, où un algorithme exécute la même procédure sur toutes les strates du réseau, permettent de pallier partiellement ce défaut.

Dans l'approche multirésolutions, la modularité ou toute autre fonction globale, intègre un paramètre de résolution que l'on peut faire varier pour percevoir les communautés à une échelle déterminée. Arenas et al.[71] ajoutent des boucles de poids r à chaque nœud, introduisant ainsi la résolution r dans la modularité. Dans la méthode de Reichardt et Bornholdt [37], que nous utilisons dans notre exposé, la modularité devient :

$$Q_\lambda(\mathcal{P}) = \sum_{C\in\mathcal{P}} \left(\frac{l(C)}{m} - \lambda \left(\frac{d(C)}{2m} \right)^2 \right) \tag{1.9}$$

Si λ est petit, un algorithme qui partitionne en optimisant Q_λ a tendance à construire des grandes communautés. À la limite lorsque $\lambda = 0$, le graphe complet forme une seule communauté. Inversement, en augmentant λ, les petites communautés apparaissent pour ne contenir qu'un seul nœud lorsque $\lambda \to \infty$.

Le principe multiniveaux a été introduit et développé pour le partitionnement de graphe [72, 73] et donne de très bons résultats en réduisant la complexité des algorithmes de partitionnement. Dans une première phase, une méthode gloutonne agglomère les groupes jusqu'à un certain degré en partant de communautés singletons. La partition ainsi obtenue est transformée en un graphe "grossier" où un groupe devient un nœud en utilisant les poids d'arête pour représenter le nombre de liens entre deux groupes. Ce graphe est de taille réduite et le même processus peut lui être appliqué jusqu'à obtention d'un graphe qui ne peut plus être partitionné. Dans la

seconde phase, en procédant à l'envers, les groupes de niveau supérieur sont projetés aux niveaux inférieurs, en affinant le partitionnement à chaque étape par un algorithme comme celui de Kernighan-Lin. Ce principe est très efficace et permet de traiter des graphes de très grande taille. Il a été adapté au problème de la détection de communautés par Noack et Rotta [68] et par Blondel et al. dans l'algorithme de Louvain [67] que nous décrivons dans la section qui suit.

1.4.4 Algorithme de Louvain

Les algorithmes multiniveaux pour le partitionnement de graphe utilisent un partitionnement grossier lors de la première phase puis un raffinement précis pendant la seconde phase. La définition de la détection de communautés comme un problème d'optimisation d'une fonction globale permet d'appliquer cette optimisation dès la première phase. C'est le cas de l'algorithme de Noack et Rotta qui utilise l'agglomération de communautés dirigée par la maximisation de Q lors de la première phase. Il obtient de meilleurs résultats que l'algorithme de Louvain, mais avec un temps d'exécution plus long.

Dans l'algorithme de Louvain, la première phase utilise l'heuristique VM (*vertex mover*). À partir d'un graphe d'origine à partitionner $G^0 = G$, l'application de VM au premier niveau 0 donne une partition $\mathcal{P}^0$. Le graphe de niveau 1, noté G^1 est engendré à partir de $\mathcal{P}^0$ selon la procédure décrite plus bas. Au niveau 1, l'heuristique VM est appliquée à G^1 et cette procédure récursive se poursuit jusqu'au dernier niveau L où VM ne déplace aucun nœud, c'est-à-dire lorsque $|\mathcal{P}^L| = |G^L|$.

Pour passer d'un niveau l au niveau supérieur $l+1$, chaque nœud de G^{l+1} est associé à une communauté de $\mathcal{P}^l$ par une fonction que nous notons $T^l : \mathcal{P}^l \longmapsto G^{l+1}$. La construction de G^{l+1} en fonction de G^l et de $\mathcal{P}^l$ suit les règles suivantes :

1. À chaque communauté C de $\mathcal{P}^l$ est associé un nœud $T^l(C)$ dans G^{l+1}.
2. Une arête existe entre les nœuds $T^l(C_1)$ et $T^l(C_2)$, représentant les communautés C_1 et C_2 de $\mathcal{P}^l$, s'il existe au moins une arête ayant une extrémité dans C_1 et l'autre dans C_2.
3. Le poids d'une arête entre les nœuds $T^l(C_1)$ et $T^l(C_2)$ est égal à la somme des poids des arêtes entre les deux communautés soit $d(C_1, C_2)$.
4. Une boucle est ajoutée à chaque nœud $T^l(C)$ avec un poids égal au degré interne de la communauté C dans $\mathcal{P}^l$, soit $d_{in}(C) = d(C, C)$ soit encore $2 \sum_{u \in C \text{ et } v \in C} w(u, v)$.

Cette procédure est illustrée par la figure 1.5. Nous obtenons finalement une partition hiérarchique dans laquelle les communautés de niveau L contiennent des sous-communautés de niveau inférieur $L-1$ et ainsi de suite jusqu'aux nœuds du graphe d'origine. Cet algorithme est le plus rapide pour optimiser la modularité avec une complexité constatée sur des instances de graphes peu denses en $O(m)$, qui autorise le traitement de très grands graphes. Son optimisation de Q est correcte, mais n'égale pas celle des méta-heuristiques et souffre surtout de dégénérescence comme nous le mettrons en évidence par la suite. Toutefois, cet algorithme fournit un compromis très intéressant entre performance et vitesse d'optimisation pour une utilisation dans les algorithmes d'optimisation poussée que nous avons développés.

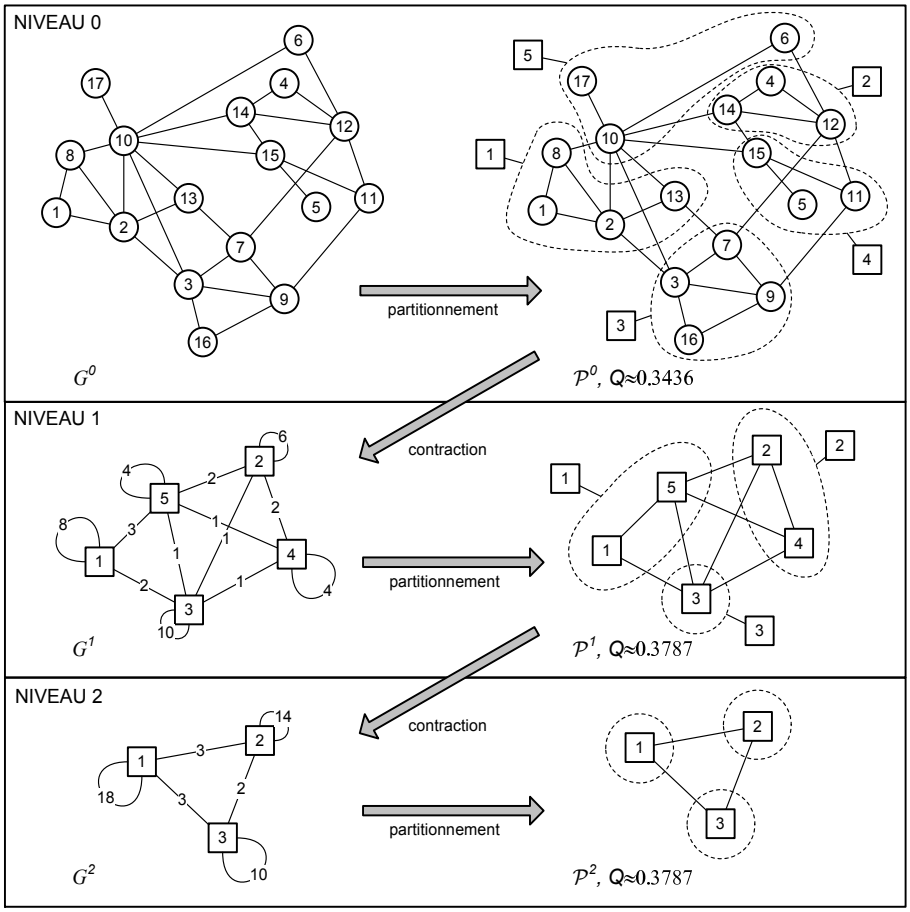


FIGURE 1.5 – Illustration de l'algorithme de Louvain. Au niveau 0, le partitionnement par l'heuristique VM du graphe G^0 donne un découpage en cinq communautés (partition $\mathcal{P}^0$). Dans le graphe contracté du niveau suivant, il y a cinq nœuds (un par communauté du niveau précédent) et les arêtes sont pondérées de façon à ce que la modularité soit la même, si l'on prend des communautés singletons (un nœud dans chacune) dans le nouveau graphe. Le processus se poursuit avec ce graphe jusqu'au niveau 2 où la modularité de la partition singleton de départ ne peut pas être améliorée.

2

Protocole d'expérimentation

Sommaire

Nous présentons ici les modalités de tests et de comparaison des nouveaux algorithmes de détection de communautés décrits dans les chapitres suivants. Tous ces algorithmes sont soumis aux mêmes graphes de test, certains réels, issus de relevés de spécialistes des domaines d'application, et d'autres artificiels selon le modèle de génération LFR. Nous décrivons enfin les mesures de performance qui permettront d'apprécier l'efficacité de nos algorithmes, d'abord pour optimiser la modularité, ensuite pour réduire les effets de la limite de résolution.

Introduction

Pour mesurer les performances des algorithmes existants et de ceux que nous présentons, nous avons choisi deux jeux de tests complémentaires. Le premier est constitué de graphes dits "réels", issus d'études et de relevés de données dans les différents champs d'application des réseaux complexes. Les réseaux que nous avons choisis sont couramment utilisés pour étudier les résultats des algorithmes de détection de communautés. Pour le second jeu, nous avons généré des graphes qui présentent certaines caractéristiques des réseaux complexes, choisies par avance d'après un modèle dénommé LFR. Ce type de graphe a l'avantage d'être associé dès sa génération à une partition idéale qui sert de référence pour valider les partitions proposées par les algorithmes testés.

Pour mesurer les performances structurelles des algorithmes et apprécier l'étendue des défauts de la modularité, nous utilisons des mesures intrinsèques pour les graphes réels, sans évidemment aucune possibilité de comparaison avec une solution type. Pour les graphes artificiels, nous utilisons des mesures de similarité entre partitions. Ces mesures sont présentées dans la troisième section de ce chapitre.

Enfin, nous détaillons les conditions de comparaison des algorithmes testés, avec en particulier le délicat problème de la nature déterministe ou stochastique de ces algorithmes.

2.1 Graphes réels

De nombreux graphes ont été compilés par des spécialistes des domaines où les réseaux complexes apparaissent, essentiellement des réseaux sociaux, des réseaux de protéines et des réseaux de collaboration scientifique. Nous en avons choisis 22, parmi les plus utilisés, pour offrir un panel assez large du plus petit avec un peu moins d'une centaine d'arêtes au plus grand qui atteint presque le million d'arêtes. La liste de ces graphes est fournie dans le tableau 2.1. Ces graphes sont simples, non orientés, et seuls deux d'entre eux sont pondérés, *USAir97* et *Condmat2003*.

2.2 Graphes artificiels

Nous avons choisi le modèle de graphes artificiels présenté par Lancichinetti et al. en 2008 [40] comme un prolongement et une amélioration du modèle GN de Girvan et Newman. L'algorithme de génération détermine aléatoirement un graphe et une structure communautaire à partir des paramètres suivants :

- le nombre de nœuds n ;
- le degré de nœud moyen $\bar{d}$ et le degré de nœud maximum $\max d_i$;
- l'exposant de distribution en loi de puissance des degrés de nœuds, noté α_v ;
- l'exposant de distribution en loi de puissance des tailles de communautés, noté α_c ;
- l'indice de communauté (*mixing parameter*), noté μ, qui représente la part des arêtes externes d'un nœud, c'est-à-dire des arêtes ayant une extrémité en dehors de la communauté du nœud ;
- les tailles minimale et maximale de communautés, notées $\min |C_i|$ et $\max |C_i|$.

TABLE 2.1 – Liste des graphes réels de test des algorithmes de détection de communautés.

Graph	Description	n	m	Source
Karate Club	club de karaté Zachary	34	78	[74]
Dolphins	réseau d'association de dauphins	62	159	[6]
Political Books	réseau d'achat de livres politiques	105	441	[75]
College Football	tournois entre équipes de football	115	613	[50]
Codeminer	structure de code en Java	724	1015	[76]
C. elegans	réseau métabolique du nématode C.Elegans	453	2025	[77]
Jazz	réseau de collaboration entre musiciens de jazz	198	2742	[78]
E-mail	réseau d'emails universitaires	1133	5451	[79]
Power	topologie du réseau d'alimentation des États de l'ouest des USA	4941	6594	[80]
Yeast	réseau d'interaction entre protéines yeast	2284	6646	[81]
Epa	liens entre pages dans le moteur de recherche www.epa.gov	4271	8909	[82]
Erdos	réseau de collaboration Erdös	6927	11850	[83]
California	pages correspondant à la requête "California" dans un moteur de recherche	6175	15969	[84]
Arxiv	réseau de citations entre articles scientifiques	9377	24107	[85]
PGP	réseau de signatures mutuelles par clé de cryptographie	10680	24316	[86]
Zemail	réseau d'emails	6640	54173	[87]
Condmat2003	réseau de collaboration d'écriture d'article dans le domaine de la matière condensée en physique	27519	116181	[88]
Astro-ph	réseau de collaboration d'écriture d'article en astrophysique	16046	121251	[89]
Enron	réseau d'emails provenant d'Enron	36692	183831	[90]
Brightkite	réseau d'amitié à partir d'un service de localisation	58228	214078	[91]
Slashdot	réseau social issu du site d'information Slashdot	77352	468938	[90]
Gowalla	réseau d'amitié à partir d'un service de localisation	196591	950326	[91]

L'algorithme choisit tout d'abord une distribution aléatoire des tailles de communauté qui suit la loi de puissance $P(\delta) \sim \delta^{-\alpha_c}$ et de sorte évidemment que la somme des tailles soit égale à n. Il en résulte un nombre de communautés k que l'on ne peut pas fixer à l'avance. Ensuite, l'algorithme procède à partir de nœuds isolés, par un rattachement itératif au graphe en construction en respectant le degré moyen, la borne maximale de degré et la distribution en loi de puissance $P(\delta) \sim \delta^{-\alpha_v}$. En même temps, chaque nœud v est affecté à une communauté de sorte que $\frac{d_{out}(v)}{d(v)} \sim \mu$. Certains nœuds sont reconnectés et réaffectés si ces conditions ne peuvent pas être remplies. L'algorithme s'arrête lorsque tous les nœuds sont rattachés au graphe et affectés à une communauté. Il s'exécute généralement en temps linéaire selon la taille du graphe bien que ce ne soit pas une garantie.

Nous avons choisi des paramètres de génération et sélectionné 20 graphes pour couvrir une large variété de cas types rencontrés dans les réseaux complexes selon l'ordre des graphes, leur nombre d'arêtes, leur densité, la répartition des degrés de nœuds, la répartition des tailles de communautés et enfin la caractérisation plus ou moins forte des communautés (paramètre μ). La liste des graphes choisis est présentée dans le tableau 2.2, leur typologie dans le tableau 2.3 et enfin leurs mesures de qualité dans le tableau 2.4.

Ce modèle permet de tester les algorithmes de partitionnement en comparant la partition trouvée par un algorithme avec celle de référence générée avec le graphe. Il engendre des partitions plus réalistes que le modèle GN car la taille des communautés varie. Les communautés respectent les propriétés caractéristiques des réseaux complexes, avec toutefois une structure plus régulière. Cela doit nous conduire à relativiser les résultats obtenus par ces graphes artificiels qui forment un sous-ensemble particulier de l'ensemble des réseaux complexes.

TABLE 2.2 – Liste des graphes artificiels de test des algorithmes de détection de communautés.

graph	n	m	$\min d_i$	$\bar{d}$	$\max d_i$	k	$\min \overline{\lvert C_i \rvert}$	$\max \overline{\lvert C_i \rvert}$	α_v	α_c	μ
#1	80	237	4	6	12	10	4	18	2	1	0.3
#2	1000	3353	3	7	30	32	3	100	2	1	0.3
#3	1000	3436	3	7	30	92	3	30	2	2	0.4
#4	1000	5000	10	10	10	100	10	10	2	1	0.5
#5	1000	7625	7	15	51	32	15	80	2	1	0.6
#6	1000	7692	7	15	50	36	15	57	2	1	0.4
#7	5000	21911	5	9	100	50	10	328	3	1	0.5
#8	2000	29878	13	30	100	58	10	81	2	1	0.7
#9	10000	34742	3	7	30	906	2	30	2	2	0.4
#10	10000	76710	7	15	50	13	510	988	2	1	0.4
#11	10000	77051	7	15	50	193	20	118	2	1	0.7
#12	5000	124908	29	50	100	65	20	182	2	1	0.8
#13	20000	146850	8	15	50	275	10	459	3	2	0.4
#14	50000	172854	3	7	30	984	4	200	2	2	0.3
#15	8000	201088	34	50	100	35	57	534	3	1	0.8
#16	30000	219864	5	15	100	699	10	912	2	2	0.6
#17	50000	248663	6	10	50	1144	10	938	3	2	0.5
#18	8000	320836	55	80	150	78	20	287	3	1	0.85
#19	50000	476468	7	19	100	1224	5	10371	2	2	0.5
#20	100000	977547	10	20	50	252	100	999	2	1	0.5

TABLE 2.3 – Scores et densités des graphes artificiels.

graph	SN	SC	SCM	δ_{min}	$\bar{\delta}$
#1	1.0000	1.0000	1.0000	0.3490	0.1209
#2	0.9990	1.0000	1.0000	0.1962	0.0238
#3	0.9280	0.9783	1.0000	0.2531	0.0649
#4	1.0000	1.0000	1.0000	0.2778	0.2778
#5	0.0280	0.0000	1.0000	0.1221	0.0354
#6	0.9980	1.0000	1.0000	0.1855	0.0893
#7	0.7028	0.5600	1.0000	0.0597	0.0064
#8	0.0000	0.0000	1.0000	0.1681	0.0559
#9	0.9312	0.9570	1.0000	0.2472	0.0678
#10	0.9999	1.0000	1.0000	0.0063	0.0046
#11	0.0005	0.0000	1.0000	0.0578	0.0201
#12	0.0000	0.0000	1.0000	0.1085	0.0264
#13	0.9995	1.0000	1.0000	0.1574	0.0096
#14	0.9984	1.0000	1.0000	0.1275	0.0111
#15	0.0000	0.0000	1.0000	0.0420	0.0117
#16	0.0754	0.0000	1.0000	0.1259	0.0035
#17	0.7881	0.5428	1.0000	0.1329	0.0029
#18	0.0000	0.0000	1.0000	0.1160	0.0207
#19	0.7287	0.5539	0.9967	0.3015	0.0005
#20	0.7652	0.5317	1.0000	0.0192	0.0049

TABLE 2.4 – Typologie des graphes artificiels.

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16	#17	#18	#19	#20
Mêmes degrés, mêmes tailles de communautés				×																
Densité faible				×	×	×										×	×			
Densité très faible	×	×	×				×		×					×						
Densité élevée								×				×			×			×		
Large étendue des tailles de communautés		×					×						×	×						
Très large étendue des tailles de communautés																×	×		×	
Communautés faiblement définies ($0.5 \leq \mu \leq 0.7$)					×			×			×					×				
Communautés très faiblement définies ($\mu > 0.7$)												×			×			×		
Queue de distribution de communauté très longue (beaucoup de petites communautés)														×		×				
Uniquement des communautés de grande taille										×										×

2.3 Mesures de qualité et de performance

Notre objectif étant de quantifier les défauts de la modularité lorsque son optimisation est poussée, il est nécessaire de se doter d'autres mesures de la qualité de partitionnement que la modularité.

Les graphes générés offrent des possibilités de mesure de similarité ou de différence structurelle entre la partition trouvée et la partition de référence. Les mesures les plus utilisées sont l'indice de Jaccard et le *Rand Index* [92], qui existent en version ajustée pour tenir compte de similarités fortuites dues au hasard. Les mesures fondées sur la théorie de l'information sont d'un usage plus récent et présentent deux principaux avantages : l'ajustement n'est pas nécessaire et leur sensibilité à la distribution des tailles de communautés est accrue, ce qui rend la mesure plus pertinente si les partitions comparées diffèrent surtout par leurs tailles communautés. Nous avons choisi les mesures NMI (information mutuelle normalisée) et NID (distance d'information normalisée).

Nous ne disposons pas d'étalon de comparaison pour la majorité des graphes réels, hormis pour quelques très petits graphes, ne dépassant pas la centaine de nœuds, qui ont été partitionnés par des spécialistes du domaine d'origine. Pour les autres graphes, nous pouvons apprécier une partition proposée par un algorithme en utilisant une grande variété de mesures. Nous avons choisi de quantifier d'une part les critères d'existence des communautés et d'autre part les densités minimale et moyenne calculées sur l'ensemble des communautés d'une partition.

Information mutuelle normalisée

Cette mesure de similarité se fonde sur l'entropie définie par Shannon en 1948. Si une source X émet un signal pouvant prendre les valeurs $\{v_1, v_2, ..., v_n\}$ avec une probabilité d'apparition p_i de la valeur v_i, l'entropie de la source est définie par $H(X) = -\sum_{i=1}^{n} p_i \log p_i$. X est souvent formalisée comme une variable aléatoire prenant n valeurs. L'entropie représente l'incertitude ou le désordre sur l'émission de la source d'information. En prenant un logarithme en base 2, $l \times H(X)$ représente le nombre moyen de bits nécessaires pour coder un message de longueur l émis par la source X. Avec deux variables aléatoires dépendantes, l'entropie conditionnelle est $H(X/Y) = -\sum_{x,y} P(X = x \cap Y = y) \log P(X = x/Y = y)$. L'information mutuelle [93] entre les deux variables est définie par $I(X, Y) = H(X) - H(X/Y)$ et représente la réduction moyenne de l'incertitude sur X sachant la valeur de Y et vice-versa (relation symétrique). Cette valeur est nulle, si la connaissance de Y n'apporte aucune information sur X donc si X et Y sont totalement non similaires. Elle est maximale si la similarité est parfaite. L'information mutuelle s'écrit :

$$I(X,Y) = \sum_{x,y} P(X = x \cap Y = y) \log \frac{P(X = x \cap Y = y)}{P(X = x)P(Y = y)} \tag{2.1}$$

Pour une partition $\mathcal{P} = \{C_1, C_2, ..., C_k\}$ à transmettre sur un support, un élément d'information est le label d'un nœud v, c'est-à-dire l'indice de sa communauté d'appartenance dans cet ensemble. En choisissant un nœud au hasard, la valeur de ce label suit une variable aléatoire qui prend ses valeurs dans $\{1, 2, ..., k\}$. La probabilité qu'une valeur particulière ν soit prise par la variable est donc $|C_\nu|/n$. L'entropie de la partition, c'est-à-dire l'incertitude sur la transmission d'un label sur un support, est donc $H(\mathcal{P}) = -\sum_{i=1}^{k} \frac{C_i}{n} \log(\frac{C_i}{n})$. Le nombre moyen de bits nécessaires à cette transmission est $kH(\mathcal{P})$. Avec deux partitions $\mathcal{P} = \{C_1, C_2, ..., C_k\}$ et $\mathcal{P}' = \{C'_1, C'_2, ..., C'_{k'}\}$ du même graphe $G = (V, E)$, la probabilité qu'un nœud pris au hasard appartienne à la communauté C_i dans $\mathcal{P}$ et C'_j dans $\mathcal{P}'$ est $|C_i \cap C'_j|/n$. Soit l'information

mutuelle entre $\mathcal{P}$ et $\mathcal{P}'$:

$$I(\mathcal{P},\mathcal{P}') = \sum_{i=1}^{k}\sum_{j=1}^{k'} \frac{|C_i \cap C'_j|}{n} \log \frac{n|C_i \cap C'_j|}{|C_i||C'_j|} \tag{2.2}$$

Si la partition $\mathcal{P}$ ne contient que des surcommunautés des communautés de $\mathcal{P}'$, ou l'inverse, l'entropie conditionnelle est toujours nulle alors qu'il y a une grande variété de partitions $\mathcal{P}$ ayant cette propriété pour une partition $\mathcal{P}'$ donnée. Ces partitions ont la même information mutuelle alors qu'elles diffèrent sensiblement. Pour pallier ce défaut, mais aussi pour permettre des comparaisons en normalisant l'intervalle de valeurs de $I(\mathcal{P},\mathcal{P}')$, seule la version normalisée [94] est utilisée pour apprécier la similarité entre deux partitions :

$$NMI(\mathcal{P},\mathcal{P}') = \frac{2I(\mathcal{P},\mathcal{P}')}{H(\mathcal{P}) + H(\mathcal{P}')} \tag{2.3}$$

Distance d'information normalisée

La NMI ne représente qu'une façon de quantifier des différences structurelles complexes entre deux partitions. Bien qu'elle soit éprouvée par une utilisation très large, nous préférons lui adjoindre une deuxième mesure, fondée également sur la théorie de l'information, mais présentant de plus l'avantage d'être une distance mathématique respectant notamment l'inégalité triangulaire. Elle se nomme distance d'information normalisée (NID) [95] et s'exprime à partir de l'information mutuelle de la sorte :

$$NID(\mathcal{P},\mathcal{P}') = 1 - \frac{I(\mathcal{P},\mathcal{P}')}{max\{H(\mathcal{P}), H(\mathcal{P}')\}} \tag{2.4}$$

Existence de communautés

Les trois principales définitions d'une communauté sont données dans la Section 1.2.1. Nous pouvons en déduire des critères pour quantifier la validité des communautés d'une partition. Au sens fort, une communauté est telle que tous ces nœuds ont un degré interne strictement supérieur à leur degré externe, c'est-à-dire plus de liens à l'intérieur de la communauté qu'à l'extérieur. Une mesure de qualité qui en découle est tout simplement le pourcentage de nœuds respectant cette condition, que nous nommons SN (score de nœud), défini par :

$$SN(\mathcal{P}) = \frac{|\{v \in V, d_{in}(v) > d_{out}(v)\}|}{n} \tag{2.5}$$

Au sens faible, une communauté a un degré interne strictement supérieur à son degré externe. Le degré interne est le double du nombre d'arêtes internes à une communauté alors que le degré externe représente le nombre d'arêtes qui ont seulement une extrémité dans la communauté. Ainsi, le nombre d'arêtes internes doit être au moins égal à la moitié du nombre d'arêtes externes. Il en découle un score, que nous désignons par SC, calculé comme la part de communautés au sens faible dans une partition :

$$SC(\mathcal{P}) = \frac{|\{C \in \mathcal{P}, d_{in}(C) > d_{out}(C)\}|}{k} = \frac{|\{C \in \mathcal{P}, l(C) > \frac{1}{4}d(C)\}|}{k} \tag{2.6}$$

Enfin, au sens le plus faible, une communauté est définie si son degré interne est strictement supérieur au nombre d'arêtes entre cette communauté et n'importe quelle autre communauté. Nous obtenons ainsi le dernier score de communauté bien définie, nommé SCM (score de communauté minimal) :

$$SCM(\mathcal{P}) = \frac{|\{C \in \mathcal{P}, d_{in}(C) > \max_{C' \in \mathcal{P}}\{l(C,C')\}\}|}{k} \tag{2.7}$$

Ces scores vont de 0 (non-respect de la définition) à 1 (respect total de la définition).

Densités dans une partition

La modularité à maximiser capture en une seule fonction l'objectif de connexions internes aux communautés nombreuses et de connexions entre communautés relativement peu nombreuses. Intuitivement, une communauté est très forte d'autant plus qu'elle se rapproche d'une clique, c'est-à-dire que sa densité tend vers 1. Dans sa composante interne avec le terme $l(C)/m$, la modularité ne traduit pas exactement ce critère de densité, car le dénominateur m a une portée globale, dans tout le graphe, alors que la densité est localisée au niveau des communautés. Cela se traduit par un problème d'échelle, d'autant plus que le graphe est grand et les communautés petites, qui aboutit à la limite de résolution. L'absorption des petites communautés par les très grandes, avec un gain de modularité, se traduit par des communautés de très grandes tailles qui, tout en respectant les critères d'existence des communautés, ont une densité de plus en plus faible et s'éloignent de l'idéal de clique. Pour apprécier ce phénomène, nous proposons d'observer la densité minimum $\delta_{min}(\mathcal{P})$ et la densité moyenne $\bar{\delta}(\mathcal{P})$ d'une partition $\mathcal{P}$ définies pour la première comme la plus petite densité de communauté et pour la seconde comme la moyenne de densité de toutes les communautés de $\mathcal{P}$.

2.4 Conditions d'expérimentation

Les meilleurs algorithmes d'optimisation de la modularité utilisent la technique d'agglomération, c'est-à-dire qu'ils construisent une partition, par fusion de communautés et déplacement de nœuds, à partir d'une partition où chaque nœud est seul dans sa communauté. Généralement, ils sont sensibles à l'ordre des nœuds et ne donnent pas les mêmes résultats pour deux graphes identiques à l'ordre des nœuds près. Pour réaliser des comparaisons d'algorithmes équitables, nous avons donc choisi de générer une fois pour toutes 100 instances de chaque graphe testé, réel et artificiel, et de soumettre ces mêmes 100 instances à tous les algorithmes testés de ce type. Pour cela, les algorithmes sont rendus déterministes de façon à ce qu'une instance de graphe soumise plusieurs fois à un algorithme donne toujours la même partition en résultat. En général, sauf mention contraire, un indicateur donné en résultat pour un graphe est la moyenne de l'indicateur pour les 100 instances.

Nous utilisons une version déterministe des algorithmes Louvain, Louvain+ (Chapitre 3) et Louvain+ avec condition de fusion (Chapitre 5). Pour cela, la procédure VM utilisée en première phase de ces algorithmes reprend l'ordre des nœuds du fichier en entrée représentant un graphe. Exceptionnellement pour un test réduit de paramètre (algorithme Mr-Come, Section 6.2.1) nous utilisons une version stochastique de telle sorte que les nœuds sont aléatoirement réordonnés juste après la lecture du fichier et avant l'exécution de l'algorithme. Ce principe ne s'applique pas aux algorithmes évolutionnaires (algorithmes MA-COM et MemCM des Chapitres 4 et 7) qui sont stochastiques par nature. Un individu de la population initiale est produit par la version stochastique de l'algorithme Louvain+, de façon à produire une partition différente à chaque fois. Pour un graphe donné, l'algorithme testé est exécuté 100 fois avec ce graphe et tout indicateur donné en résultat est la moyenne de l'indicateur pour ces 100 exécutions.

Les algorithmes que nous présentons sont implantés en Free Pascal et compilés sur une plate-forme Windows x86. La plate-forme d'exécution est dotée d'un processeur Intel i5 750 4 cœurs cadencé à 2.67 GHz et de 4 Go de mémoire vive.

Optimisation pure de la modularité

3

Algorithme Louvain+

Sommaire

Ce chapitre est consacré à notre contribution pour améliorer l'efficacité de l'algorithme de Louvain. Nous y ajoutons une phase de raffinement, que l'on trouve classiquement dans les algorithmes multiniveaux. L'optimisation de la modularité est légèrement améliorée alors que la justesse des partitions augmente significativement en restant pourtant très éloignée en moyenne de l'idéal, avec un nombre de communautés détectées deux fois trop faible sur les graphes artificiels. Ainsi, l'impact sur les défauts de la modularité est faible. Toutefois, ce raffinement contribue à renforcer la cohésion des communautés dont les scores d'existence se rapprochent de l'idéal.

Ce chapitre a été présenté à la conférence *Artificial Evolution (EA-2013)* et publié dans ses actes.

Introduction

Nous nous attachons dans la première partie à repousser les limites connues de l'optimisation de la modularité pour en observer l'impact sur les défauts de cette fonction. Ce chapitre décrit une amélioration de l'algorithme de Louvain, que nous utiliserons dans tous nos algorithmes. Dans le chapitre suivant, nous présentons un algorithme génétique hybride, utilisant l'algorithme de Louvain pour son optimisation locale. Cet algorithme procure des valeurs de modularité impossibles à atteindre avec les recherches locales dédiées à la détection de communauté. L'impact sur les défauts de la modularité est significatif, confirmant la limite de résolution, mais réduisant la dégénérescence.

L'algorithme de Louvain emprunte le principe multiniveau du partitionnement de graphe avec la particularité de rechercher une solution, en optimisant la modularité, dès la première phase de contraction. Un algorithme s'appuyant sur le paradigme multiniveau réduit le graphe à traiter en regroupant les nœuds en *clusters* selon un certain critère, procède à une recherche de solution sur le graphe réduit (par exemple, recherche un partitionnement) et enfin effectue un raffinement lors de la phase d'expansion. L'algorithme de Louvain appliqué à la détection de communautés se contente de la solution obtenue à l'issue de la première phase, présentant l'avantage de fournir une hiérarchie de communautés. Nous proposons d'améliorer l'optimisation de la modularité en ajoutant à l'algorithme de Louvain une phase d'expansion avec raffinement. La hiérarchie est perdue du fait de l'expansion, mais elle peut être préservée par une implémentation particulière sans expansion.

3.1 Principe du raffinement

L'algorithme de Louvain, décrit en Section 1.4.4 est formé de deux phases :

1. Phase d'initialisation : le graphe d'origine est découpé en communautés par la procédure VM.

2. Phase de contraction : à chaque niveau, le graphe et la partition obtenus sont réduits en un graphe où les communautés deviennent au niveau suivant des nœuds, ce graphe étant partitionné par la procédure VM et passé au niveau supérieur jusqu'à obtenir un graphe non partitionnable.

Nous ajoutons une troisième phase, dite d'expansion et de raffinement, qui consiste d'une part à projeter les communautés vers les niveaux inférieurs et d'autre part à affiner le placement des nœuds après chaque projection par une simple procédure VM. En effet, un nœud placé en première phase par VM va appartenir à de nouvelles super-communautés au fur est à mesure de la contraction, mais ne change pas de communauté directe d'appartenance. Lors de l'expansion, le nœud est versé dans sa plus haute communauté, les intermédiaires disparaissant, et sa situation doit être revue, car un placement dans une autre communauté peut améliorer la modularité.

Les deux premières phases de l'algorithme de Louvain donnent, à partir d'un graphe à partitionner G, les graphes de niveaux notés $(G^0, G^1, ..., G^L)$ et les partitions de niveaux notées $(\mathcal{P}^1, \mathcal{P}^2, ..., \mathcal{P}^L)$ de sorte que $G^0 = G$ (graphe de départ) et $|\mathcal{P}^L| = |G^L|$ (VM inopérante sur le dernier graphe). A noter que la partition pertinente de plus haut niveau est obtenue au niveau $L-1$, car le graphe G^L n'est pas partitionné. La troisième phase d'expansion consiste, en partant de $\mathcal{P}^{L-1}$, à aller à rebours en démarrant par la projection de $\mathcal{P}^{L-1}$ dans $\mathcal{P}^{L-2}$ pour donner une nouvelle partition notée $\bar{\mathcal{P}}^{L-2}$. Ensuite, à chaque niveau l de $L-2$ à 0, la partition $\bar{\mathcal{P}}^l$ est améliorée par l'heuristique VM puis projetée au niveau inférieur pour donner $\bar{\mathcal{P}}^{l-1}$. Finalement,

Algorithme 1 Pseudo-code de l'algorithme Louvain+

Require: Graphe $G = (V, E)$.
Ensure: Une partition $\mathcal{P}$ de G avec une modularité maximale.

$l \leftarrow 0$
$G^0 \leftarrow G$
$arret =$ **false**
repeat
  $\mathcal{P}^l \leftarrow$ PartitionnerEnSingleton(G^l) /* un nœud par communauté */
  $\mathcal{P}^l \leftarrow$ VertexMover($\mathcal{P}^l$)
  if $|\mathcal{P}^l| < |G^l|$ **then**
    $G^{l+1} \leftarrow$ ContracterGraphe($G^l, \mathcal{P}^l$) /* génération du graphe contracté de niveau supérieur */
    $l \leftarrow l + 1$
  else
    $arret =$ **true**
  end if
until $arret$
$l \leftarrow l - 1$ /* partition singleton au niveau l, il faut descendre d'un niveau */
$\bar{\mathcal{P}}^l \leftarrow \mathcal{P}^l$ /* $\bar{\mathcal{P}}$ est la partition projetée et améliorée par VM */
while $l > 0$ **do**
  $\bar{\mathcal{P}}^{l-1} \leftarrow$ Projeter($\bar{\mathcal{P}}^l, \mathcal{P}^{l-1}$)
  $\bar{\mathcal{P}}^{l-1} \leftarrow$ VertexMover($\bar{\mathcal{P}}^{l-1}$)
  $l \leftarrow l - 1$
end while
$\mathcal{P} = \bar{\mathcal{P}}^0$ /* partition résultat */

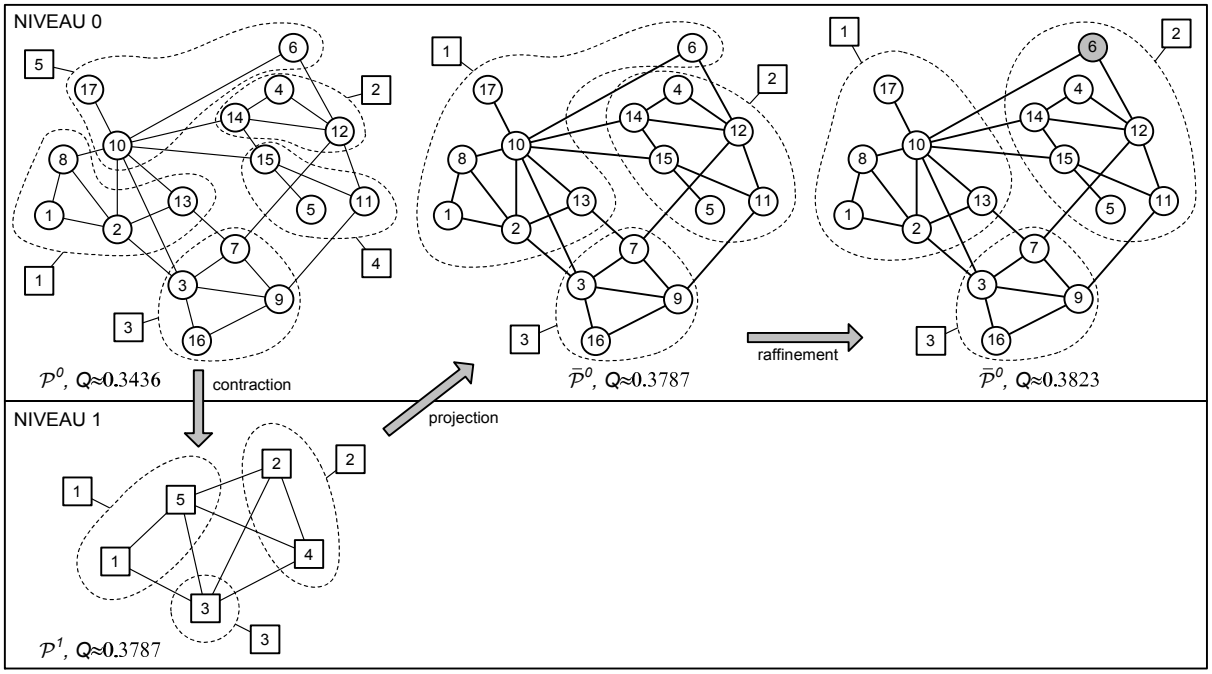


FIGURE 3.1 – Illustration de l'algorithme Louvain+. Dans Louvain, la première phase d'initialisation donne la partition $\mathcal{P}^0$, puis la phase de contraction $\mathcal{P}^1$ avec trois communautés. Dans la troisième phase propre à Louvain+, ces trois communautés sont projetées dans la partition inférieure $\mathcal{P}^0$. Par exemple les nœuds 1, 2, 6, 8, 10, 13 et 17 sont versés directement dans la communauté 1 de $\bar{\mathcal{P}}^0$ par la projection, car ils appartiennent aux sous-communautés 1 ou 5 de la communauté 1 dans $\mathcal{P}^1$. Le raffinement en application de VM à $\bar{\mathcal{P}}^0$ améliore la modularité en déplaçant le nœud 6 dans la communauté 2.

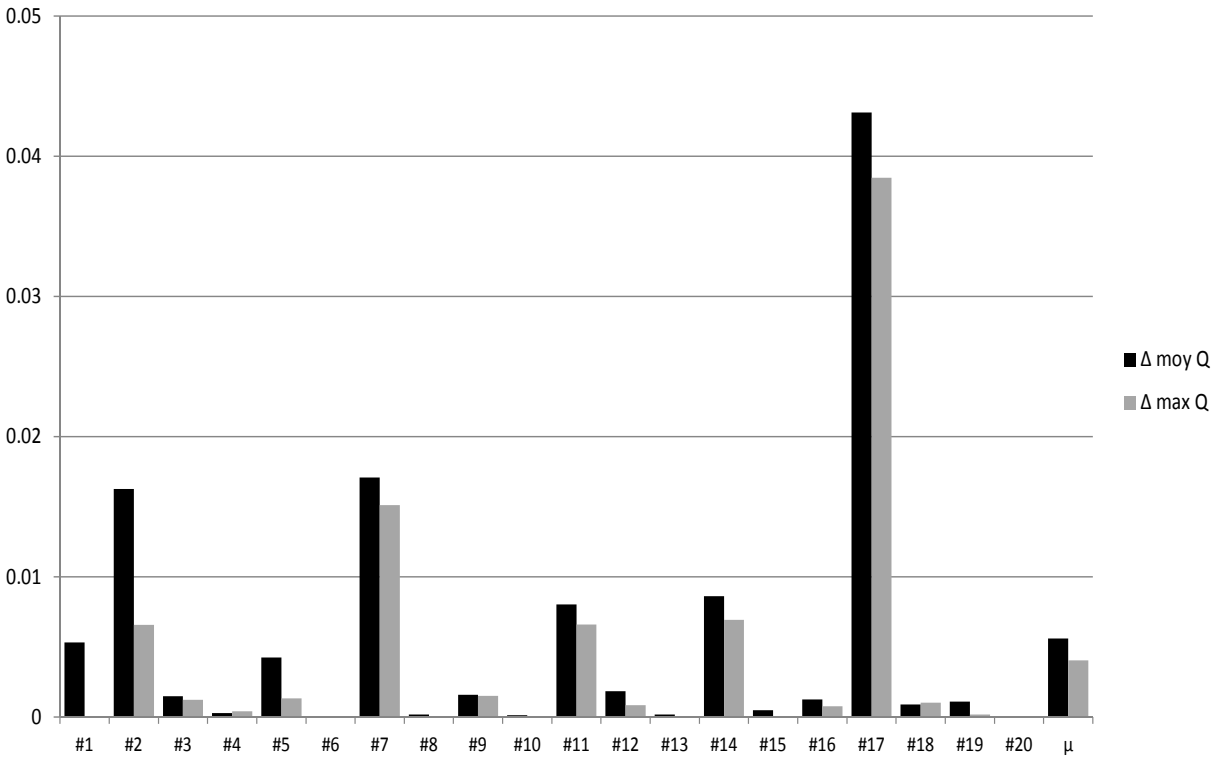


FIGURE 3.2 – Modularité comparée entre Louvain et Louvain+ pour les graphes artificiels. Les barres noires (resp. claires) représentent la différence de modularité moyenne (resp. maximale), calculée sur les 100 instances de graphe, entre Louvain+ et Louvain. L'élément μ représente la moyenne pour l'ensemble des graphes.

la partition $\bar{\mathcal{P}}^0$ est la partition du graphe G fournie par la version améliorée de l'algorithme de Louvain, que nous appelons simplement Louvain+ (schéma général dans l'Algorithme 1).

Formalisons l'opération de projection d'une partition $\bar{\mathcal{P}}^{l+1}$ de niveau $l+1$ dans la partition de niveau inférieur $\mathcal{P}^l$ pour donner une nouvelle partition $\bar{\mathcal{P}}^l$. Soit deux nœuds du graphe G^l notés v_1^l et v_2^l. Leurs communautés respectivement dans $\bar{\mathcal{P}}^l$ sont notés $\mu_{\bar{\mathcal{P}}^l}(v_1^l)$ et $\mu_{\bar{\mathcal{P}}^l}(v_2^l)$ et, d'après la notation définie en section 1.4.4, les nœuds correspondants dans le graphe réduit de niveau supérieur sont $T^l(\mu_{\bar{\mathcal{P}}^l}(v_1^l))$ et $T^l(\mu_{\bar{\mathcal{P}}^l}(v_2^l))$. Si et seulement si ces nœuds appartiennent à la même communauté dans la partition de niveau $l+1$, c'est-à-dire si $\mu_{\bar{\mathcal{P}}^{l+1}}(T^l(\mu_{\bar{\mathcal{P}}^l}(v_1^l))) = \mu_{\bar{\mathcal{P}}^{l+1}}(T^l(\mu_{\bar{\mathcal{P}}^l}(v_2^l)))$, alors v_1^l et v_2^l sont placés dans la même communauté dans la partition projetée $\bar{\mathcal{P}}^l$. Cette règle, illustrée et commentée dans la figure 3.1, permet à elle seule de définir la projection. A la fin de l'expansion, dans la partition $\bar{\mathcal{P}}^0$, seules les super-communautés définies au plus haut niveau $L-1$ sont conservées en plaçant dans chacune d'entre elles tous les nœuds qui sont dans des sous-communautés.

3.2 Validation expérimentale

Pour mesurer l'impact de la troisième phase de raffinement, nous avons soumis aux algorithmes Louvain et Louvain+ les 20 graphes artificiels LFR et les 22 graphes réels présentés dans le chapitre Protocole d'expérimentation. Pour accélérer ces deux algorithmes, nous avons adopté comme critère d'arrêt de l'heuristique VM (voir section 1.4.1) non pas l'épuisement des déplacements de nœud augmentant la modularité, mais une précision ϵ_c sur la modularité obtenue en première et deuxième phase. Si à l'issue d'un examen de tous les nœuds par VM, l'aug-

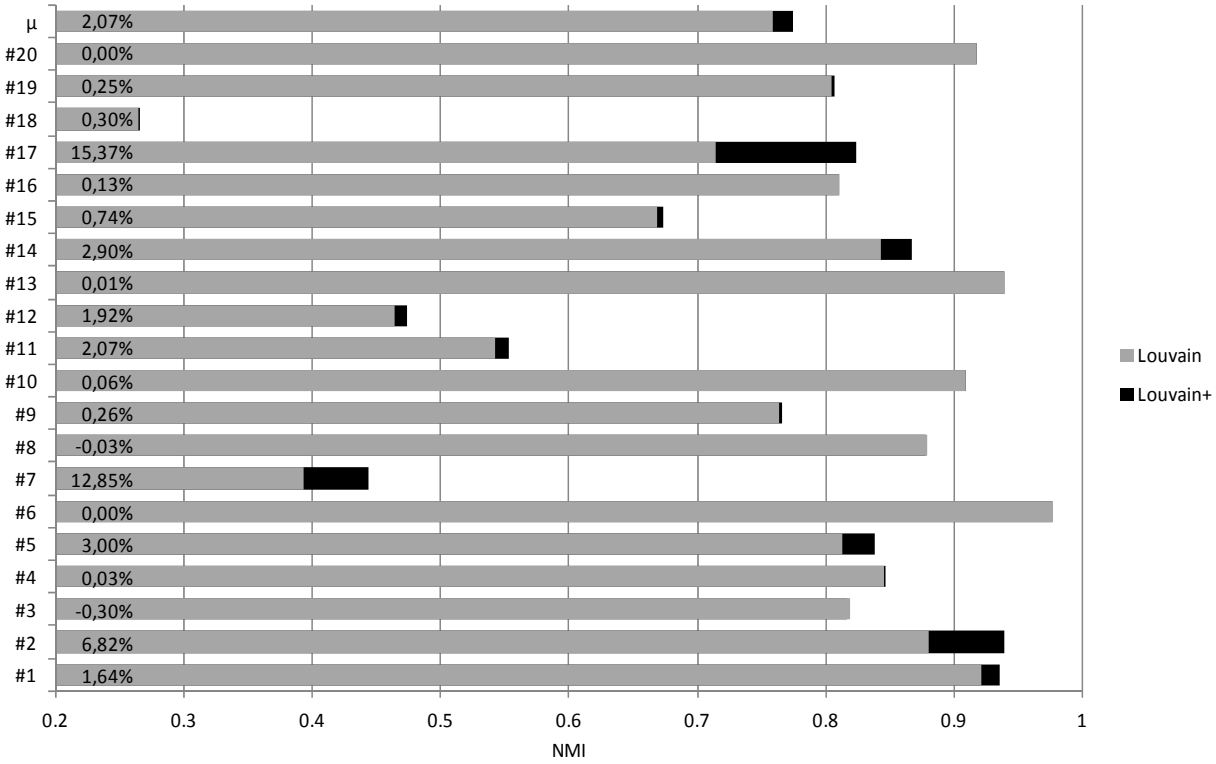


FIGURE 3.3 – NMI comparée entre Louvain et Louvain+ pour les graphes artificiels. Les barres représentent la moyenne, sur les 100 instances de graphe, de la NMI calculée entre la partition trouvée et la partition attendue. Sur chaque barre figure le taux de variation de la moyenne de NMI de Louvain à Louvain+.

mentation de la modularité est inférieure ou égale à ϵ_c, alors la procédure s'arrête. Ce critère est valable en première phase d'initialisation de partition et en deuxième phase de contraction, dans lesquelles VM est utilisée. Le seuil ϵ_c s'applique aux deux algorithmes qui ont en commun les deux premières phases. Nous définissons également le seuil ϵ_r qui marque la précision de VM appliquée au raffinement de la troisième phase de Louvain+. Pour tous les résultats présentés dans cette section, nous adoptons $\epsilon_c = \epsilon_r = 10^{-5}$. Nous verrons dans la section 3.4 que jouer sur ces deux paramètres permet de réduire le temps d'exécution de Louvain+ avec les mêmes résultats en terme de modularité.

3.2.1 Graphes artificiels

La figure 3.2 montre la différence de modularité obtenue par Louvain+ relativement à Louvain. Nous constatons que Louvain+ améliore la modularité moyenne pour tous les graphes, de façon sensible pour 5 d'entre eux (#2, #7, #11, #14 et #17). Le maximum calculé pour 100 instances est amélioré avec Louvain+ dans 16 cas sur 20, ce qui montre la capacité de cet algorithme à trouver de meilleures solutions en terme de modularité. L'augmentation de la modularité moyenne pour les graphes artificiels va de 8.10^{-6} à 0.05 en valeur absolue, avec une moyenne d'environ 0.006 (1.3 % en relatif).

La mesure de similarité entre partition proposée par l'algorithme et partition attendue est donnée dans la figure 3.3. Nous constatons une amélioration de NMI dans 18 cas sur 20, qui peut être significative (graphes #2, #7 et #17), et une baisse faible dans les deux autres cas. Nous pouvons aussi remarquer une performance assez médiocre de l'algorithme de Louvain, que Louvain+ n'améliore pas beaucoup, avec 14 graphes dont la NMI moyenne ne dépasse pas

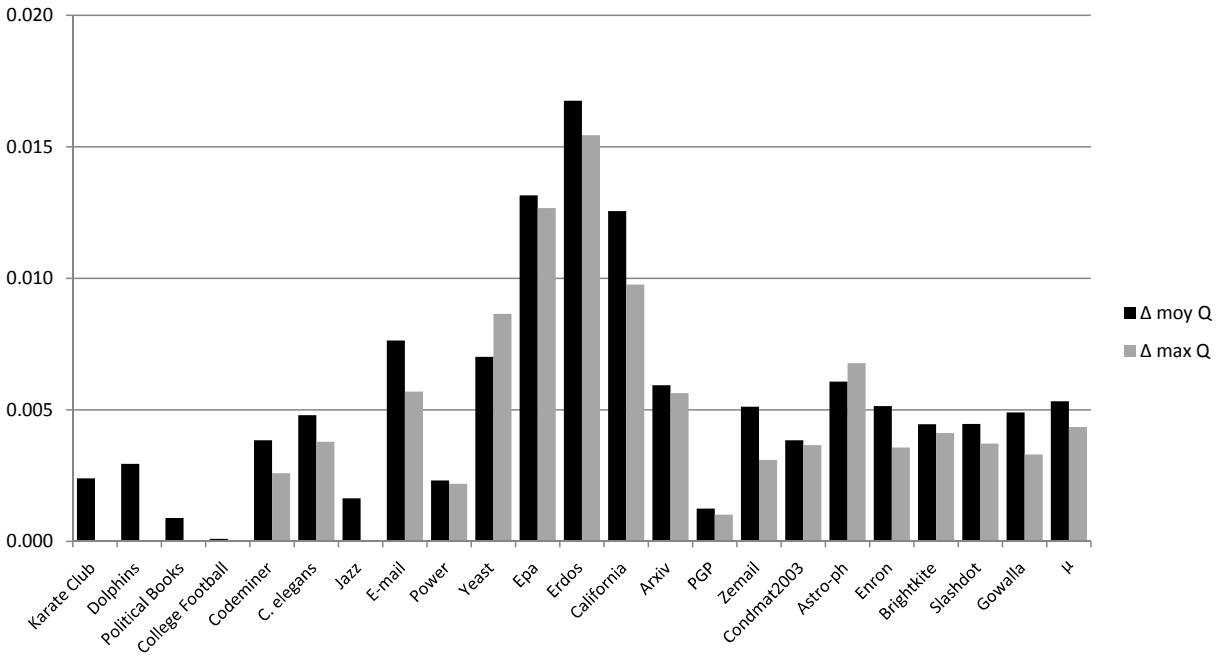


FIGURE 3.4 – Modularité comparée entre Louvain et Louvain+ pour les graphes réels. Le graphique présente, pour chaque graphe réel testé, les différences de modularité moyenne (barre noire) et maximale (barre foncée) calculée sur les 100 instances de chaque graphe, entre Louvain+ et Louvain. L'élément le plus à droite en abscisse représente la moyenne pour l'ensemble des graphes.

0.9, dont 5 graphes qui sont en dessous de 0.7.

Les améliorations constatées aussi bien pour la modularité que pour la justesse de partition mesurée par la NMI montrent une faiblesse de l'algorithme de Louvain avec les graphes de densité faible et possédant une large amplitude de taille de communauté (graphes #2, #7, #14, #17). Cette faiblesse s'explique probablement par la présence de nœuds de degré faible, mais supérieur ou égal à 2, dont l'appartenance à une communauté est incertaine. Ces nœuds peuvent être mal placés dès la première phase VM, ce placement étant conservé lors de la deuxième phase de contraction. Seule l'amélioration apportée par Louvain+ est susceptible de corriger cette erreur de placement.

3.2.2 Graphes réels

De la même manière que pour les graphes LFR, nous présentons une comparaison de modularité entre Louvain et Louvain+ sur les graphes réels du protocole d'expérimentation, dans la figure 3.4. Les résultats sont similaires, montrant une amélioration de modularité systématique apportée par Louvain+, entre 1.10^{-4} et 0.017 (moyenne d'environ 0.006).

Le score de nœud (SN), le score de communauté (SC) et le score minimal de communauté (SCM), pour les graphes réels, sont présentés sur la figure 3.5, sous la forme de boîtes à moustaches (*Box and Whiskers Plot*) qui montrent la répartition des valeurs en quatre parts de même effectif (quartiles). Nous constatons d'abord une bonne performance de SN avec une distribution étroite et élevée, entre 0.8 et 1 et une légère amélioration apportée par Louvain+. Ceci est dû au placement fin des nœuds par la projection et le raffinement. La médiane élevée de SC montre là aussi une bonne performance liée à l'optimisation de Q, mais une dispersion très étirée vers le bas dénote un problème de définition des communautés dans certains graphes, certainement

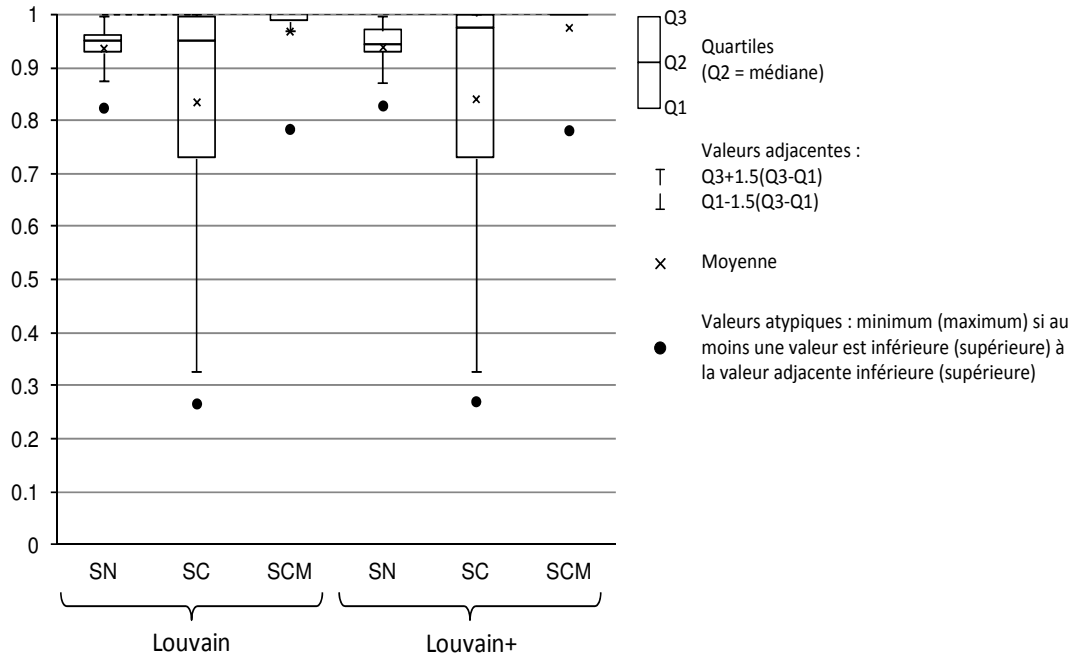


FIGURE 3.5 – Boîtes à moustaches des grandeurs SC, SN et SCM, pour Louvain et Louvain+, sur les graphes réels. Cette représentation, dont la légende est fournie à droite, permet d'apprécier la dispersion d'une grandeur mesurée (SC, SN et SCM) sur l'ensemble des graphes réels.

en raison d'une structure particulière. Toutefois, l'existence des communautés est bien affirmée par la boîte de SCM très élevée et resserrée. La aussi Louvain+ apporte une amélioration appréciable avec pratiquement toutes les valeurs très proches de 1, à l'exception du graphe *Jazz* qui donne une valeur de SCM de 0.78, pour les deux algorithmes. Ces éventuels problèmes de structures seront examinés à l'aune des algorithmes présentés par la suite, plus puissants pour l'optimisation et la justesse des partitions.

3.2.3 Temps d'exécution et complexité

Les courbes de temps d'exécution des algorithmes Louvain et Louvain+ (figure 3.6) montrent une consommation de temps légèrement supérieure pour Louvain+. L'écart entre les deux représenté par la courbe Delta augmente linéairement avec le nombre d'arêtes m.

Dans l'ensemble, ces courbes semblent confirmer la conjecture des auteurs de l'algorithme de Louvain [67], selon laquelle la complexité en temps est de l'ordre de m, le nombre d'arêtes, pour les graphes creux que l'on rencontre habituellement. Toutefois, la fonction de temps présente une légère courbure qui laisse penser qu'un autre paramètre que m intervient dans la complexité en temps, dans une très faible mesure. Selon nos observations ce paramètre intervient dans les passes de l'heuristique VM dont le nombre paraît impossible à déterminer à priori.

Selon nos tests, l'heuristique VM appliquée à plusieurs niveaux s'exécute en $O(m)$, en négligeant le paramètre que nous venons de présenter. Il faut ajouter à ce temps d'exécution le temps de construction des graphes de niveau, qui dépend de la représentation des données et de l'implémentation de l'algorithme, car la construction nécessite le calcul des poids entre paire de communautés reliées. Dans notre implémentation, nous avons choisi d'utiliser une table de hachage pour ce calcul afin de rendre le temps de construction des graphes de niveau linéaire en fonction de m. Ainsi, globalement, la quasi-linéarité selon m est préservée pour l'algorithme complet.

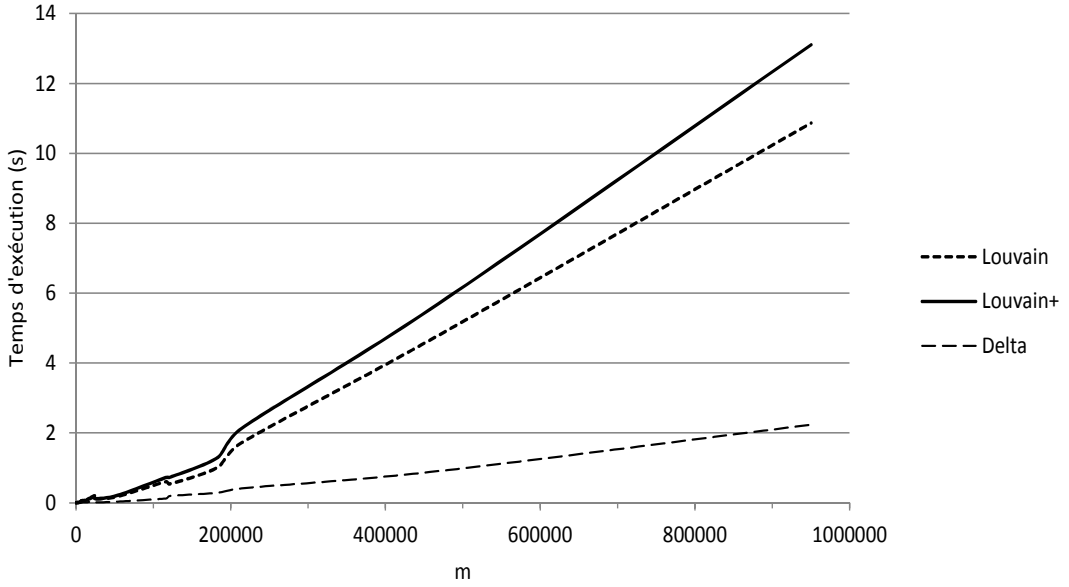


FIGURE 3.6 – Temps d'exécution de Louvain et Louvain+. Les courbes représentent le temps d'exécution moyen en secondes, calculé pour chaque graphe à partir des 100 instances, en fonction du nombre d'arêtes m.

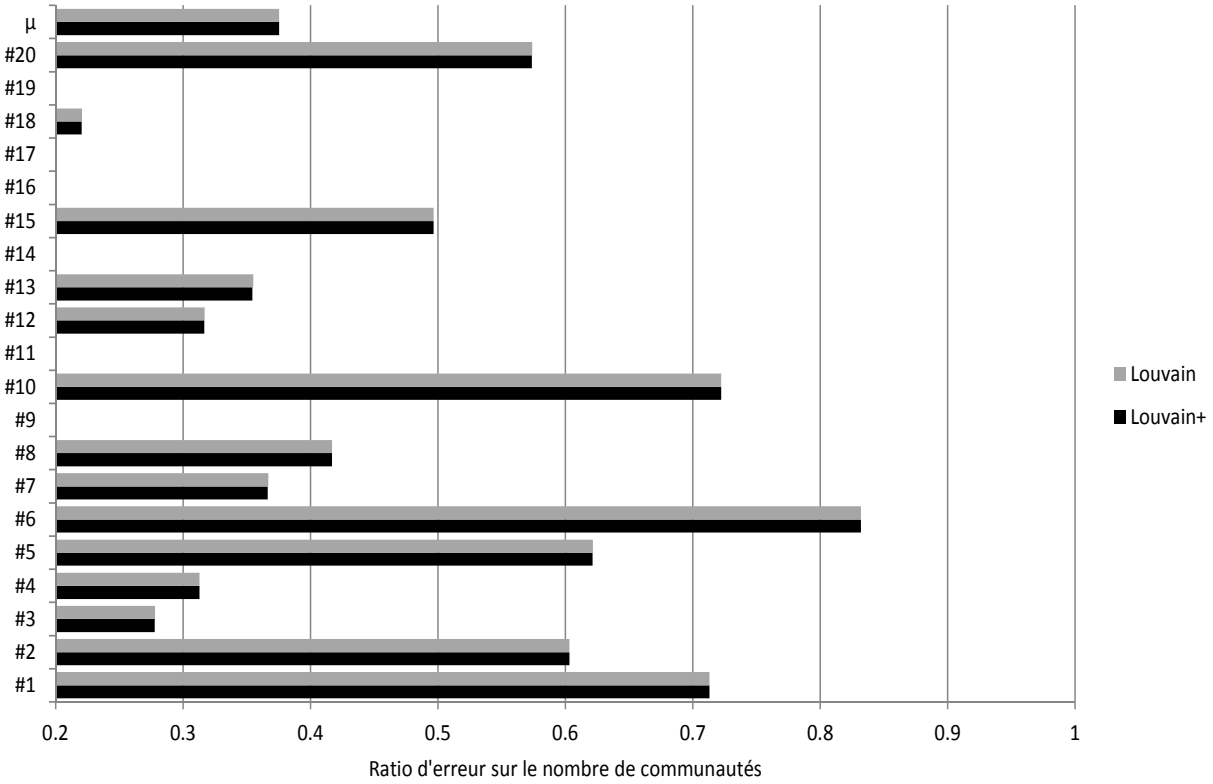


FIGURE 3.7 – Ratios d'erreur sur le nombre de communautés des graphes artificiels, comparés entre Louvain et Louvain+. Un ratio est calculé, pour chaque graphe, comme le nombre de communautés moyen sur les 100 instances divisé par le nombre de communautés de la solution. La moyenne des ratios figure en vis-à-vis du symbole μ.

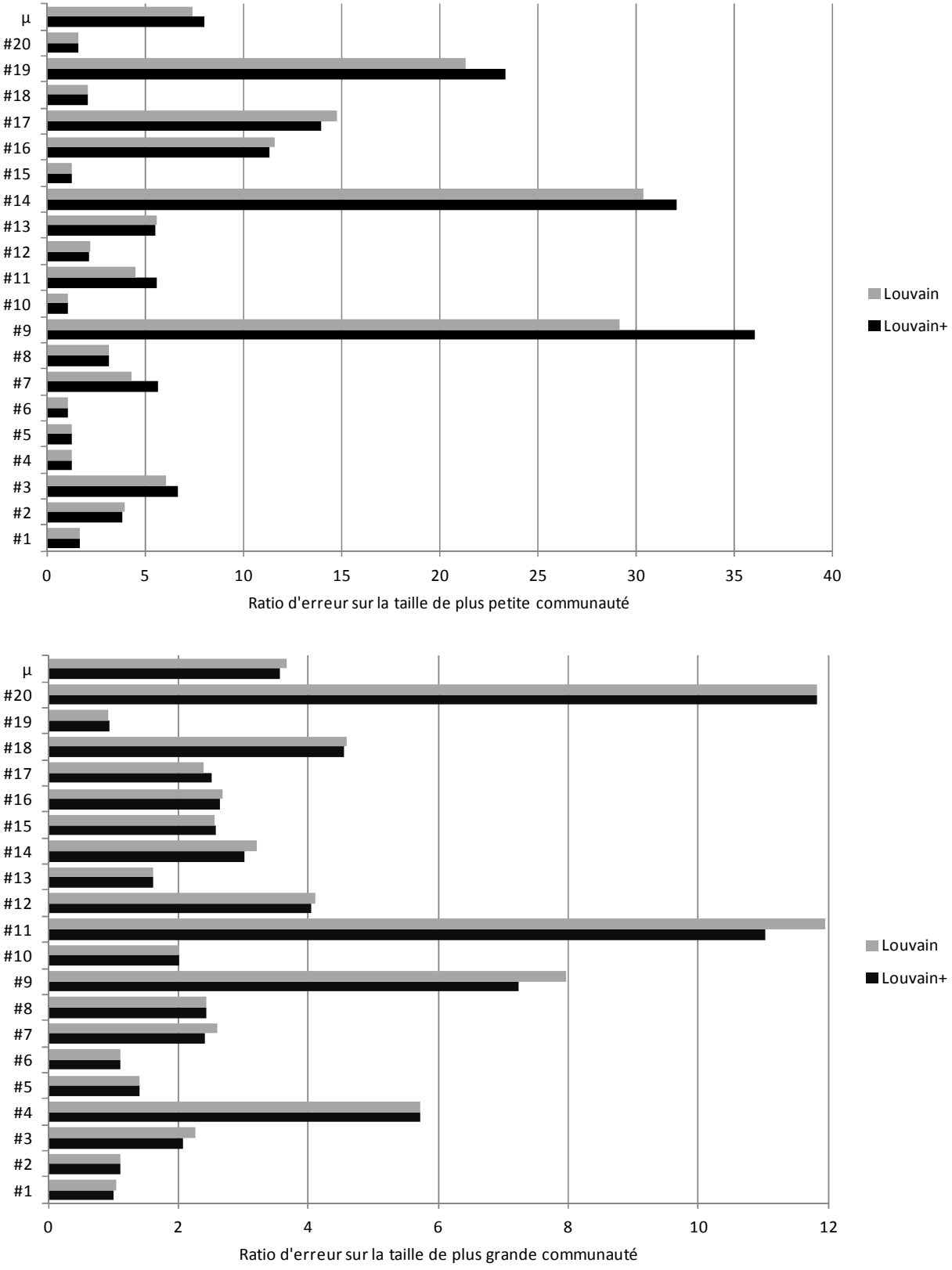


FIGURE 3.8 – Ratios d'erreur sur les tailles minimale et maximale de communauté des graphes artificiels, comparés entre Louvain et Louvain+. Un ratio est calculé, pour chaque graphe, comme la taille de plus petite (resp. grande) communauté moyenne sur les 100 instances, divisé par la taille de plus petite (resp. grande) communauté de la solution. La moyenne de ratio figure en vis-à-vis du symbole μ.

3.3 Impact sur les défauts de la modularité

3.3.1 Limite de résolution

Les graphes artificiels LFR permettent d'apprécier la pertinence des partitions proposées par un algorithme en mesurant les effets de la limite de résolution, globalement avec la NMI et plus spécifiquement on observant le nombre de communautés et les tailles de communauté extrêmes. Les résultats de similarité par la mesure de NMI présentés dans la section précédente montrent la faiblesse de l'algorithme de Louvain. Une optimisation un peu plus poussée de la modularité par Louvain+ ne présente pas d'amélioration des partitions proposées par l'algorithme en regard des solutions attendues.

Les figures 3.7 et 3.8 renseignent sur l'erreur du nombre de communautés et des tailles de plus petite et plus grande communautés par rapport à la solution associée à chaque graphe artificiel. Un ratio de 1 est idéal, au-dessus il indique une valeur fournie par l'algorithme trop grande et trop petite sinon. Ces trois graphiques sont importants et vont guider notre analyse, car ils montrent nettement les effets de la limite de résolution. En optimisant la modularité par l'algorithme de Louvain, le nombre de communautés est toujours sous-évalué dans notre jeu de test, souvent de façon excessive (pour trois quarts des graphes, le nombre trouvé est inférieur à la moitié du nombre attendu). La plus petite communauté est systématiquement trop grande, avec en moyenne une taille 7 fois plus importante que la moyenne des solutions. Enfin, la plus grande communauté est aussi systématiquement trop grande, avec en moyenne une taille trois fois supérieure. Nous devons noter également que l'algorithme Louvain+ conserve la même erreur sur le nombre de communautés, produit des petites communautés un peu plus grandes et des grandes communautés un peu plus petites. Dans l'ensemble, sur ces critères, il conserve les caractéristiques et les défauts de l'algorithme de Louvain.

Nous pouvons ici préciser le diagnostic des faiblesses de l'algorithme de Louvain, reprises par Louvain+, en observant le graphe artificiel #7. L'erreur de taille de plus grande communauté (taille deux fois plus grande) ne traduit pas l'erreur importante de justesse de partition constatée par la NMI de la figure 3.3. En observant la distribution des tailles de communauté dans une partition fournie par Louvain/Louvain+, cinq communautés ont une taille supérieure à la plus grande communauté de la solution (327), avec les tailles 388, 449, 752, 770 et 837,soit près de 62 % des nœuds qui ont un problème de communauté. Le phénomène d'erreurs dues à la densité faible de ce graphe constatées précédemment est amplifié par la constitution erronée, dès la première phase VM de Louvain, de cinq communautés trop grandes couvrant plus de 60 % des nœuds du graphe. Ces erreurs ne peuvent pas être corrigées par la seconde phase qui procède à des regroupements de communautés.

3.3.2 Inconsistance de solution

Le problème d'inconsistance de solution apparaît clairement avec les résultats de la figure 3.9. La similarité moyenne entre partitions obtenues pour un même graphe va de 0.35 à 1, avec une moyenne sur l'ensemble des graphes réels d'environ 0.76. Ces valeurs dénotent des différences structurelles importantes et révèlent une grande sensibilité à l'ordre d'examen des nœuds pour l'algorithme de Louvain. L'apport de Louvain+ sur l'inconsistance de solution est quasi nul, la distance entre solutions de même graphe étant pratiquement la même pour les deux algorithmes.

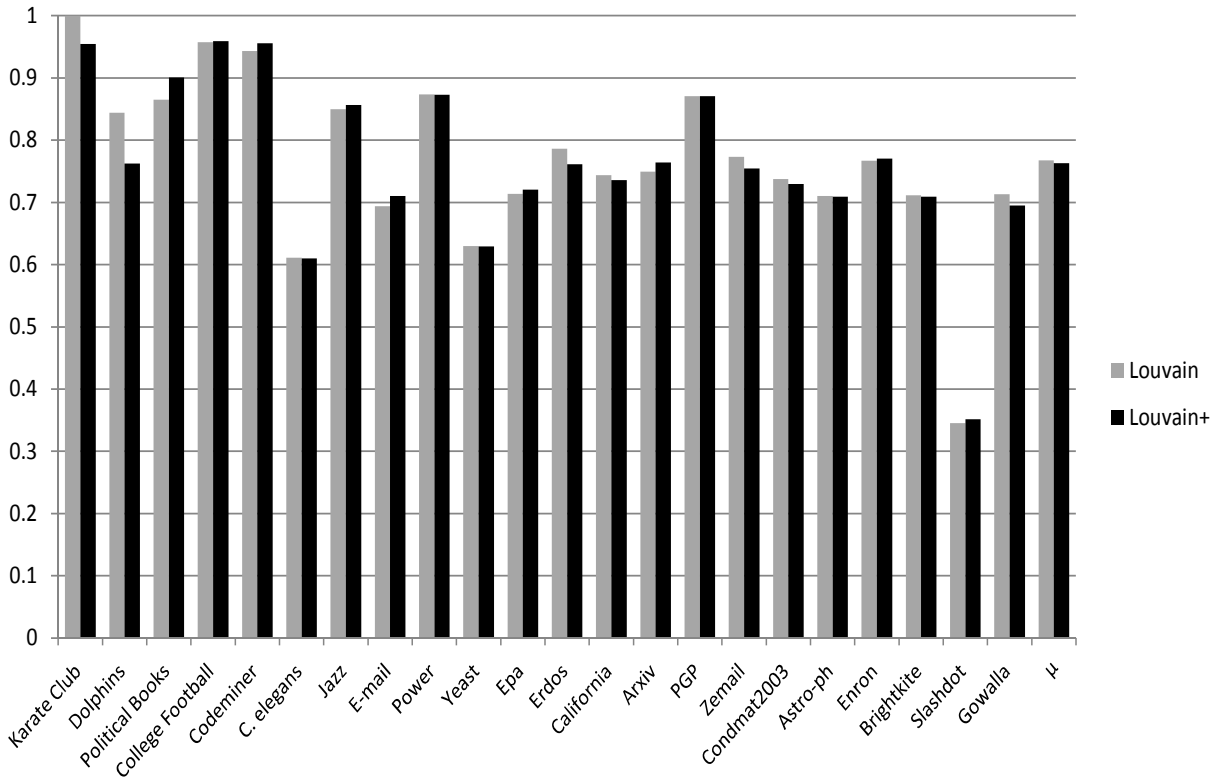


FIGURE 3.9 – Inconsistance de solution sur les graphes réels, avec Louvain et Louvain+. La version stochastique de l'algorithme est exécutée 20 fois sur chaque graphe. La similarité moyenne est calculée comme la moyenne de la NID de toutes les paires distinctes de solutions. Une similarité proche de 1 indique des solutions structurellement proches et donc une inconsistance liée à la modularité réduite.

3.4 Améliorations

3.4.1 Temps d'exécution

L'algorithme Louvain+ a un temps d'exécution légèrement supérieur à Louvain, cet écart augmentant avec le nombre d'arêtes (voir la figure 3.6). Il est possible de préserver les performances d'optimisation de Louvain+ avec un temps d'exécution pratiquement équivalent à celui de Louvain. L'idée est de jouer sur les paramètres ϵ_c et ϵ_r désignant la précision souhaitée de la procédure VM respectivement en phase de contraction et en phase d'expansion et de raffinement. Le paramètre ϵ_c peut être diminué, par exemple à 10^{-3}, pour fournir des partitions plus grossières à l'issue de la phase de contraction. Cette imprécision est compensée par un raffinement au moins aussi précis voir un peu plus, par exemple avec $\epsilon_r = 10^{-6}$. Les déplacements de nœuds à chaque niveau en phase d'expansion corrigent les erreurs de la phase précédente. Il en résulte un temps d'exécution réduit, car la première phase occupe la très grande partie du temps d'exécution.

3.4.2 Préservation de la hiérarchie

Louvain+ présente l'inconvénient de perdre la hiérarchie obtenue par le procédé multiniveau de l'algorithme de Louvain. La structure hiérarchique établie par cet algorithme, bien que sa pertinence pour un graphe donné ne soit pas établie, est intéressante pour l'étude du graphe,

ne serait-ce que pour naviguer entre plusieurs échelles d'observation. Elle est irrémédiablement perdue durant la phase d'expansion, du fait de la projection.

Nous avons développé une version capable de procéder au raffinement sans détruire la hiérarchie issue de la contraction. Pour cela, une partition est modélisée avec son graphe sous la forme d'un unique graphe composé de deux types de nœud : des nœuds du graphe d'origine appelés sommets et des nœuds de communauté. De même, deux types d'arc existent : les arêtes entre sommets issues du graphe d'origine et les arcs de parenté entre un nœud quelconque et son "parent", un nœud de communauté, désignant sa communauté d'appartenance. Ces liens sont orientés, car la relation de parenté n'est pas symétrique. Le sous-graphe formé des seuls arcs de parentés est un arbre qui constitue l'ossature de la hiérarchie.

Dans la phase de contraction, à chaque niveau, des nœuds de communauté sont ajoutés et reliés par parenté à des nœuds de niveau immédiatement inférieur. La phase de raffinement est réalisée en une seule passe en adaptant la procédure VM. Tout d'abord la relation d'inclusion ou d'appartenance est définie, par récurrence, de sorte qu'un nœud quelconque v appartient à un nœud de communauté v', on note $v \subseteq v'$, si et seulement si $v = v'$ ou $\mu(v) \subseteq v'$. Cela signifie que, si v est distinct de v', v est dans la communauté v' par transitivité ou autrement dit que v' est un ancêtre de v. Il en découle une relation de voisinage généralisée entre deux nœuds quelconques v_1 et v_2 : v_1 est voisin de v_2, noté $v_1 \ominus v_2$, si et seulement si soit $\{v_1, v_2\} \in E$ si v_1 et v_2 sont des nœuds d'origine, soit dans le cas contraire si $\exists v'_1 \subseteq v_1$ et $\exists v'_2 \subseteq v_2$ tels que $v'_1 \neq v'_2$ et $v'_1 \ominus v'_2$. Cela signifie qu'au moins un sommet descendant de v_1 ou v_1 lui-même est voisin dans le graphe d'origine d'un descendant de v_2 ou de v_2 lui même.

Pour un nœud v quelconque examiné par VM, sommet ou communauté, la liste des voisins de plus haut niveau, c'est-à-dire sans parent, est établie. Du point de vue strict de la valeur de modularité, le placement dans une communauté voisine de plus haut niveau détermine la nouvelle valeur de modularité, peu importe dans quelle sous-communauté est effectivement placé v. Comme dans l'heuristique VM classique, le déplacement qui augmente le plus Q, parmi tous les voisins de v, est choisi et exécuté. Il demeure un problème relatif à la structure interne de la communauté cible C. Pour préserver une hiérarchie qui ait du sens, le nœud v ne peut pas être placé n'importe où dans C. Il doit logiquement être placé dans la plus petite sous-communauté de C contenant tous les sommets de C reliés à v. Cette règle garantie que v est placé dans la bonne sous-communauté. Ce mécanisme est illustré sur la figure 3.10, à partir de l'exemple précédent.

Précisons par ailleurs que ce raffinement où toutes les communautés sont en compétition pour les déplacements donne une optimisation très légèrement supérieure à celle de Louvain+ où les raffinements sont réalisés par niveau.

Conclusion

Nous avons proposé une amélioration de l'algorithme de Louvain qui ajoute une phase d'expansion et de raffinement susceptible de mieux placer certains nœuds en augmentant la modularité. Par construction, cette phase ne peut pas diminuer la modularité. Elle occasionne en moyenne un surplus de temps d'exécution d'environ 22% en préservant la complexité en temps de l'algorithme de Louvain, en $O(m)$ pour des graphes peu denses. La modularité est faiblement augmentée (environ 1%) et les scores de communautés sont améliorés, particulièrement SC et SCM, renforçant ainsi l'existence des communautés. L'impact sur les défauts de la modularité est minime, mais la justesse des partitions de graphes artificiels est accrue significativement, avec une NMI augmentée dans 18 cas sur 20, la différence allant de -0.3 % à 15.4 %.

Une version améliorée de Louvain+ produit les mêmes performances avec un temps d'exécution qui n'augmente pas par rapport à Louvain et préserve de plus la hiérarchie des com-

munautés. L’algorithme Louvain+ a pour objet de corriger les quelques erreurs de placement de nœud laissées par Louvain, sans aggravation du temps d’exécution. Il peu avantageusement remplacer ce dernier.

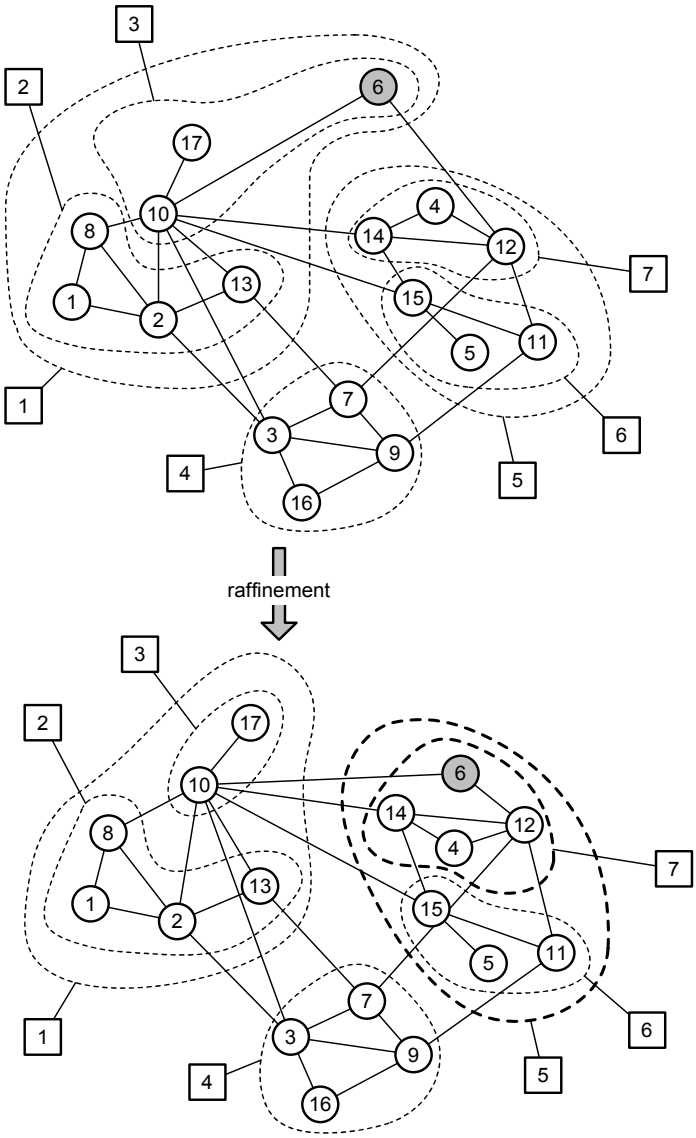


FIGURE 3.10 – Illustration du raffinement de Louvain+ qui préserve la hiérarchie. Le graphe hiérarchique issu de la phase de contraction est présenté en haut avec 7 communautés, dont 3 de plus haut niveau. Seule la composition de ces dernières communautés détermine la valeur de modularité. Cette partition est équivalente, à la hiérarchie près, à celle de la figure 3.1. L'amélioration est la même et consiste à déplacer le nœud 6 dans la communauté n° 5. Cette communauté est composée de deux sous-communautés, mais seule la n° 7 possède des liens avec le nœud 6, il est donc logique de placer ce nœud dans la communauté n° 7 et non dans la n° 5. Notons que la différence avec le raffinement "à plat" de Louvain+ est qu'un nœud de communauté, par exemple le n° 2, peut être déplacé. Dans cet exemple, le seul déplacement du nœud n° 2 qui peut impacter la modularité est vers la communauté 4. Dans ce cas, deux possibilités de hiérarchie existent, pour une même valeur de modularité, soit placer la communauté n° 2 dans la communauté n° 4, soit réunir les deux dans une nouvelle communauté de plus haut niveau. Nous avons opté pour cette seconde solution, ce choix n'influant que sur la hiérarchie obtenue à la fin et non sur la modularité maximale.

4

Algorithme mémétique

Sommaire

Nous exposons ici, dans le dernier chapitre consacré à la pure optimisation de la modularité, une adaptation originale du principe d'algorithme mémétique à la détection de communautés. Cet algorithme de type évolutionnaire, utilise une population de partitions croisées selon deux techniques et améliorées par l'algorithme Louvain+. L'une de ces techniques de croisement, inédite, est une contribution de cette thèse, l'autre a récemment été développée dans notre laboratoire. Cet algorithme atteint des valeurs de modularité inédites, pour tous les graphes testés. Les résultats obtenus confirment le diagnostic de corrélation entre l'optimisation extrême de la modularité et la limite de résolution. Le second apport de cet algorithme est la très forte réduction de l'inconsistance de solution, car les optimums atteints sont rares.

Ce chapitre a été présenté aux conférences *Modèles et Analyse des Réseaux (MARAMI-2011)* et *Parallel Problem Solving (PPSN XII)* et publié dans leurs actes.

Introduction

Les algorithmes génétiques, et plus encore mémétiques, sont connus pour traiter efficacement de nombreux problèmes d'optimisation difficiles [96]. Les premiers utilisent un ensemble de solutions initiales qu'ils font évoluer par croisement et mutation dans le but de sélectionner les meilleures solutions selon un critère d'optimisation. Les seconds reprennent ce principe en y adjoignant une optimisation locale qui améliore les nouvelles solutions. Les applications à la détection de communautés de cette classe d'algorithmes dits évolutionnaires ne rivalisent pas avec les meilleurs algorithmes dédiés pour optimiser la modularité [62, 63, 64, 70]. En général, l'opération de croisement est essentielle à l'efficacité de l'algorithme. Nous présentons un algorithme mémétique original pour la détection de communautés, selon deux versions qui diffèrent seulement par le croisement. L'une utilise un croisement que nous avons conçu et qui cherche à préserver des communautés entières. L'autre, récemment développé dans notre laboratoire, s'attache à conserver les plus grandes parties de communautés communes aux deux parents. L'algorithme Louvain+ est utilisé pour améliorer toutes les partitions obtenues par croisement. Ces versions d'algorithme mémétique amènent à des niveaux d'optimisation de la modularité jamais atteints.

4.1 Description de l'algorithme

4.1.1 Principe général

Typiquement, un algorithme mémétique combine un opérateur de recombinaison ou croisement et une optimisation locale appliqués sur un ensemble de solutions appelé population. Le croisement a pour objectif de produire de nouvelles solutions en couvrant potentiellement tout l'espace de recherche. Il opère une diversification de la population en étant susceptible d'atteindre des zones non explorées de l'espace des solutions. L'optimisation locale améliore une nouvelle solution produite par croisement en recherchant un optimum local aux alentours de celle-ci.

Le mécanisme de l'algorithme (voir le pseudo-code de l'algorithme 2) est le suivant :

1. Génération d'un nombre déterminé de partitions en exécutant la version stochastique de Louvain+ et placement de ces partitions dans la population P.

2. Croisement de deux partitions de P choisies au hasard pour obtenir une nouvelle partition I.

3. Application de Louvain+ à la partition I pour optimiser sa modularité.

4. Mise à jour de la population avec la partition I qui, si elle est sélectionnée, prend la place d'une partition de la population jugée moins intéressante.

5. Répétition des étapes 2 à 4 jusqu'à la satisfaction d'un critère d'arrêt.

Nous nommons la succession des opérations 2 à 4, qui conduisent à la production d'une nouvelle partition, un tour. A l'issu d'un tour, la partition de plus grande modularité dans la population est jugée la meilleure partition et sera la solution proposée par l'algorithme une fois son exécution achevée. Les quatre opérations fondamentales, l'initialisation, le croisement, l'optimisation et la mise à jour de P, sont décrites ci-dessous. Nous avons adopté comme critère d'arrêt un nombre t_{max} de tours successifs sans nouvelle meilleure partition, à ne pas dépasser.

Algorithme 2 Pseudo-code de l'algorithme MA-COM

Require: Graphe $G = (V, E)$.
Ensure: Une partition I^* de G avec une modularité maximale.

$P = \{I^1, I^2, ..., I^{|P|}\} \leftarrow$ InitialiserPopulation()
$I^* = \arg\max_{I \in P}\{Q(I)\}$ /* enregistre la meilleure partition */
$t = 0$ /* compteur du nombre de tours sans amélioration */
repeat
  $(I^i, I^j) \leftarrow$ ChoisirParents(P)
  $I \leftarrow$ CroiserParents(I^i, I^j)
  $I \leftarrow$ Louvain+(I) /* améliore la partition avec Louvain+ */
  if $Q(I) > Q(I^*)$ **then**
    $I^* \leftarrow I$
    $t \leftarrow 0$
  else
    $t \leftarrow t + 1$
  end if
  $P \leftarrow$ MettreAJourPopulation(I, P, d_{min})
until $t \geq t_{max}$

4.1.2 Population initiale

La population est formée d'un ensemble d'"individus" qui sont différentes solutions du problème et qui vont évoluer par croisement et optimisation. Sa taille est constante et fixée arbitrairement au départ. Généralement, quelques dizaines d'individus dans la population suffisent à rendre opérant un algorithme génétique ou mémétique, à condition que la diversité des individus soit maintenue. Cette taille, notée $|P|$ constitue un paramètre important de l'algorithme.

L'algorithme Louvain+, dans sa version stochastique qui produit à chaque exécution une partition différente, est parfaitement approprié pour initialiser les solutions de la population. D'une part en raison de sa complexité linéaire permettant d'obtenir rapidement un ensemble de partitions. D'autre part en raison sa capacité à produire des partitions de bonne qualité (modularité assez élevée compte tenu de la faible complexité) et diversifiées. La distance importante entre partitions de différentes exécutions, perçue comme une faiblesse des algorithmes Louvain et Louvain+, devient ici un avantage.

4.1.3 Croisement par préservation de communautés

Par analogie avec la reproduction sexuée et la théorie de l'évolution, le croisement d'un algorithme évolutionnaire opère à partir de deux parents pris dans la population pour engendrer une progéniture, c'est-à-dire une nouvelle solution du problème traité. Ce croisement, que nous avons conçu, se fonde sur l'idée de préserver certaines communautés des deux parents selon un mécanisme de priorités.

Deux solutions, notées $\mathcal{P}_1$ et $\mathcal{P}_2$ sont choisies au hasard dans la population. Les communautés des deux parents sont numérotées de 1 à $k_1 + k_2$, k_1 (resp. k_2) étant le nombre de communautés du premier (resp. second) parent. Chacune de ces communautés possède une priorité propre désignée par un nombre allant de 1 (haute priorité) à $k_1 + k_2$ (basse priorité), selon une attribution aléatoire. Ceci étant posé, le croisement procède dans l'ordre de priorité des communautés. La communauté de plus haute priorité est intégralement préservée dans la nouvelle solution. La seconde est préservée à l'exception des nœuds déjà affectés, c'est-à-dire des nœuds placés dans la communauté précédente, et ainsi de suite jusqu'à la communauté de priorité la plus faible. Dans la partition résultante, les communautés sont finalement renumérotées en commençant

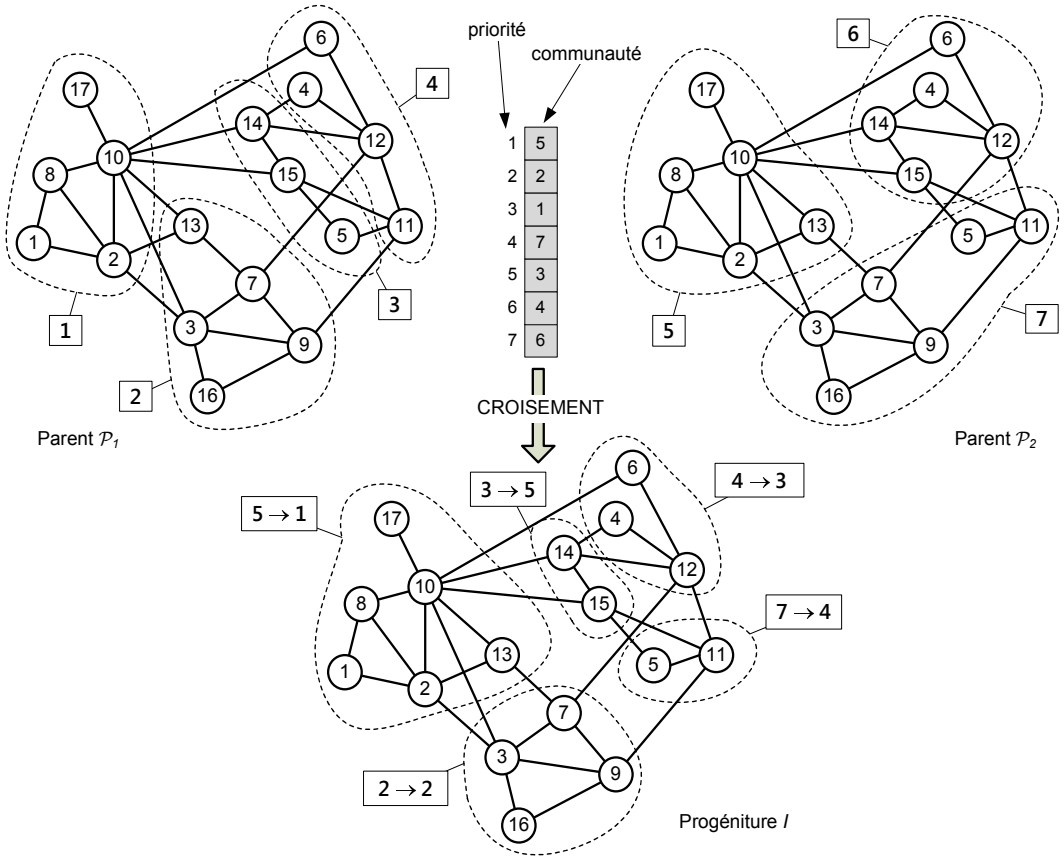


FIGURE 4.1 – Illustration de l'opérateur de croisement par préservation de communautés. La communauté n° 5 a la plus haute priorité, elle est intégralement préservée et deviendra la communauté n° 1 dans la progéniture. La communauté n° 2, qui vient du premier parent (à gauche) suit dans l'ordre des priorités. Tous ces nœuds sont conservés à l'exception du n° 13 qui est déjà placé, et ainsi de suite. La flèche dans les légendes de communauté de la progéniture indique la renumérotation des communautés.

par 1. Ce mécanisme assure la conservation de certaines communautés, éléments constitutifs de chaque parent, et oblige la scission d'autres communautés. Selon le paradigme des algorithmes génétiques, il produit un nouvel individu, possiblement dans une nouvelle zone prometteuse de l'espace de recherche, du fait de la scission et en même temps, il reprend certaines briques de base des parents qui doivent être bien adaptées du fait de la sélection (voir figure 4.1).

Ce mécanisme ne fonctionne qu'à la condition que les parents ne soient pas trop proches structurellement. Cette exigence est assurée par une fonction de distance utilisée à la mise à jour de la population. De plus, nos tests ont montré qu'une attribution aléatoire des priorités est aussi efficace qu'un choix orienté par la valeur de modularité. Enfin, le choix aléatoire des parents est préférable à un choix orienté.

4.1.4 Croisement par intersection de communautés

Ce croisement est emprunté à Qinghua Wu, Una Benlic et Jin-Kao Hao [97, 98] et s'appuie sur l'idée qu'il est judicieux de préserver dans la progéniture les plus grands sous-ensembles de nœuds appartenant à la même communauté dans les deux parents. Le principe est simple. Deux nœuds quelconques v_1 et v_2 du graphe à partitionner se retrouve dans la même communauté dans la progéniture si et seulement si ils sont reliés par une arête et s'ils sont dans la même communauté dans chacun des parents. Dans les cas contraires, les deux nœuds doivent être

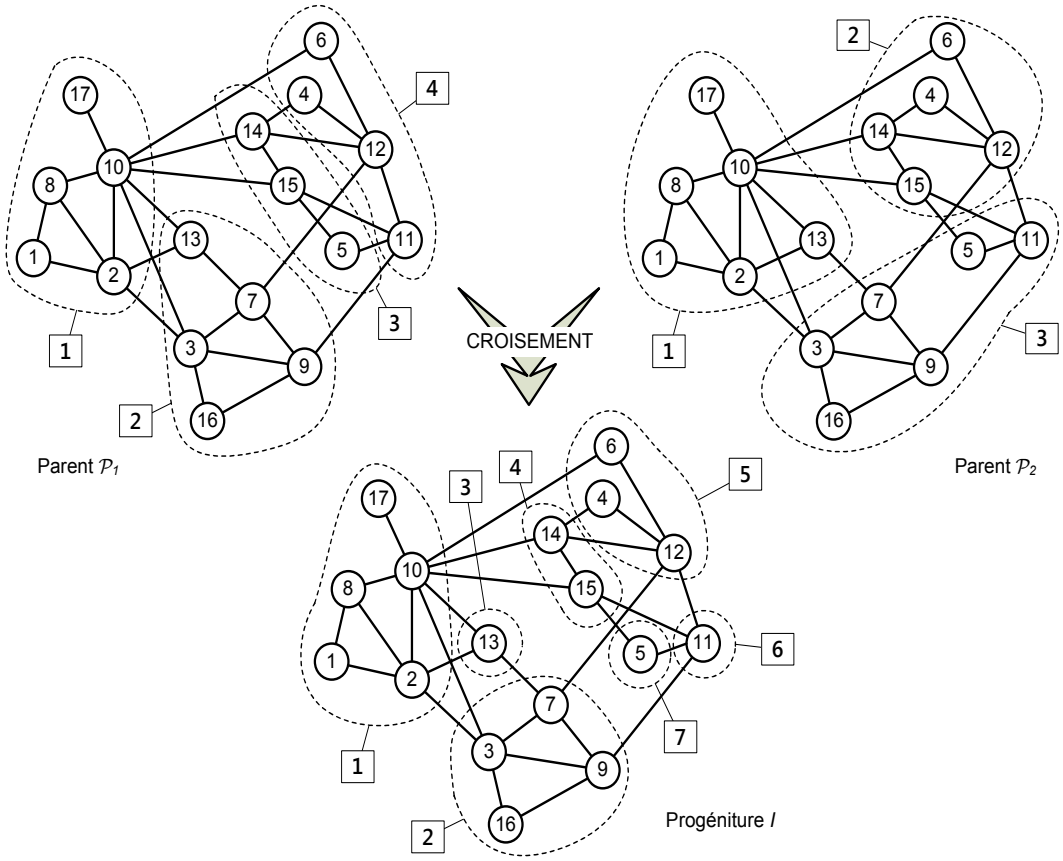


FIGURE 4.2 – Illustration de l'opérateur de croisement par intersection de communautés. Chaque communauté de la progéniture est le plus grand sous-graphe connexe ne contenant que des nœuds de même communauté dans chaque parent. Par exemple les nœuds 1, 2, 8, 10 et 17 sont dans la communauté n° 1 dans $\mathcal{P}_1$ et dans la communauté n° 1 dans $\mathcal{P}_2$. Aucun autre nœud relié à l'un de ces nœuds et vérifiant ces propriétés ne peut être ajouté à la liste. Ainsi, $\{1, 2, 8, 10, 17\}$ forme une communauté dans la progéniture. De même, le nœud n° 13 n'a aucun voisin de même communauté à la fois dans $\mathcal{P}_1$ et $\mathcal{P}_2$. Il est donc seul dans une communauté de I.

dans des communautés distinctes.

Une autre façon de décrire ce principe passe par la notion d'ensemble. Un ensemble de nœuds C est une communauté de la progéniture si C est inclus dans une communauté de $\mathcal{P}_1$ et dans une communauté de $\mathcal{P}_2$, et si le seul ensemble contenant C et vérifiant cette propriété est C lui-même (autrement dit C est maximal par l'inclusion). Ce mécanisme est illustré sur la figure 4.2.

4.1.5 Optimisation locale

Chaque progéniture produite par croisement est optimisée par la simple application de l'algorithme Louvain+. Pour être précis, les algorithmes Louvain et Louvain+ démarrent avec une partition singleton (un nœud par communauté), applique l'heuristique VM puis procède à la succession des contractions et enfin des raffinements pour Louvain+. Ici, l'heuristique VM de départ s'applique directement à la partition à optimiser. Louvain/Louvain+ peuvent donc servir à constituer une partition à partir d'un graphe, mais aussi à optimiser une partition déjà constituée.

4.1.6 Mise à jour de la population

Nous utilisons comme fonction de distance une forme simplifiée de Rand Index [92], limitée aux arêtes du graphe et non à tous les couples de nœuds. La complexité passe de n^2 à m, ce qui est intéressant pour des réseaux complexes peu denses ($m \ll n^2$). Le calcul est simple : la valeur 0 est associée à chaque arête dont les extrémités appartiennent à la même communauté dans les deux partitions ou appartiennent à des communautés différentes dans les deux partitions. Si elles appartiennent à la même communauté dans une partition et à des communautés différentes dans l'autre, l'arête reçoit la valeur 1. La valeur de distance est la somme de ces valeurs pour toutes les arêtes, divisée par le nombre total d'arêtes m. Nous avons démontré que cette fonction est une distance au sens mathématique à la condition que toutes les communautés soient des sous-graphes connexes, c'est-à-dire qu'il n'y ait pas un nœud sans voisin interne à sa communauté. La modularité s'oppose à cette disposition, un partitionnement n'étant pas optimal si une communauté n'est pas connexe. Un déplacement de nœud avec VM corrige cette situation sous-optimale. En revanche, le croisement peut produire une telle partition, mais l'optimisation qui suit l'empêche, de sorte que cette condition de distance mathématique est toujours vérifiée au moment de la mise à jour. Ainsi, cette fonction de distance garantit deux propriétés importantes :

1. toutes les paires d'individus proches structurellement ont une distance proche de zéro ou, dit autrement, toutes les paires d'individus dont la distance s'éloigne de zéro ne sont pas proches structurellement : par cette propriété, un nouvel individu inséré dans la population, donc de distance éloignée de tous les individus présents, n'est pas ajouté par erreur (propriété essentielle qui garantit la diversité de la population) ;

2. toutes les paires d'individus de distance proche de zéro sont proches structurellement : par cette propriété, un nouvel individu ne sera pas écarté par erreur parce qu'il a une distance quasi nulle avec un individu existant dans la population (propriété importante qui garantit que la distance n'est pas trop excluante).

Pour une nouvelle solution I, la distance minimale $\mathcal{D}(I)$ avec les solutions actuelles de la population est calculée. Si $\mathcal{D}(I) > d_{min}$, d_{min} étant une "distance de sécurité", la solution est insérée dans la population en lieu et place de la solution de plus basse modularité. Dans le cas contraire, la solution remplace l'individu le plus proche, uniquement si sa modularité est supérieure. Dans ce dernier cas, la contrainte de distance est relâchée, car elle n'impose pas que le nouvel individu soit distant d'au moins d_{min} avec *tous* les individus de la population. En effet, la solution remplace un individu trop proche (distance entre eux inférieure à d_{min}), mais il est possible qu'un autre individu de la population soit également trop proche de la solution insérée. Nos tests ont montré qu'une condition stricte, qui impose à tout instant une distance minimale entre deux individus quelconques de la population, allonge le temps de calcul sans améliorer la convergence.

4.2 Validation expérimentale

Nous avons testé deux versions de notre algorithme mémétique, l'une nommée MA-COM/PC qui utilise le croisement par préservation de communautés, et l'autre nommée MA-COM/IC qui s'appuie sur le croisement par intersection de communautés. Ces tests portent sur les 15 plus petits graphes artificiels et les 18 plus petits graphes réels (de *Karate Club* à *Astro-ph*) en laissant de côté les plus grands pour conserver un temps raisonnable d'exécution. Les algorithmes génétiques et mémétiques fournissent généralement une optimisation sans égale pour des problèmes difficiles, mais au prix d'un temps d'exécution qui peut devenir prohibitif pour des problèmes de grande taille.

L'algorithme Louvain+, utilisé dans l'algorithme mémétique, adopte les paramètres des tests du chapitre 3, soit $\epsilon_c = \epsilon_r = 10^{-5}$. Les paramètres spécifiques de l'algorithme sont :

- $|P|$, la taille fixe de population ;
- d_{min}, la distance minimale pour insérer une nouvelle solution dans la population ;
- t_{max}, le nombre maximal de tours sans amélioration de la modularité, qui constitue le critère d'arrêt de l'algorithme.

Le dernier paramètre n'est pas crucial, mais doit être suffisamment grand pour pousser l'optimisation assez loin. Les algorithmes évolutionnaires ont en général une grande capacité d'optimisation, mais évidemment sans garantie d'atteindre l'optimum global et surtout sans garantie de la vitesse de convergence, l'algorithme procédant par saut dans des régions de l'espace de recherche, différentes, mais pas toujours prometteuses. Nos tests préliminaires nous ont permis d'établir une valeur satisfaisant de $t_{max} = 100$ qui sera utilisée pour tous les résultats présentés. Les deux premiers paramètres sont importants et font l'objet d'une expérimentation spécifique.

4.2.1 Test des paramètres

Notre objectif n'est pas de réaliser une étude fouillée de l'influence des paramètres $|P|$ et d_{min} sur les performances de notre algorithme, qui demanderait un développement sortant du cadre de cette étude. Nous souhaitons simplement réaliser un test succinct, mais significatif pour choisir des valeurs de paramètre convenables, à utiliser par la suite dans tous les tests complets.

Le test de paramètre consiste à exécuter une seule fois chaque version d'algorithme sur une seule instance des 15 premiers graphes artificiels. Bien que ce test paraisse très limité, notre algorithme présente une faible variabilité de solutions, car l'optimisation est poussée, ce qui rend significative une seule exécution pour apprécier des valeurs de paramètre. Cette exécution, résumée par une moyenne de temps et de modularité sur les figures 4.3 et 4.4, est réalisée pour chaque valeur de $|P| \in \{2, 5, 10, 20\}$ combinée avec chaque valeur de $d_{min} \in \{0, 0.1, 0.01, 0.001\}$.

Pour la version MA-COM/PC, avec le croisement par préservation de communauté, la taille de population $|P|$ est prépondérante pour l'optimisation de la modularité, le meilleur résultat étant obtenu à partir de 20 et demeurant le même jusqu'à 50. Malheureusement, le temps d'exécution croît exponentiellement avec $|P|$. Ainsi, la valeur de 20 nous paraît un très bon compromis entre performance et rapidité. La distance minimale entre individus de la population est d'autant plus prépondérante que la taille de population est réduite. Elle paraît non significative à partir de $|P| = 20$. Toutefois, ces résultats ne sont qu'indicatifs, car ils portent sur une seule instance d'exécution. Nous pensons préférable de conserver une distance, plutôt à 10^{-3}, car cette valeur donne de meilleurs résultats pour $|P| > 20$. Ainsi, pour tous les tests plus poussés qui suivent, les paramètres sont $|P| = 20$ et $d_{min} = 10^{-3}$.

La version MA-COM/IC est plus performante, en optimisation de la modularité, mais aussi en temps d'exécution. Les meilleurs résultats arrivent dès $|P| = 5$ et l'optimisation est maximale et constante pour $10 \geq |P| \geq 50$. À partir de $|P| = 5$, la distance paraît sans impact sur les performances d'optimisation, mais une distance de 10^{-2} ou 10^{-3} réduit le temps d'exécution. La convergence est plus rapide du fait d'une diversité de la population. Ici, un choix judicieux serait $|P| = 10$, mais nous choisissons la même taille que la version PC, soit $|P| = 20$, d'une part parce que les résultats précis montrent une légère meilleure optimisation (non visible sur le graphique) et d'autre part parce que cela permettra une comparaison équitable avec la version PC, en particulier sur le temps d'exécution. La distance de 10^{-3} offre une optimisation un peu plus rapide et très légèrement meilleure que l'absence de distance. Les valeurs supérieures

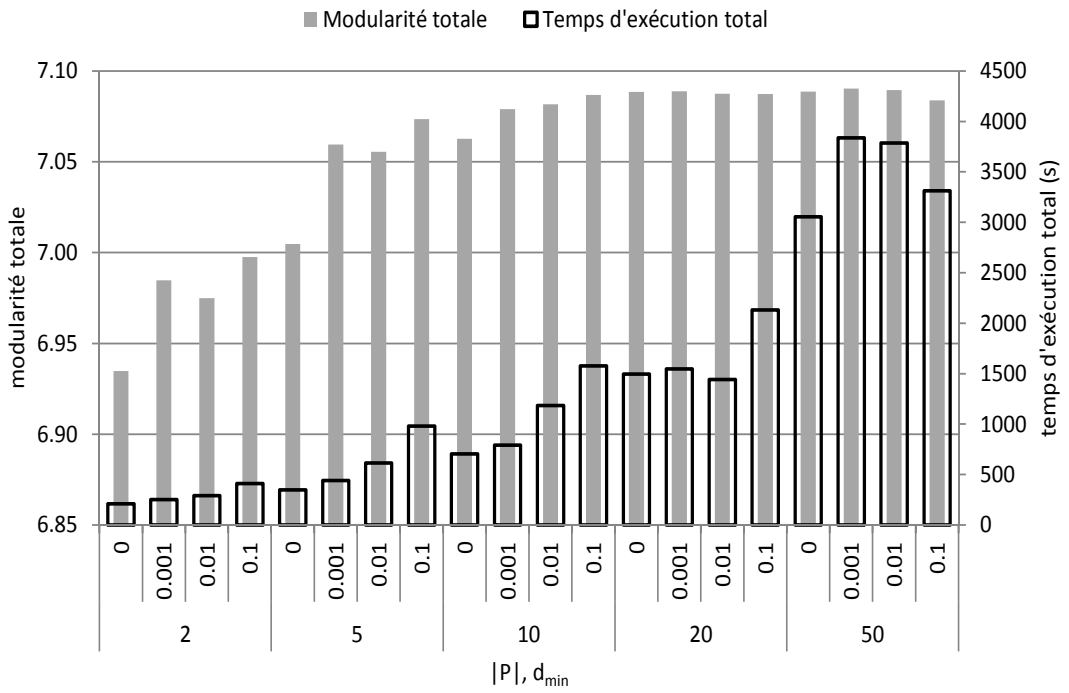


FIGURE 4.3 – Modularité et temps d'exécution de MA-COM/PC selon les paramètres $|P|$ et d_{min}. Le graphique représente la somme des modularités et la somme des temps d'exécution obtenues par une exécution de l'algorithme MA-COM/PC sur les 15 plus petits graphes artificiels, selon la combinaison des valeurs de $|P| \in \{2, 5, 10, 20\}$ et de $d_{min} \in \{0, 0.1, 0.01, 0.001\}$.

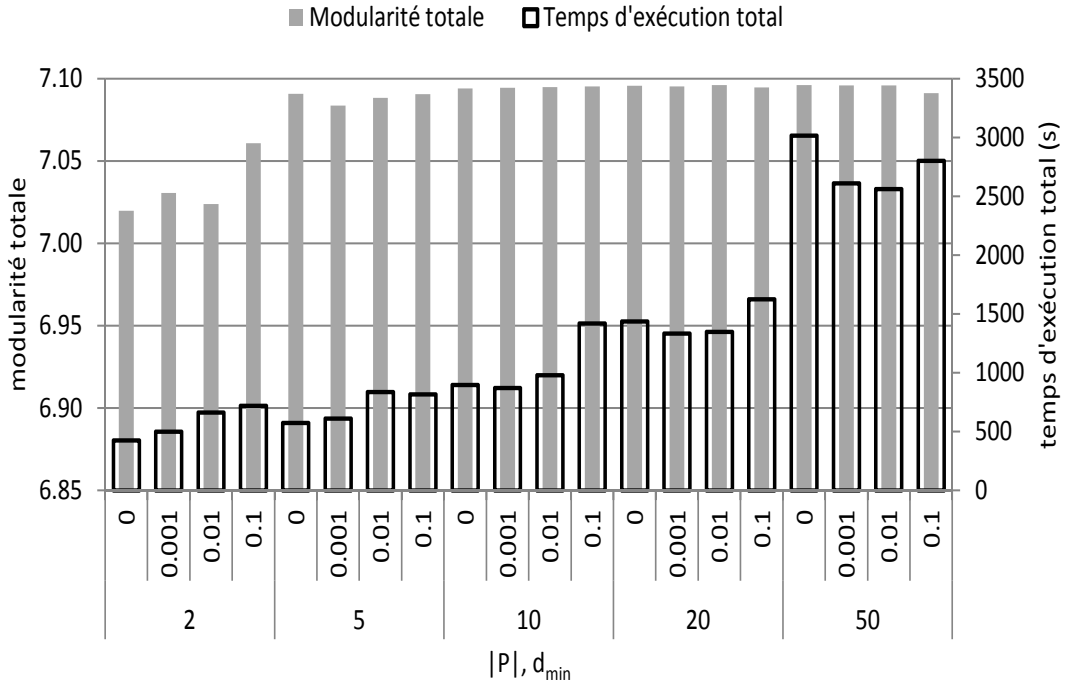


FIGURE 4.4 – Modularité et temps d'exécution de MA-COM/IC selon les paramètres $|P|$ et d_{min}. Le graphique représente la somme des modularités et la somme des temps d'exécution obtenues par une exécution de l'algorithme MA-COM/IC sur les 15 plus petits graphes artificiels, selon la combinaison des valeurs de $|P| \in \{2, 5, 10, 20\}$ et de $d_{min} \in \{0, 0.1, 0.01, 0.001\}$.

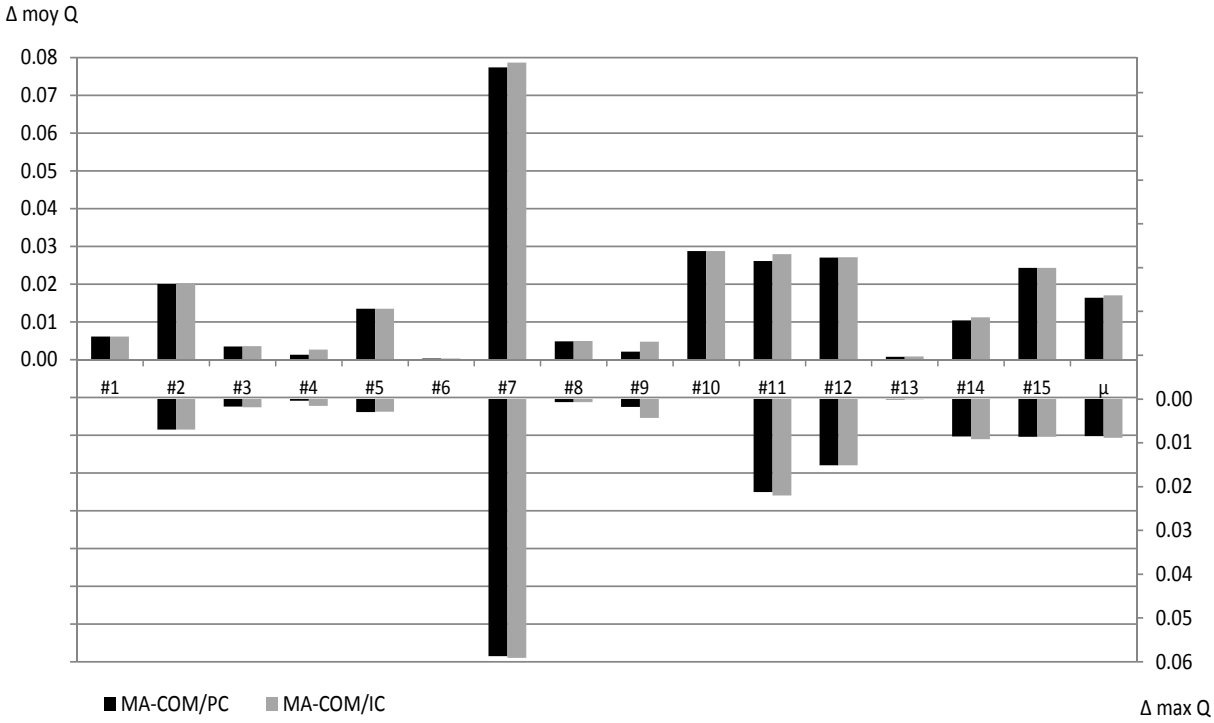


FIGURE 4.5 – Modularité comparée entre les deux versions de MA-COM et Louvain+ pour les graphes artificiels. Le graphique présente, pour chaque graphe LFR testé, les différences de modularité moyenne (partie haute, axe des ordonnées de gauche) et maximale (partie basse à l'envers, axe des ordonnées de droite) calculées sur 20 itérations, entre MA-COM (versions PC et IC, paramètres $|P| = 20$ et $d_{min} = 10^{-3}$) et Louvain+ (résultats de la validation expérimentale de cet algorithme, section 3.2).

de distance n'apportent rien en optimisation et augmente le temps d'exécution significativement. Ainsi, pour tous les tests plus poussés qui suivent, les paramètres des deux versions sont $|P| = 20$ et $d_{min} = 10^{-3}$.

4.2.2 Graphes artificiels

La comparaison entre MA-COM et Louvain+, du point de vue de la modularité, est présentée sur la figure 4.5. Les barres représentent l'augmentation de modularité constatée avec MA-COM par rapport à Louvain+. Le gain est systématique et significatif avec une moyenne de près de 0.02. L'amélioration est constatée aussi bien pour la moyenne que pour la modularité maximale sur les 20 instances d'exécution. Relativement aux valeurs de modularité, le gain peut sembler faible, mais le saut est très important, car l'algorithme mémétique permet de dépasser les très nombreux optimums atteints par des méthodes conventionnelles.

L'écart de NMI montré en figure 4.6 est encore plus significatif, passant en moyenne pour l'ensemble des graphes de 0.8 avec Louvain+ à 0.9 avec MA-COM/IC. Un gain de NMI de 0.1, avec cette mesure d'échelle logarithmique, est très important. Nous constatons une amélioration ou une équivalence de NMI pour 12 graphes sur 15. La baisse pour les trois restants est très faible. En revanche le gain peut être spectaculaire, avec par exemple pour le graphe $\#7$ une valeur qui passe de 0.44 à 0.91. En général, MA-COM-IC produit une amélioration légèrement supérieure à MA-COM/PC. Ces résultats peuvent surprendre, car ils tendent à montrer que cette optimisation poussée améliore la justesse des partitions contrairement à ce que prévoit la limite de résolution. Nous affinerons ce jugement dans la section sur les défauts de la modularité,

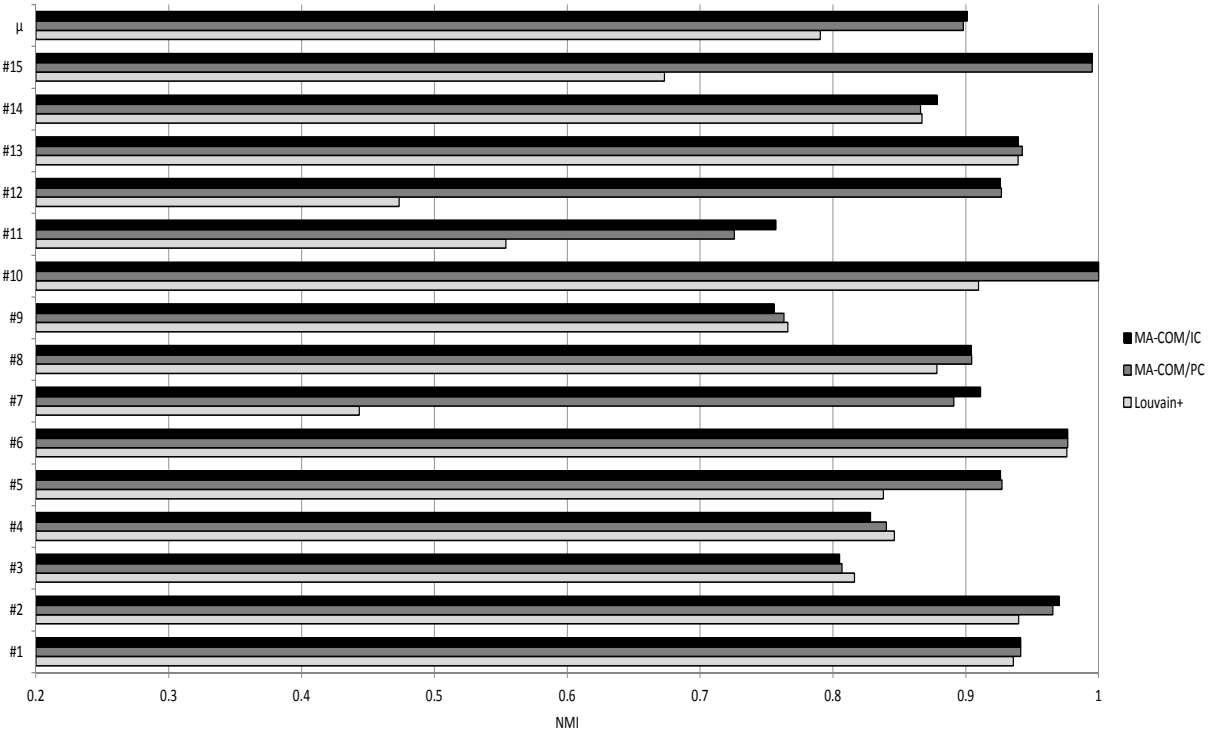


FIGURE 4.6 – NMI avec Louvain+ et les deux versions de MA-COM pour les graphes artificiels. La NMI présentée est la moyenne de NMI sur 20 instances d'exécution.

mais cela peut aussi bien signifier que l'algorithme de Louvain ne produit pas de très bonnes communautés, en dépit d'une optimisation de la modularité correcte.

4.2.3 Graphes réels

Pour apprécier les mesures de performance précises des trois algorithmes que nous avons comparés dans le seul but d'optimiser la modularité, nous publions les modularités maximales et moyennes obtenus avec Louvain, Louvain+ et les deux versions de MA-COM, selon les protocoles d'expérimentation précédemment décrits (version déterministe de Louvain et Louvain+ exécutée sur 100 instances préétablies, version stochastique de MA-COM exécutée 20 fois). Le tableau 4.1 fournit ces résultats. Nous constatons que Louvain+ est meilleur que Louvain pour tous les graphes, de même MA-COM/PC est meilleur que Louvain+. La version MA-COM/IC est presque toujours meilleure que MA-COM/PC (les cas de performances inférieures sont rares avec des pertes très faibles), mais l'amélioration est faible, inférieure à 10^{-3} en général. Elle devient plus importante pour les deux plus grands graphes. Nous pouvons ici constater avec précision le gain très appréciable du raffinement de Louvain+, avec un temps d'exécution presque équivalent.

Le tableau fournit également le support à une comparaison avec les meilleurs résultats d'optimisation publiés, pour une partie seulement des graphes de notre jeu de test. Nous pouvons constater que l'algorithme mémétique égale ou dépasse la meilleure modularité connue pour tous ces graphes, que ce soit en valeur maximale ou en valeur moyenne sur l'ensemble des instances d'exécution.

La figure des scores SC, SN et SCM comparés entre Louvain+ et MA-COM ne montre pas de grande différence concernant la structure des partitions (voir Figure 4.7). Nous remarquons tout de même une amélioration globale apportée par MA-COM, le premier quartile étant plus

TABLE 4.1 – Modularités moyenne et maximale pour Louvain, Louvain+ (100 instances de graphes, paramètres de la Section 3.2) et MA-COM (20 instances d'exécution, $|P| = 20$ et $d_{min} = 10^{-3}$) sur les graphes réels. La colonne MRP indique la modularité des meilleurs résultats publiés, en se fondant sur les deux algorithmes les plus performants que sont LPAm+ [55] et SS-ML [68].

moyenne de Q	Louvain	Louvain+	MRP	MA-COM/PC	MA-COM/IC
Karate Club	0.4163	0.4187	0.4198	0.4198	0.4198
Dolphins	0.5198	0.5227	0.529	0.5282	0.5283
Political Books	0.5258	0.5267	0.527	0.5272	0.5272
College Football	0.6036	0.6037	0.605	0.6046	0.6046
Codeminer	0.8665	0.8703		0.8725	0.8727
C. elegans	0.4368	0.4416	0.452	0.4527	0.4528
Jazz	0.4428	0.4444	0.4451	0.4451	0.4451
E-mail	0.5684	0.5761	0.582	0.5827	0.5826
Power	0.9357	0.9380		0.9400	0.9404
Yeast	0.5924	0.5995		0.6075	0.6107
Epa	0.6494	0.6626		0.6744	0.6754
Erdos	0.6963	0.7131	0.7162	0.7173	0.7180
California	0.6567	0.6693		0.6800	0.6816
Arxiv	0.8143	0.8202	0.813	0.8240	0.8254
PGP	0.8822	0.8835	0.8841	0.8861	0.8864
Zemail	0.6736	0.6787		0.6839	0.6838
Condmat2003	0.8102	0.8141	0.8146	0.8152	0.8183
Astro-ph	0.7312	0.7373		0.7433	0.7454

maximum de Q	Louvain	Louvain+	MRP	MA-COM/PC	MA-COM/IC
Karate Club	0.4198	0.4198	0.4198	0.4198	0.4198
Dolphins	0.5277	0.5277	0.529	0.5285	0.5285
Political Books	0.5272	0.5272	0.527	0.5272	0.5272
College Football	0.6046	0.6046	0.605	0.6046	0.6046
Codeminer	0.8701	0.8727		0.8727	0.8727
C. elegans	0.4471	0.4509	0.452	0.4531	0.4531
Jazz	0.4451	0.4451	0.4451	0.4451	0.4451
E-mail	0.5757	0.5814	0.582	0.5828	0.5828
Power	0.9370	0.9392		0.9403	0.9407
Yeast	0.5962	0.6048		0.6093	0.6121
Epa	0.6561	0.6688		0.6757	0.6763
Erdos	0.6999	0.7154	0.7162	0.7178	0.7184
California	0.6666	0.6763		0.6812	0.6821
Arxiv	0.8166	0.8223	0.813	0.8246	0.8258
PGP	0.8842	0.8852	0.8841	0.8864	0.8865
Zemail	0.6794	0.6825		0.6844	0.6843
Condmat2003	0.8114	0.8151	0.8146	0.8162	0.8185
Astro-ph	0.7344	0.7412		0.7443	0.7459

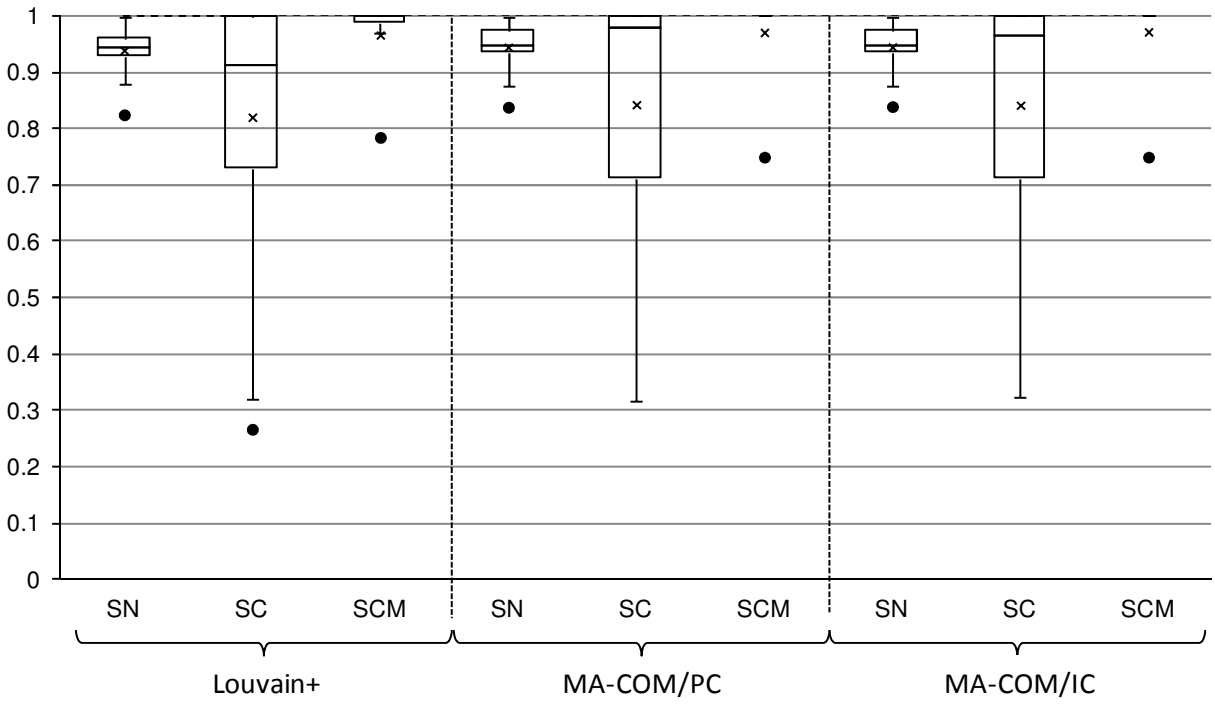


FIGURE 4.7 – Boîtes à moustaches des scores SC, SN et SCM, pour Louvain+ et MA-COM, sur les graphes réels.

haut et plus étroit pour les trois scores. Les valeurs atypiques sont toutes dues aux graphes *Celegans* et *Jazz* qui n'obtiennent pas de très bons scores. Les deux graphes posant un problème de structure ou de partitionnement sont *Politics* et *Jazz* qui sont les seuls à obtenir un score de communauté minimal (SCM) nettement inférieur à 1, respectivement 0.8 et 0.75, quelque soit l'algorithme. Cette anomalie sera étudiée de façon plus approfondie dans la grande partie suivante consacrée aux palliatifs des défauts de la modularité.

4.2.4 Temps d'exécution et complexité

Il est difficile d'interpréter les courbes de temps d'exécution présentées en figure 4.8, car le critère d'arrêt et la nature même d'un algorithme évolutionnaire, qui optimise la fonction objectif par des sauts imprévisibles, provoquent une variabilité conséquente de la durée d'exécution. Toutefois, les courbes tendent à montrer une complexité en temps quasi linéaire quelque soit la version. MA-COM/IC est nettement plus rapide à partir de $m = 200\,000$, alors que les deux courbes sont très proches en dessous. Nous remarquons aussi des à-coups dans les deux courbes traduisant la difficulté de l'algorithme avec certains graphes, par exemple *Jazz* pour les deux versions qui est plus bas qu'attendu (creux vers $m = 2\,700$).

Un algorithme génétique ou mémétique peut s'exécuter longuement, selon la définition du critère d'arrêt, s'il parvient à augmenter la valeur de la fonction objectif par de très petits sauts qui repoussent sans cesse son arrêt. Il est intéressant de mesurer la vitesse de convergence, en mettant en rapport les valeurs de modularité avec le temps d'exécution. Nous avons réalisé cette mesure pour les graphes *Yeast* et *Astro-ph*, dont le résultat est présenté en figures 4.9 et 4.10. La performance finale de MA-COM/IC sur MA-COM/PC apparaît clairement avec l'écart de modularité en fin d'exécution. Mais le plus remarquable est la forme de la courbe. MA-COM/IC a une convergence très rapide caractérisée par une pente raide sur le premier tiers du temps. Après ce premier tiers, la valeur de modularité est très faiblement améliorée, ce qui se traduit

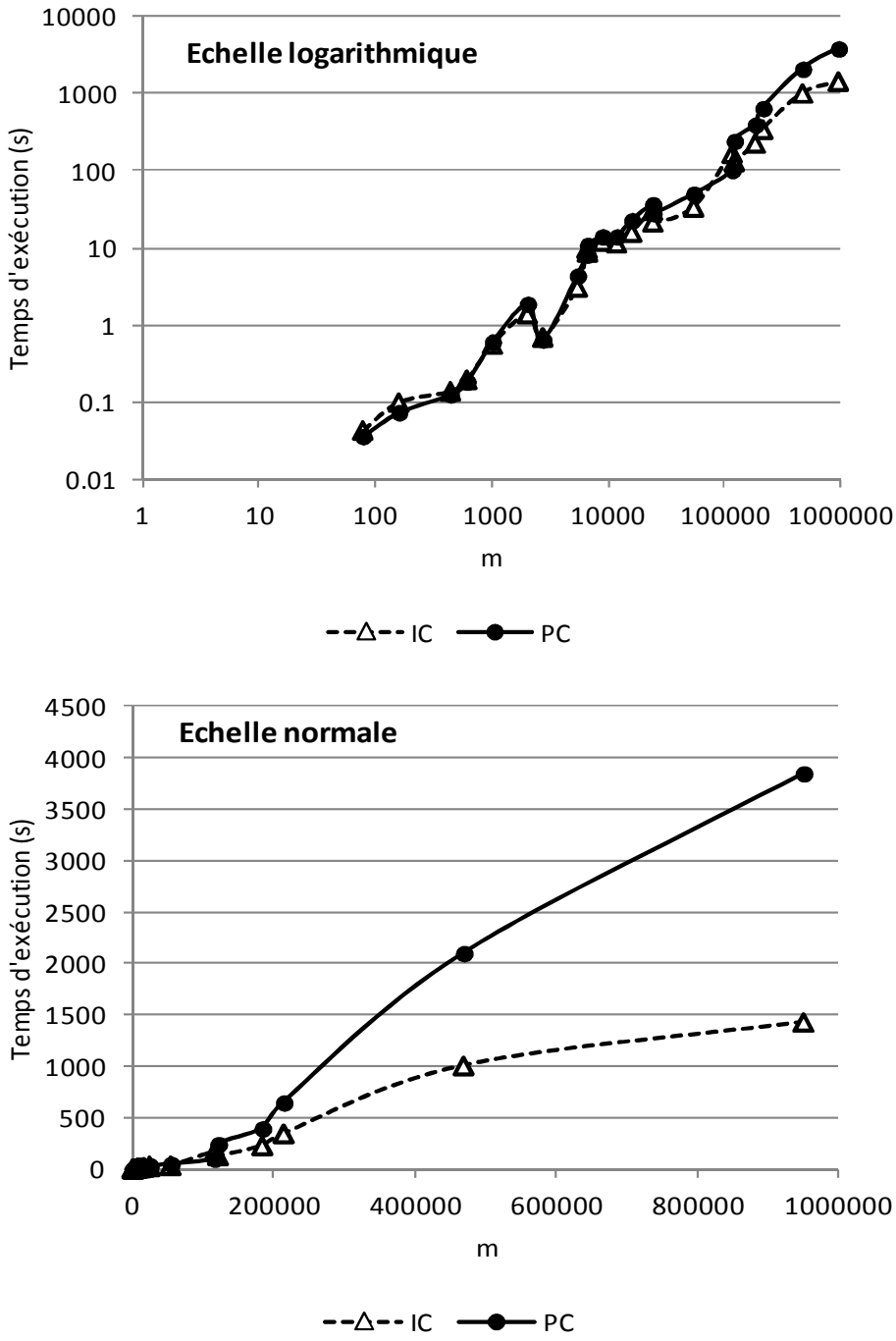


FIGURE 4.8 – Temps d'exécution de MA-COM en fonction du nombre d'arêtes m. Nous avons ajouté aux tests précédents les graphes manquants sur seulement 5 exécutions, pour obtenir un test de temps d'exécution sur l'ensemble des graphes en un temps raisonnable. Le temps présenté est la moyenne en seconde des temps calculés pour chaque graphe. Les deux axes du graphique supérieur sont à l'échelle logarithmique pour juger des petites valeurs.

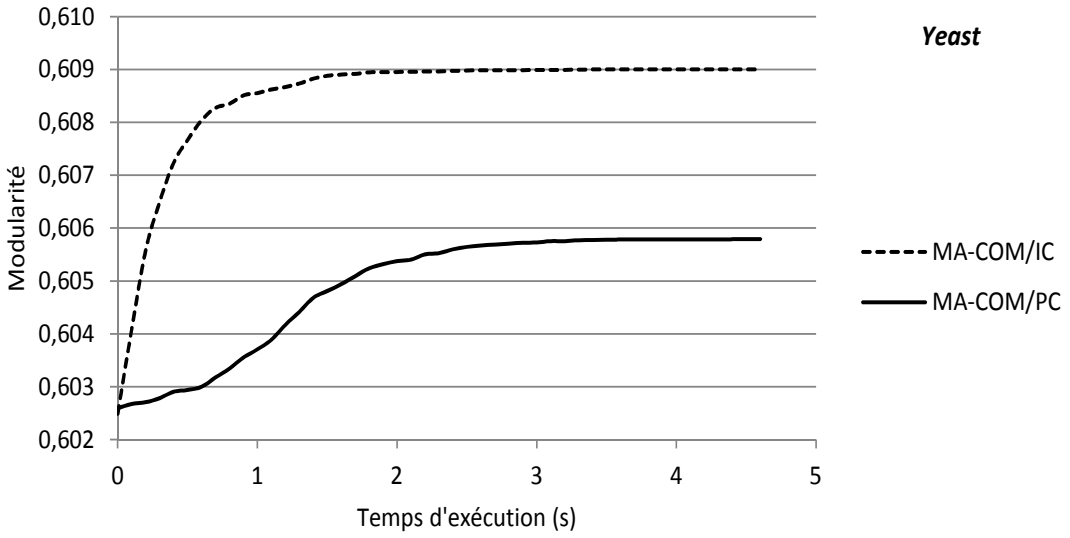


FIGURE 4.9 – Évolution moyenne de la modularité par rapport au temps d'exécution de MA-COM appliqué au graphe *Yeast*. La moyenne est calculée sur les 20 instances d'exécution.

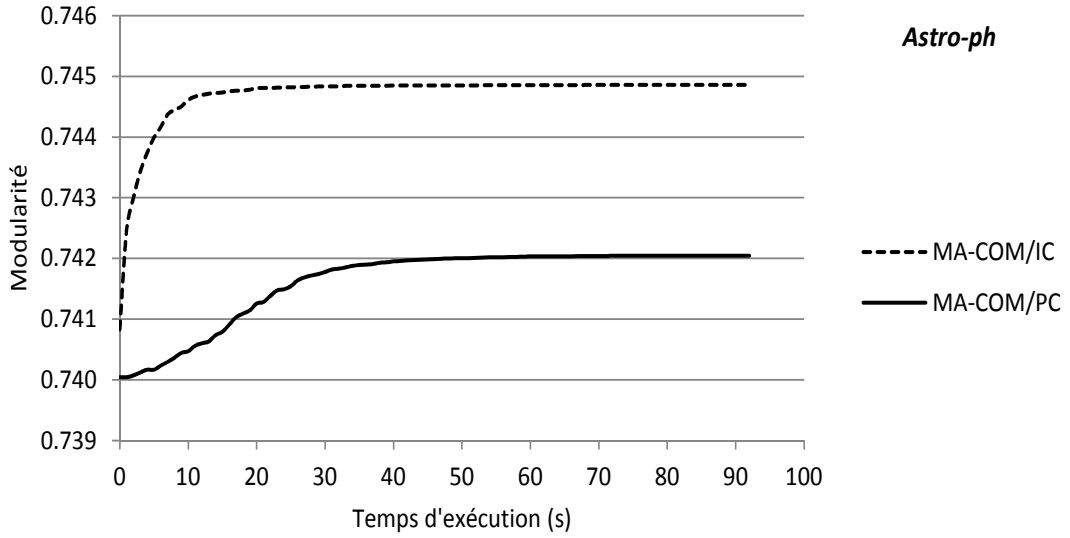


FIGURE 4.10 – Évolution moyenne de la modularité par rapport au temps d'exécution de MA-COM appliqué au graphe *Astro-ph*. La moyenne est calculée sur les 20 instances d'exécution.

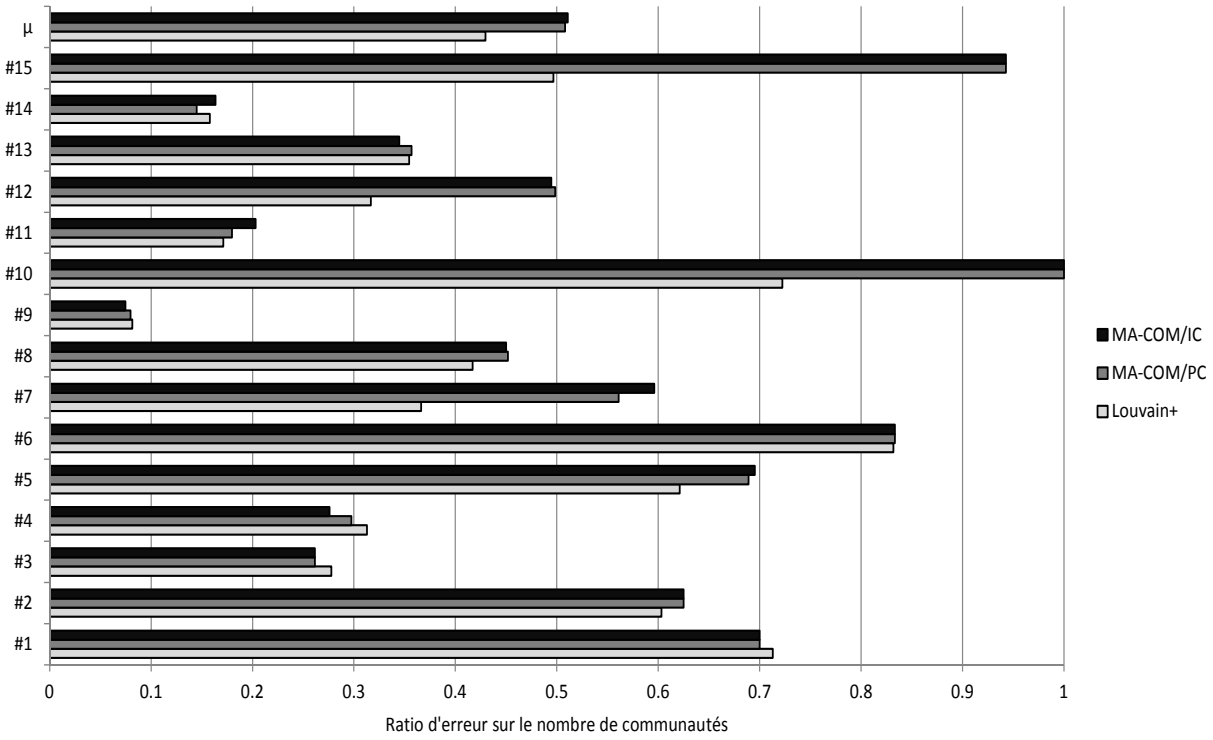


FIGURE 4.11 – Ratio d'erreur sur le nombre de communautés détectées par MA-COM dans les graphes artificiels. Le ratio est calculé, pour chaque graphe, comme le nombre de communautés moyen sur les 20 instances divisé par le nombre de communautés de la solution. La moyenne du ratio figure en vis-à-vis du symbole μ.

par un plateau jusqu'à la fin du temps d'exécution. Pour MA-COM/PC, le plateau apparaît bien plus tard, environ aux deux tiers du temps, et la convergence pour y arriver est beaucoup plus lente, la pente étant inférieure à 1.

4.3 Impact sur les défauts de la modularité

4.3.1 Limite de résolution

Sur le modèle de la comparaison entre Louvain et Louvain+ du chapitre précédent, nous présentons sur les figures 4.11 et 4.12 les erreurs du nombre de communautés et des tailles de plus petite et de plus grande communauté par rapport à la solution associée à chaque graphe artificiel. La comparaison porte désormais sur Louvain+, MA-COM/PC et MA-COM/IC.

Le nombre de communautés trouvé par MA-COM, dans les deux versions, est meilleur que celui obtenu par Louvain+ pour 8 graphes, dont 5 pour lesquels l'augmentation de NMI est très nette ($\#5$, $\#7$, $\#10$ $\#12$ et $\#15$). Le résultat est légèrement dégradé pour 4 autres graphes et incertain (dégradé pour une version, améliorée pour l'autre) pour les 3 restants. Dans l'ensemble, la moyenne des ratios d'erreur du nombre de communautés passe de 0.42 pour Louvain+ à un peu plus de 0.5 pour MA-COM, l'idéal sans erreur étant de 1. L'amélioration est nette, mais le résultat est loin d'être satisfaisant avec, en moyenne, un nombre de communautés détectées deux fois trop faible. La taille de plus petite communauté est pratiquement inchangée et demeure très mauvaise, traduisant le plus significativement la limite de résolution due à l'optimisation de la modularité. Enfin, les plus grandes communautés sont en moyenne meilleures

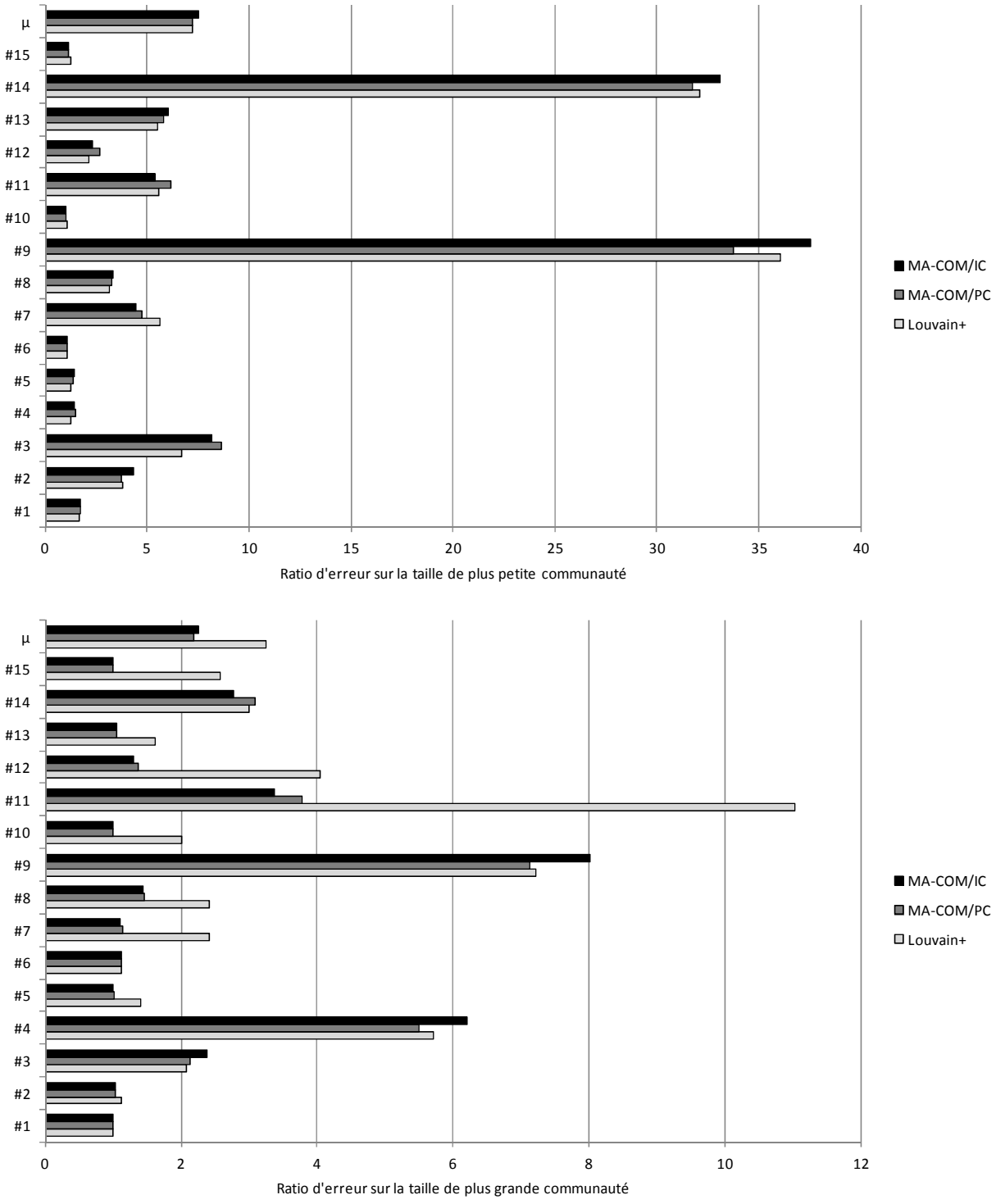


FIGURE 4.12 – Ratio d'erreur sur les tailles minimale et maximale des communautés détectées par MA-COM dans les graphes artificiels. Les ratios sont calculés, pour chaque graphe, comme la taille de plus petite (resp. grande) communauté moyenne sur les 20 instances d'exécution divisée par la taille de plus petite (resp. grande) communauté de la solution. La moyenne de ratio figure en vis-à-vis du symbole μ.

avec MA-COM. C'est sans doute l'impact le plus remarquable de cet algorithme même si les erreurs sont loin d'être complètement corrigées, le ratio moyen passant de 3.3 avec Louvain+ à 2.2 avec MA-COM (en moyenne, les grandes communautés sont encore deux fois trop grandes par rapport à la solution attendue).

Selon nos observations détaillées des résultats et des structures communautaires des graphes artificiels, l'algorithme mémétique agit efficacement par rapport à Louvain+ selon deux modalités. Tout d'abord, il existe des déplacements de paire de nœuds pouvant augmenter la modularité alors que le déplacement de l'un des deux ne le fait pas. Cette amélioration, individuellement infime, mais pouvant cumulativement être importante si ces paires sont présentes en nombre, est possible uniquement avec l'algorithme mémétique grâce au croisement. De plus, les graphes peu denses ou possédant une structure communautaire faiblement définie, comme les graphes #7, #11, #12 et #15 pour lesquels la justesse de partition est fortement améliorée par MA-COM, mettent en échec l'algorithme de Louvain dès la première phase VM du fait de communautés trop grandes constituées par une succession de déplacements de nœuds. Cette erreur ne peut pas être corrigée par la deuxième phase de l'algorithme de Louvain qui ne procède qu'à des regroupements de communautés. En revanche, l'algorithme mémétique est capable de casser les grandes communautés par son croisement, qui ne sont pas nécessairement reconstituées par l'algorithme Louvain+ appliqué à une progéniture du fait d'un ajustement préalable du placement des nœuds par la première phase VM de cet algorithme. Ce constat est flagrant avec le résultat de ratio de plus grande communauté. Ainsi les deux algorithmes associés fonctionnent en une synergie qui explique les performances de cette association.

4.3.2 Inconsistance de solution

L'un des apports essentiels de l'algorithme mémétique est la réduction de l'inconsistance de solution corrélée à une optimisation très poussée, constatée sur la figure 4.13. En moyenne, la similarité entre solutions de même graphe passe de 0.8 avec Louvain+ à 0.9 avec MA-COM/PC et même 0.92 avec MA-COM/IC, ce qui constitue un rapprochement des solutions très important, compte tenu, rappelons-le, de l'échelle logarithmique de la NID. L'amélioration est constatée pour tous les graphes avec logiquement de meilleurs résultats pour les plus petits graphes.

Conclusion

Les algorithmes mémétiques présentés utilisent deux mécanismes de croisement dédiés au partitionnement de graphe qui permettent une optimisation de la modularité d'un niveau inédite. Les solutions trouvées par différentes instances d'exécution sont très proches entre elles, réduisant ainsi l'inconsistance de solution constatée avec les algorithmes classiques du fait d'un très grand nombre d'optimum locaux dans l'espace de recherche de la modularité. Pourtant, la justesse des partitions, selon le modèle de graphes artificiels LFR, n'est que très insuffisamment améliorée, car les effets de la limite de résolution s'aggravent avec l'optimisation poussée de la modularité. Hormis ce défaut, la modularité offre un cadre pratique au partitionnement de graphe en s'affranchissant de la méconnaissance a priori du nombre de communautés et en se prêtant à des algorithmes très rapides et simples d'implantation comme Louvain/Louvain+. Dans une deuxième partie, notre objectif est d'identifier dans les mécanismes de l'algorithme de Louvain une cause d'aggravation de la limite de résolution et d'y remédier pour préserver le cadre général de cet algorithme et en particulier l'usage de la modularité.

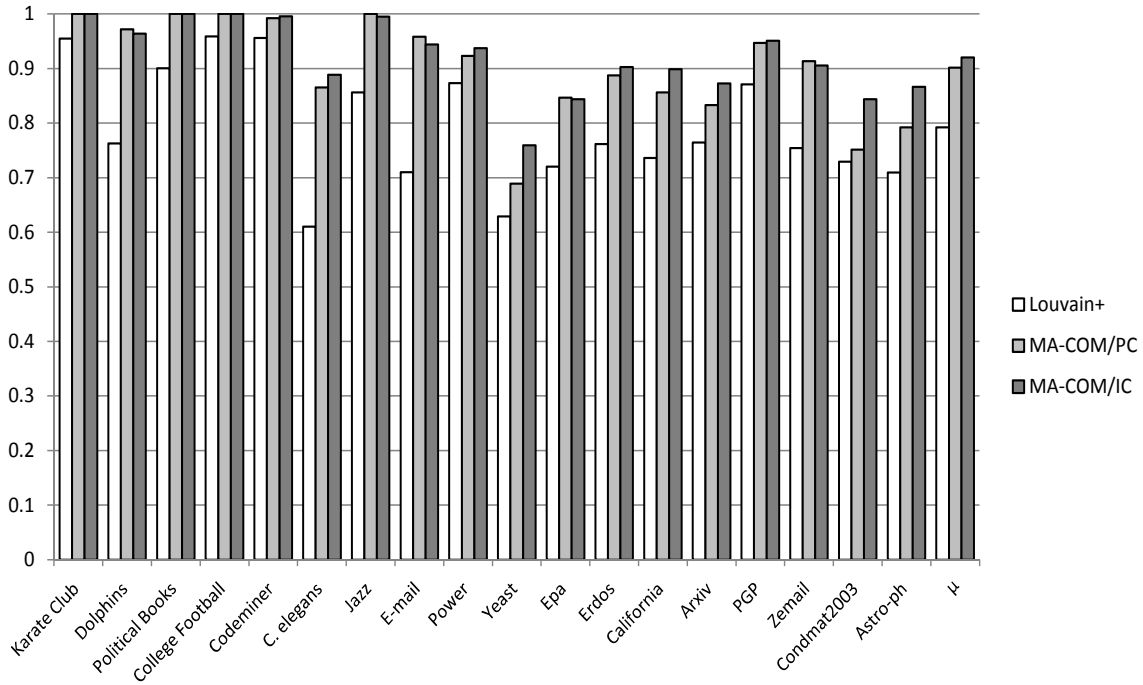


FIGURE 4.13 – Inconsistance de solution de MA-COM sur les graphes réels. La version stochastique de l'algorithme est exécutée 20 fois sur chaque graphe. La similarité moyenne est calculée comme la moyenne de la NID de toutes les paires distinctes de solutions. Une similarité proche de 1 indique des solutions structurellement proche et donc une inconsistance liée à la modularité réduite.

Palliatifs des défauts de la modularité

5

Condition de fusion

Sommaire

Nous décrivons dans ce chapitre la principale contribution de notre travail au problème de la détection de communautés, une condition de fusion de communautés, qui est le fondement des deux nouveaux algorithmes présentés dans les chapitres suivants. Pour éviter certaines des fusions erronées réalisées par la phase de contraction de l'algorithme de Louvain, nous avons établi un critère de fusion, fondé sur la modularité paramétrique calculée entre deux communautés. Ce critère, appliqué à Louvain+, réduit fortement la limite de résolution, de même que l'inconsistance de solution. L'objet de ce chapitre est de poser et d'étudier la formulation du critère de fusion, selon un paramètre λ, puis de vérifier expérimentalement son efficacité, toujours selon les valeurs de ce paramètre. Les chapitres suivants sont consacrés aux tests plus détaillés des deux algorithmes implantant ce critère.

Ce chapitre a été présenté à la conférence *Modèles et Analyse des Réseaux (MARAMI-2013)* et publié dans ses actes.

Introduction

L'algorithme de Louvain procède en deux phases : une phase de constitution des communautés par déplacement de nœuds (procédure VM, *vertex mover*), puis une phase de regroupement hiérarchique des communautés. Les résultats de ces deux phases montrent que la limite de résolution se manifeste très majoritairement dans la seconde phase. Les opérations réalisées dans celle-ci sont fondamentalement équivalentes aux fusions de communautés réalisées dans des algorithmes comme FastGreedy, à ceci près que les fusions sont réversibles. En observant les graphes artificiels LFR, le nombre de communautés obtenu à l'issu de VM est plus grand que le nombre de communautés de la solution, alors qu'il devient plus petit après la phase de fusion de Louvain. Il en va de même pour les petites communautés qui grossissent et deviennent trop grandes après cette seconde phase. C'est la manifestation la plus évidente de la limite de résolution. Même si intrinsèquement l'optimisation de la modularité engendre ce défaut, quel que soit la façon de construire une partition, nous pouvons conjecturer que certaines fusions sont erronées alors que d'autres sont justifiées.

Notre idée est d'établir un critère local de fusion de communautés. À partir de deux communautés C et C' et du sous-graphe $C \cup C'$ qu'elles engendrent, ce critère est vrai si C et C' doivent être fusionnées et faux sinon. Les graphes LFR nous permettent d'apprécier la performance d'un tel critère car nous pouvons déterminer, d'après la partition solution, si deux communautés doivent ou non être fusionnées. Des tests préliminaires à l'aide des 20 graphes artificiels de test ont montré que la modularité paramétrique Q_λ fournit un critère pertinent avec un paramètre λ proche de 0.25, alors que la modularité simple n'est pas pertinente. De manière générale, la modularité (simple ou paramétrique) peut être calculée pour la partition $\{C, C'\}$ dans le sous-graphe $C \cup C'$ et le critère est positif si cette modularité est inférieure à un seuil à déterminer. L'idée sous-jacente, déjà formulée par plusieurs auteurs pour la modularité simple, est que deux communautés doivent être réunies si elles ne constituent pas séparément deux modules forts, c'est-à-dire si leur modularité est faible.

5.1 Formulation

5.1.1 Formule générale

Soit deux communautés C et C' de la partition $\mathcal{P}$ définie dans le graphe $G = (V, E)$. Notons $G_{C,C'} = (C \cup C', E_{C,C'})$ le sous-graphe de G engendré par le sous-ensemble de nœuds $C \cup C'$. $E_{C,C'}$ est l'ensemble des arêtes de ce sous-graphe et son cardinal est noté e pour simplifier les formules. Le degré d'un nœud v relatif à un sous-graphe S, noté $d_S(v)$ est défini comme la somme des poids des arêtes de S incidente à v. Le degré d'une communauté C relativement à un sous-graphe S, noté $d_S(C)$, est alors défini comme la somme des degrés $d_S(v)$ des nœuds v appartenant à C, soit $d_S(C) = \sum_{v \in C} d_S(v)$. Cette définition préalable permet d'écrire la modularité paramétrique de la partition $\{C, C'\}$ dans le graphe $G_{C,C'}$:

$$Q_\lambda(C, C') = \frac{l(C) + l(C')}{e} - \lambda \frac{d_{G_{C,C'}}(C)^2 + d_{G_{C,C'}}(C')^2}{4e^2} \tag{5.1}$$

Le critère de fusion $M_\theta(C, C')$ s'écrit alors, pour un paramètre θ compris entre 0 et 1 :

$$Q_\lambda(C, C') < \theta \tag{5.2}$$

Cela signifie que les communautés C et C' sont fusionnées si la modularité de paramètre λ de la partition $\{C, C'\}$ dans le sous-graphe $C \cup C'$ est strictement inférieure au seuil θ.

Pour caractériser les valeurs de θ, nous devons tout d'abord reformuler le critère en réduisant les variables, pour l'instant au nombre de cinq : $l(C)$, $l(C')$, $d_{G_{C,C'}}(C)$, $d_{G_{C,C'}}(C')$ et $e = |E_{C,C'}|$.

5.1.2 Reformulation

Le degré de communauté est la somme des degrés internes et externes, $d(C) = d_{out}(C) + d_{in}(C)$. De plus, $l(C)$, le poids total des arêtes internes à la communauté C est tel que $l(C) = \frac{1}{2}d_{in}(C)$, car en faisant la somme des degrés internes à C, chaque arête est comptée deux fois. Nous obtenons $d(C) = 2l(C) + d_{out}(C)$. Dans le sous-graphe $G_{C,C'}$, les arêtes qui ont une extrémité dans C et l'autre en dehors, dont la somme des poids est $d_{out}(C)$, relient nécessairement C à C', car il n'y a que deux communautés. Donc, $d_{out}(C) = d_{out}(C') = l(C, C')$, la somme des poids des arêtes reliant C et C'. Nous en déduisons trois relations importantes pour la suite, dans le contexte de deux communautés C et C' engendrant un sous-graphe $G_{C,C'}$:

$$\begin{cases} d_{G_{C,C'}}(C) = 2l(C) + l(C, C') \\ d_{G_{C,C'}}(C') = 2l(C') + l(C, C') \\ l(C) + l(C') + l(C, C') = e \end{cases} \tag{5.3}$$

Selon ces relations, $d_{G_{C,C'}}(C) = e + l(C) - l(C')$ et $d_{G_{C,C'}}(C') = e - l(C) + l(C')$. La modularité se réécrit :

$$\begin{aligned} Q_\lambda(C, C') &= \frac{l(C) + l(C')}{e} - \frac{\lambda}{4e^2}\left((e + l(C) - l(C'))^2 + (e - l(C) + l(C'))^2\right) \\ &= \frac{l(C) + l(C')}{e} - \frac{\lambda}{4e^2}\left(2e^2 + 2(l(C) - l(C'))^2\right) \\ &= \frac{l(C) + l(C')}{e} - \frac{\lambda}{2e^2}(l(C) - l(C'))^2 - \frac{\lambda}{2} \end{aligned} \tag{5.4}$$

Dans cette forme, l'expression de la modularité ne conserve que trois variables : $l(C)$, $l(C')$ et $e = |E_{C,C'}|$.

5.1.3 Expression de θ

Pour borner le paramètre θ, nous considérons des cas extrêmes, en partant du principe que e, le nombre d'arêtes du sous graphe engendré par $C \cup C'$, est constant et en faisant varier $l(C)$ ou $l(C')$ entre 0 et e, le maximum possible.

Borne supérieur de θ

Supposons que les communautés C et C' ne soient pas connectées, c'est-à-dire que $l(C, C') = 0$. La troisième égalité dans 5.3 donne $l(C) + l(C') = e$ soit $l(C) - l(C') = 2l(C) - e$. La modularité écrite en 5.4 devient :

$$\begin{aligned} Q_\lambda(C, C') &= 1 - \frac{\lambda}{2e^2}(2l(C) - e)^2 - \frac{\lambda}{2} \\ &= \frac{-2\lambda l(C)^2}{e^2} + \frac{2\lambda l(C)}{e} + 1 - \lambda \end{aligned} \tag{5.5}$$

Dans la situation du sous-graphe ayant e arêtes internes, la modularité obtenue peut être vue comme une fonction de la variable $l(C)$ définie sur $\{0, ..., e\}$. Cette fonction est croissante sur l'intervalle $[0...e/2]$ et décroissante sur $[e/2...e]$. Pour $l(C) = 0$ et $l(C) = e$, $Q_\lambda(C, C') = 1 - \lambda$. Nous avons donc $Q_\lambda(C, C') \geq 1 - \lambda$. Le critère de fusion, qui s'écrit $Q_\lambda(C, C') < \theta$ doit toujours être négatif, car dans la situation d'hypothèse ($l(C, C') = 0$), C et C' ne doivent jamais être fusionnées. Il faut donc choisir θ de façon à ce que $Q_\lambda(C, C') \geq \theta$, quelque soit C et C' non connectées. Cela est vérifié si $\theta \leq 1 - \lambda$, car dans ce cas $Q_\lambda(C, C') \geq 1 - \lambda \geq \theta$.

Borne inférieure de θ

A l'inverse ici, considérons que C et C' sont connectées, donc que $l(C, C') > 0$. Cette hypothèse nous assure que les communautés peuvent être fusionnées. Considérons de plus que la communauté C' contient seulement un nœud et donc aucune arête interne : $l(C') = 0$. Il en résulte que $e = l(C) + l(C, C')$ et, selon la première hypothèse, $l(C) < e$. La modularité utilisée dans le critère de fusion devient :

$$Q_\lambda(C, C') = -\frac{\lambda}{2e^2} l(C)^2 + \frac{l(C)}{e} - \frac{\lambda}{2} \tag{5.6}$$

Comme précédemment, la modularité peut être vue comme une fonction de la variable $l(C)$ définie sur $\{0, ..., e\}$. Dans l'intervalle $[0...e]$, cette fonction est strictement croissante avec pour borne $-e/\lambda$ pour $l(C) = 0$ et $1 - \lambda$ pour $l(C) = e$. Donc, dans l'intervalle de variation de $l(C)$, comme de plus par hypothèse $l(C) < e$, nous avons $Q_\lambda(C, C') < 1 - \lambda$. Dans la situation d'hypothèse, où une communauté C' ne contient qu'un nœud, il est souhaitable par principe de n'interdire aucune fusion avec C' de façon à ce que toutes les situations de fusion de C' avec d'autres communautés voisines soient examinées pour trouver une fusion de gain de modularité maximum. Donc, le critère de fusion doit être positif et il faut choisir θ de sorte que $Q_\lambda(C, C') < \theta$ ce qui est vérifié si $\theta \geq 1 - \lambda$, car dans ce cas $Q_\lambda(C, C') < 1 - \lambda \leq \theta$.

La première hypothèse de connexion de C et C' n'est pas préjudiciable, car il est inutile que le critère autorise une fusion qui, par principe, n'est pas viable.

Valeur de θ

Nous avons ainsi borné le paramètre θ selon deux cas particuliers où la réponse du critère de fusion est imposée. Il résulte de ces bornes que la seule valeur acceptable de θ est $1 - \lambda$, soit un critère de fusion $M_\lambda(C, C')$ qui s'écrit simplement :

$$Q_\lambda(C, C') < 1 - \lambda, \lambda > 0 \tag{5.7}$$

5.2 Étude générale

Le critère de fusion étant posé avec un seul paramètre λ, nous souhaitons étudier dans quelles conditions et pour quelles valeurs de λ, ce critère opère correctement et en particulier atténue la limite de résolution. L'idée générale est qu'il fonctionne comme un filtre préalable avant l'examen du gain de modularité obtenue par la fusion de deux communautés.

Nous avons besoin pour la suite d'une écriture du critère de fusion sans e au dénominateur :

$$\begin{aligned} &\frac{l(C) + l(C')}{e} - \frac{\lambda}{2e^2}(l(C) - l(C'))^2 - \frac{\lambda}{2} < 1 - \lambda \\ &2e(l(C) + l(C')) - \lambda(l(C) - l(C'))^2 + (\lambda - 2)e^2 < 0 \end{aligned} \tag{5.8}$$

5.2.1 Valeurs de λ admissibles

Interrogeons-nous sur la valeur minimale de la modularité paramétrique $Q_\lambda(C, C')$ d'après sa forme générale écrite en 5.4. Posons $l(C) - l(C') = h$, h variant de $-e$ à e. La modularité paramétrique devient :

$$Q_\lambda(C, C') = \frac{2l(C)}{e} - \frac{h}{e} - \frac{\lambda}{2e^2} h^2 - \frac{\lambda}{2} \tag{5.9}$$

Pour h constant, cette fonction de $l(C)$ est croissante, donc son minimum est atteint pour $l(C) = 0$ et vaut $-h/e - \lambda h^2/(2e^2) - \lambda/2$. Il y a ainsi un minimum pour chaque valeur de h possible. Sous la condition que $l(C) = 0$, h peut varier de $-e$ à 0. La fonction $f(h) = -h/e - \lambda h^2/(2e^2) - \lambda/2$ est toujours négative, puisque $\lambda \geq 0$, et a la forme d'un polynôme du second degré. Donc sur l'intervalle $[-e...0]$ son minimum est soit $f(-e)$, soit $f(0)$, c'est-à-dire soit $1 - \lambda$, soit $-\lambda/2$. En comparant les deux valeurs, le minimum des deux, c'est-à-dire le minimum absolu de $Q_\lambda(C, C')$ est $-\lambda/2$ si $\lambda < 2$ et $1 - \lambda$ sinon. Dans ce dernier cas, le critère est toujours faux, car la modularité paramétrique doit être strictement inférieure à $1 - \lambda$. Ainsi, le critère de fusion est opérant uniquement si $\lambda < 2$.

5.2.2 Communautés mal formées

Le critère le plus faible d'existence d'une communauté, selon la définition de Hu et al. [29], exige que celle-ci ait au moins deux fois plus d'arêtes internes que de liens partagés avec n'importe quelle autre communauté. Dans la situation réduite à deux communautés C et C', cette condition nécessite que $2l(C)$ et $2l(C')$ soit strictement supérieurs à $l(C, C')$.

Plaçons-nous du point de vue d'une seule des deux communautés, par exemple C, la situation étant symétrique. Selon le critère d'existence de communauté, deux cas peuvent se présenter.

La communauté C est bien formée : $2l(C) > l(C, C')$ La condition de fusion doit opérer pleinement sans préjuger de sa valeur. A la limite, lorsque $l(C, C') = 1$, pouvons-nous exiger que la fusion soit interdite (voir section suivante avec la limite de résolution).

La communauté C est mal formée : $2l(C) \leq l(C, C')$ Dans ce cas la prévention de la limite de résolution n'est pas utile, car la fusion est a priori légitime. Il faut systématiquement autoriser la fusion pour reporter la décision de fusion effective sur la recherche du gain de modularité maximum.

Recherchons dans quelle condition le critère de fusion est toujours vrai si $2l(C) \leq l(C, C')$. Notons $l(C, C') = hl(C)$ avec $h \geq 2$. La valeur h est définie uniquement si $l(C) \neq 0$[1].

Comme $e = (1 + h)l(C) + l(C')$, la condition de fusion écrite en 5.8 devient :

$$\begin{aligned} &[(2\lambda - 2)h + 4\lambda]l(C)l(C') + [(\lambda - 2)h^2 + (2\lambda - 2)h]l(C)^2 < 0 \\ &[(2\lambda - 2)h + 4\lambda]l(C') + [(\lambda - 2)h^2 + (2\lambda - 2)h]l(C) < 0 \\ &[(2\lambda - 2)h + 4\lambda]l(C') < [(2 - \lambda)h^2 + (2 - 2\lambda)h]l(C) \end{aligned} \tag{5.10}$$

La condition doit être vraie quelque soit les valeurs de $l(C)$ et $l(C')$ qui sont positives. Cela est rendu possible si les deux conditions suivantes sont réunies :

$$\begin{cases} (2\lambda - 2)h + 4\lambda < 0 \\ (2 - \lambda)h + 2 - 2\lambda > 0 \end{cases} \tag{5.11}$$

La première inéquation est équivalente à $h < 4\lambda/(2 - 2\lambda)$ si $\lambda \geq 1$ ou $h > 4\lambda/(2 - 2\lambda)$ si $\lambda < 1$. La première est impossible, car si $\lambda \geq 1$ alors $4\lambda/(2 - 2\lambda)$ est négatif ce qui oblige h à être négatif. Il reste $h > 4\lambda/(2 - 2\lambda)$. Selon le même raisonnement que précédemment, comme

[1] Le cas où $l(C)$ est nul ne peut résulter que d'une communauté ne contenant qu'un seul nœud, car il est impossible qu'une optimisation de la modularité, par déplacement de nœuds ou fusion de communautés, amène d'une situation où les communautés sont connexes à une situation contraire. Cette constatation résulte des formules 1.5 et 1.7 qui donne toujours une perte de modularité dans ce cas. La situation où C ne contient qu'un seul nœud a été examinée pour déterminer la borne inférieure de θ. En prenant $\theta = 1 - \lambda$, nous avons montré que le critère est vrai dans cette situation, donc l'exclusion $l(C) \neq 0$ n'est pas préjudiciable pour la suite de notre raisonnement.

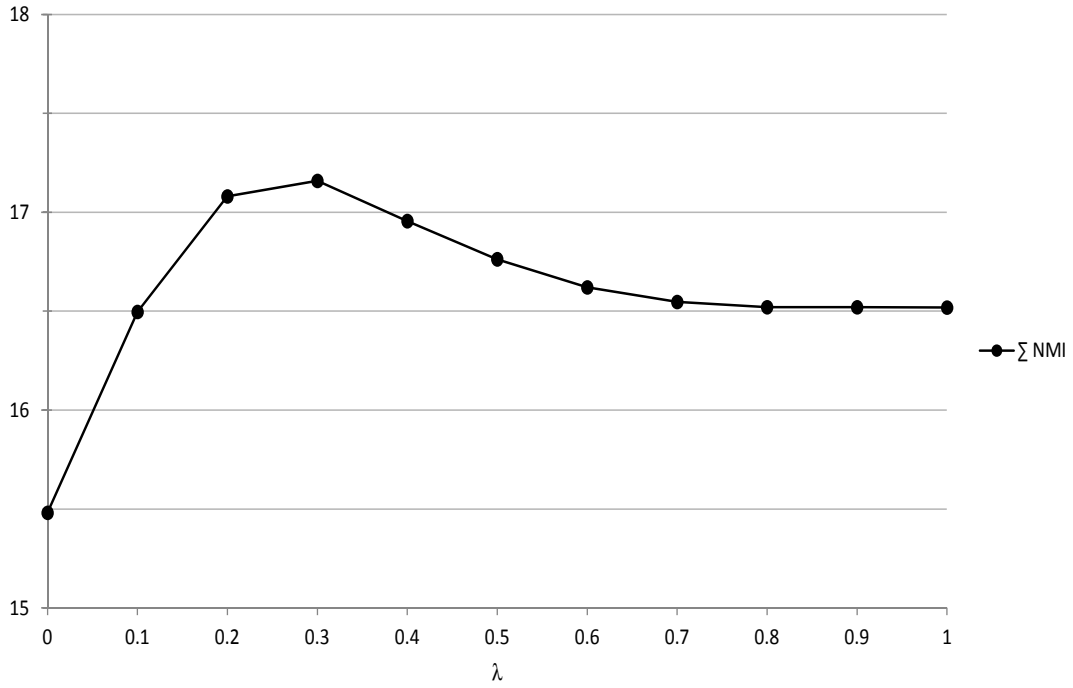


FIGURE 5.1 – Efficacité de la condition de fusion selon le paramètre λ. Pour une instance d'exécution, la mesure de similarité NMI est calculée entre la partition trouvée et la partition de référence. Pour chaque graphe, la NMI moyenne est calculée sur l'ensemble des 100 instances. Enfin, cette moyenne est cumulée pour l'ensemble des graphes et présentée sur ce graphique. La valeur maximale est 20 puisque 20 graphes sont testés (NMI comprise entre 0 et 1).

$h \geq 2$, cette inéquation est rendue toujours vraie en prenant λ de sorte que $2 > 4\lambda/(2-2\lambda)$, soit $\lambda < 1/2$ (compatible avec la condition initiale $\lambda < 1$).

La seconde inéquation est équivalente à $h > (2\lambda - 2)/(2 - \lambda)$ si $\lambda \leq 2$ (l'autre alternative est impossible, car toujours inférieur à 2). Par le même procédé, on obtient $\lambda < 3/2$.

Les deux conditions devant être vérifiées, la plus restrictive prime et il est donc nécessaire que $\lambda < 1/2$ pour que toute communauté mal formée passe le filtre du critère de fusion.

5.3 Validation expérimentale

Nous avons établi que la condition de fusion $Q_\lambda(C, C') < 1 - \lambda$ est efficace si $0 < \lambda < 1/2$. La borne inférieure est incontestable, car le critère serait toujours positif et donc non discriminant si λ était négatif. En revanche la borne supérieure est soumise à une hypothèse qu'il est préférable de vérifier expérimentalement. Nous avons donc choisi de tester les valeurs de λ de 0 à 1 par pas de 0.1. La valeur $\lambda = 0$ représente l'absence de condition puisque Q_0 est toujours inférieur ou égal à 1. La valeur $\lambda = 1$ correspond à un critère fondé sur la modularité standard $Q_1 = Q$. Nos tests consistent à appliquer la condition de fusion à l'algorithme Louvain+ sur l'ensemble des 20 graphes artificiels LFR et des 22 graphes réels présentés dans le chapitre 2. Les algorithmes Louvain et Louvain+ déterminent une partition initiale par l'heuristique VM en déplaçant des nœuds, puis, dans une seconde phase, procèdent à des regroupements de communautés qui s'apparentent à des fusions. Le critère intervient dans cette seconde phase.

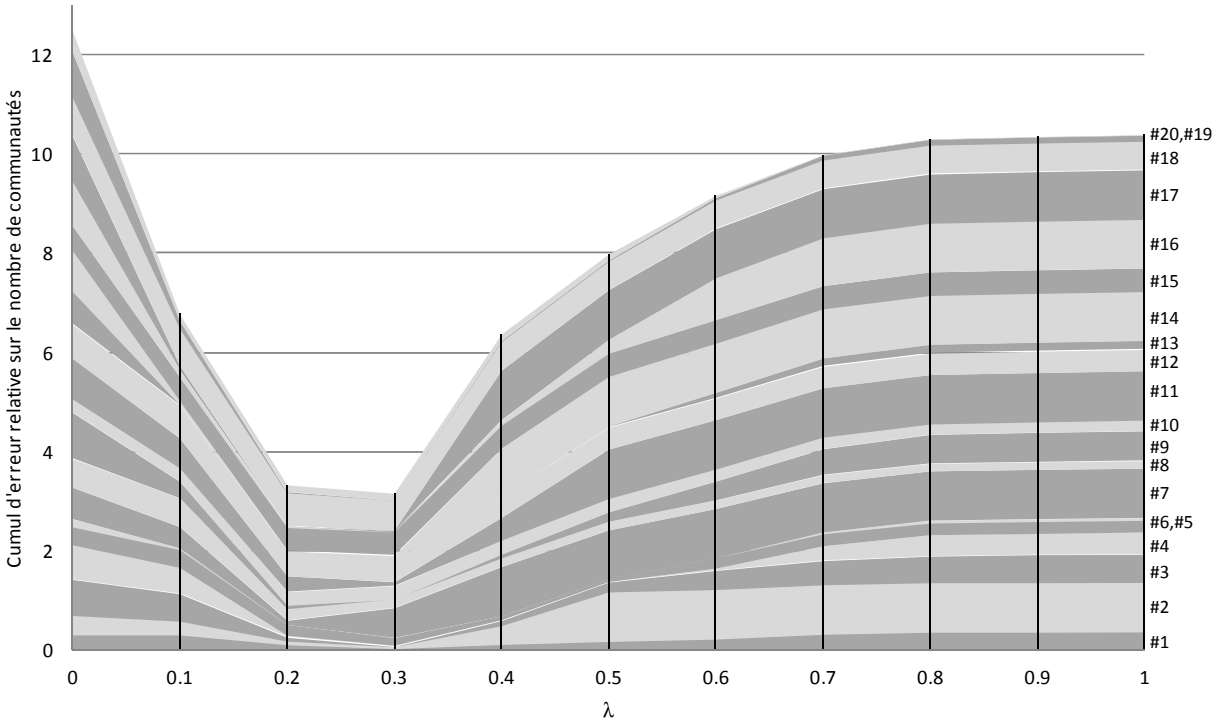


FIGURE 5.2 – Cumul d'erreur relative sur le nombre de communautés détectées sous condition de fusion selon le paramètre λ. Pour chaque graphe, l'erreur relative (taux de variation) du nombre de communautés trouvé par rapport à la solution est calculée et bornée à 1 soit, si le nombre trouvé est k et le nombre de référence k_s, $min(1, |k - k_s|/k_s$. (les erreurs au-dessus de 1 sont ramenées à 1, valeur maximale). Les valeurs sont représentées en ordonnées et empilées pour chaque graphe en commençant en bas par $\#1$. Cela donne une surface, dont la hauteur est l'erreur relative, colorée par alternance. Finalement, la courbe la plus haute représente le cumul des erreurs relatives.

5.3.1 Graphes artificiels

La validation de la condition de fusion par les graphes artificiels est essentielle, car la partition de référence permet de mesurer, par la similarité NMI, la justesse de la partition trouvée pour chaque graphe. En complément de ces mesures globales, nous nous attachons à observer trois indicateurs structurels : le nombre de communautés k, la taille de la plus petite communauté $min|C_i|$ et la taille de la plus grande communauté $max|C_i|$.

La figure 5.1 représente la somme de NMI pour les 20 graphes de test, selon le paramètre λ. La similarité augmente fortement de $\lambda = 0$ à $\lambda = 0.3$ pour diminuer ensuite moins rapidement, mais continûment jusqu'à $\lambda = 1$. La condition de fusion améliore nettement l'algorithme Louvain+ ($\lambda = 0$) quant à la justesse des partitions produites comparativement à la solution des graphes artificiels de test. D'après les résultats détaillés par graphe, non publiés, entre $\lambda = 0$ et $\lambda = 0.3$ la NMI est augmentée pour tous les graphes, avec une moyenne d'augmentation absolue de 0.08. L'augmentation relative moyenne est de 11.5 %. C'est une amélioration très significative compte tenu du caractère logarithmique de la mesure de similarité NMI.

A l'issue de la première phase VM, nous observons que le nombre de communautés k est systématiquement trop grand par rapport à la solution, car il manque des fusions de communauté. En revanche, la seconde phase procède à trop de fusions de sorte que k devient trop petit. La figure 5.2 montre l'erreur relative entre 0 et 1 (toute valeur au-dessus est plafonnée à 1) commise sur le nombre de communautés selon le paramètre λ. Ici encore, les meilleurs

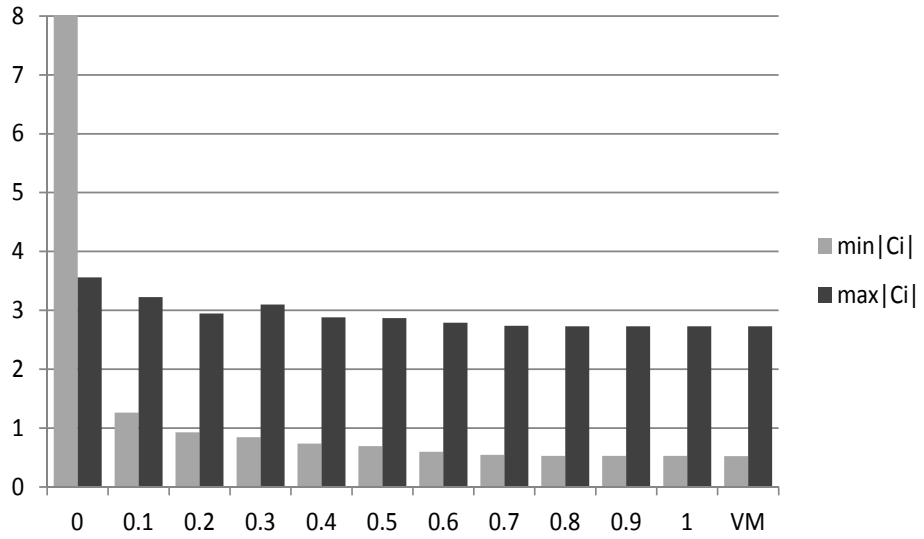


FIGURE 5.3 – Ratios d'erreur sur les tailles minimale et maximale de communauté détectées sous condition de fusion, selon le paramètre λ. Le graphique représente la moyenne, sur l'ensemble des graphes, du rapport entre la taille de la plus petite communauté trouvée et la même grandeur dans la partition solution (barres claires). Les barres foncées représentent le même calcul avec la plus grande communauté. Ces deux valeurs sont montrées par rapport au paramètre λ de l'algorithme Louvain+ avec condition de fusion et pour l'algorithme VM (première phase de Louvain+), à droite en abscisse.

résultats avec un cumul d'erreur faible, inférieur à 4 (le maximum étant 20), est obtenu dans l'intervalle $0.2 \geq \lambda \geq 0.3$. On constate que l'essentiel des erreurs avec le meilleur paramètre $\lambda = 0.3$ sont dues à un tiers des graphes, soit $\#5$, $\#7$, $\#8$, $\#12$, $\#15$, $\#18$ et $\#20$. De plus, pour certains graphes, l'erreur redevient importante et le reste à partir de $\lambda = 0.4$.

Observons enfin la plus grande et la plus petite communauté qui sont des indicateurs structurels importants, car ils peuvent révéler de manière significative la limite de résolution. Une remarquable manifestation de ce phénomène apparaît sur la figure 5.3 où l'on voit qu'avec Louvain ($\lambda = 0$), la taille de plus petite communauté est huit fois plus grande que la taille attendue, ce qui traduit bien la disparition des petites communautés. La valeur la plus juste, proche du ratio 1, est obtenue pour $\lambda = 0.2$. Ce ratio diminue pour des valeurs de λ plus grandes, aggravant l'erreur. Un autre enseignement, tout aussi important, est fourni par le ratio de plus grande communauté qui dénote une erreur importante (valeur trois fois plus grande) quelque soit le paramètre λ et même dès la première phase VM. Ainsi, l'heuristique VM souffre de la limite de résolution en constituant des communautés trop grandes, erreur qui ne peut pas être réparée par la seconde phase de fusion. Notons que cette erreur ne se manifeste pas pour tous les graphes, mais seulement pour $\#7$, $\#8$, $\#10$, $\#11$, $\#12$, $\#13$, $\#15$, $\#16$ et $\#18$.

5.3.2 Graphes réels

Avec les figures 5.4 et 5.5, qui représentent tous les indicateurs et scores que nous avons choisis pour les 22 graphes réels, nous constatons que l'évolution de la modularité par rapport à λ est l'exacte inverse de celle de la densité moyenne. La modularité décroît logiquement à mesure que la condition de fusion s'intensifie (λ croissant) alors que la densité moyenne et la densité minimale croissent. Cela traduit bien le phénomène de limite de résolution, car la modularité maximale obtenue avec $\lambda = 0$ ne conduit pas à la meilleure densité de communauté qui baisse du fait de la présence de très grandes communautés. Nous constatons également que le score

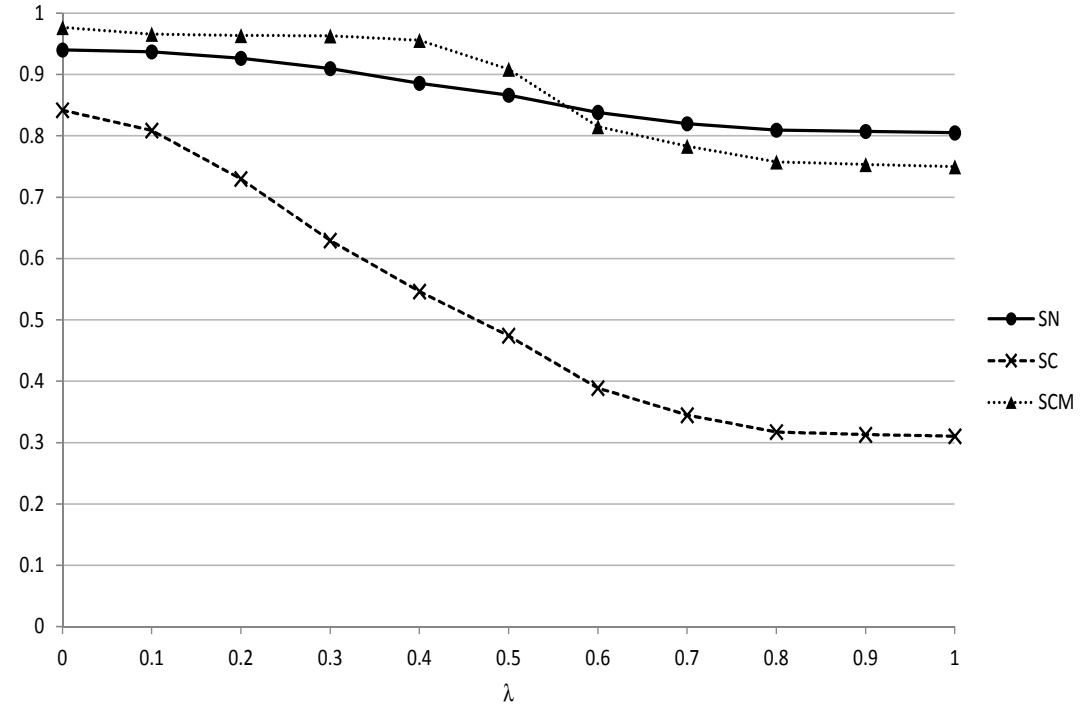


FIGURE 5.4 – Scores moyens des graphes réels sous condition de fusion, selon le paramètre λ. Les moyennes, sur l'ensemble des 22 graphes réels de test, des scores de noeuds (SN), de communauté (SC) et de communauté minimal (SCM) sont représentées par rapport au paramètre λ.

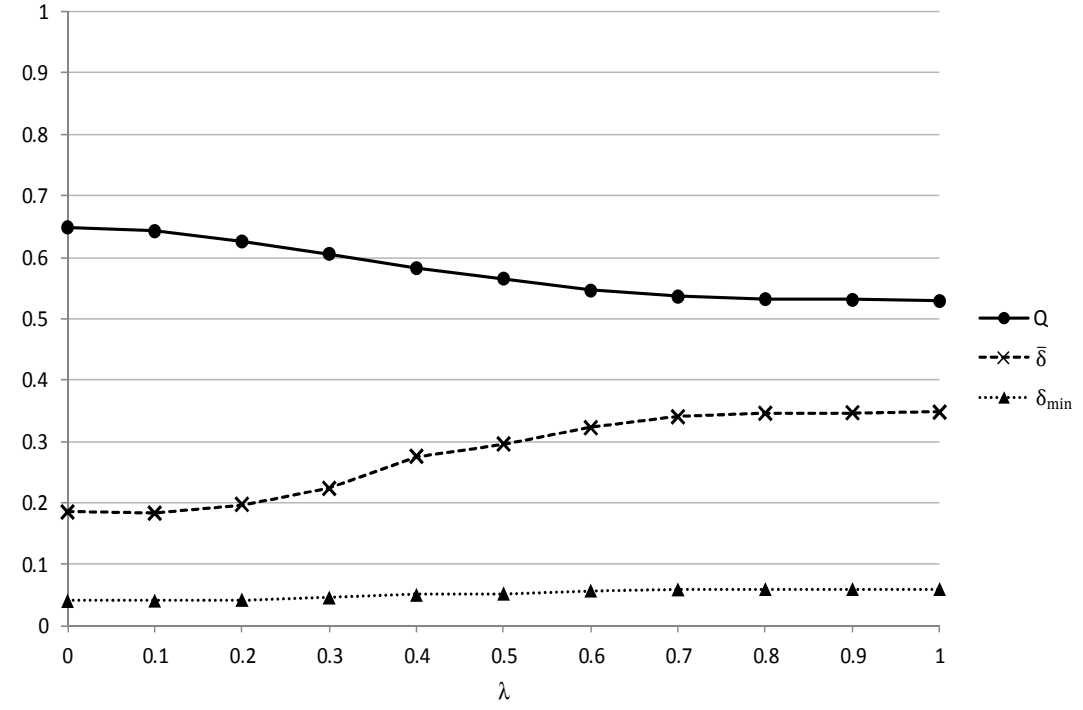


FIGURE 5.5 – Autres indicateurs moyens pour les graphes réels sous condition de fusion, selon le paramètre λ. Les moyennes, sur l'ensemble des 22 graphes réels de test, de la modularité, de la densité moyenne et de la densité minimale sont représentées par rapport au paramètre λ.

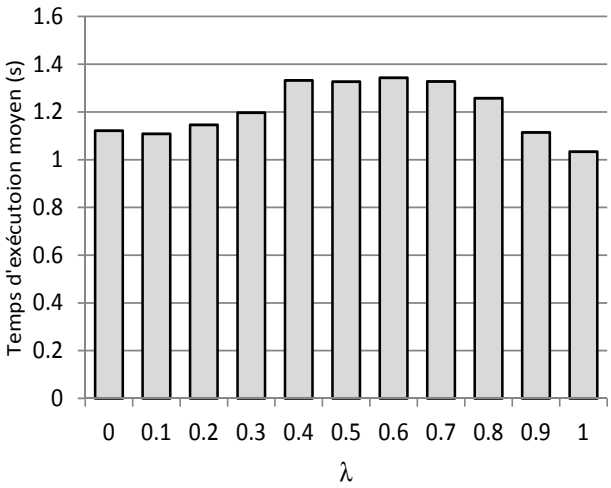


FIGURE 5.6 – Temps moyen d'exécution de Louvain+ sous condition de fusion appliquée aux graphes réels, selon le paramètre λ.

SN semble directement lié à la modularité, de part une évolution conjointe, en baisse lorsque λ croit. Le score SCM baisse également, mais plus rapidement. Nous avons remarqué le même phénomène avec les graphes artificiels pour lesquels, lorsque λ va de 0 à 0.3, ces deux scores diminuent alors que la NMI augmente. Il est donc probable, du fait de la limite de résolution, que l'optimisation de SN et SC, comme Q, ne conduise pas aux partitions les plus justes. En revanche, le critère de communauté au sens le plus faible (SCM), qui représente la part des communautés bien définies au sens le plus faible, est intéressant, car il demeure constant et proche de 1 entre $\lambda = 0$ et $\lambda = 0.4$, la baisse étant nettement amorcée à partir de 0.4. En moyenne, dans cette plage, les communautés existent et sont bien définies au sens le plus faible. En toute rigueur, nous devons rechercher la meilleure valeur du critère SCM, car c'est une condition minimale d'existence de communautés. Ce maximum est obtenu pour $\lambda = 0$ et se maintient pour la plupart des graphes jusqu'à une valeur de λ comprise entre 0.2 et 0.6. Ce principe peut constituer une façon d'utiliser la condition de fusion, que nous aborderons par la suite, en exécutant l'algorithme Louvain+ sous condition de fusion avec différentes valeurs de λ, en partant de 0 et en augmentant λ jusqu'à ce que le score SCM commence à baisser. Parmi toutes les partitions obtenues qui vérifient le critère minimal d'existence de communauté, celle qui a une valeur optimale, pour un critère autre que le modularité, est choisie. Ce critère de sélection peut être la densité, minimale ou maximale.

5.3.3 Temps d'exécution et complexité

Les éléments d'évaluation du critère de fusion, $l(C)$, $l(C')$ et $l(C, C')$ étant présents sous forme de poids d'arête dans les graphes contractés, la complexité de l'algorithme Louvain+, constatée en $O(m)$, n'est pas affectée par ce critère. Le temps de calcul est tout de même augmenté par l'évaluation du critère, mais aussi réduit par la réduction du nombre de fusions. Une estimation de la résultante des deux actions est très délicate, mais la figure 5.6 donne une indication du temps d'exécution en fonction de la valeur de λ sur les graphes réels. Au pire, il est augmenté de 14 % pour $\lambda = 0.6$ par rapport à Louvain+ sans condition de fusion.

5.4 Impact sur les défauts de la modularité

5.4.1 Limite de résolution

La limite de résolution de la modularité a été illustrée par ses auteurs avec un type de graphe très particulier dans lequel des cliques de même taille sont reliées deux à deux par une seule arête. Si le nombre total d'arêtes du graphe est suffisamment grand par rapport à la taille des cliques, la partition naturelle où chaque clique forme une communauté a une modularité inférieure à une partition où chaque paire de cliques voisines forme une communauté (voir une illustration dans la figure 1.3).

Cherchons à quelle condition le critère de fusion est faux dans le cas de deux communautés C et C' reliées par seulement une arête. À partir de l'écriture 5.8 du critère de fusion, avec la condition $l(C, C') = 1$, soit $e = l(C) + l(C') + 1$ et en posant $l(C') = h + l(C)$, nous obtenons la condition suivante pour que le critère de fusion soit faux :

$$\begin{aligned}&(4\lambda l(C) + 2\lambda - 2)l(C') + 2(\lambda - 1)l(C) + \lambda - 2 \geq 0\\&(4\lambda l(C) + 2\lambda - 2)h + 4\lambda l(C)^2 + 4(\lambda - 1)l(C) + \lambda - 2 \geq 0\end{aligned} \tag{5.12}$$

Plaçons-nous dans le cas où C est la plus petite communauté sans perte de généralité (inversion de C' et C dans le cas contraire), donc si $h \geq 0$. L'expression du critère est de la forme $Ah + B$ avec $A = 4\lambda l(C) + 2\lambda - 2$ et $B = 4\lambda l(C)^2 + 4(\lambda - 1)l(C) + \lambda - 2$. Dans l'hypothèse où $h = 0$, le signe de B est le signe de $(2l(C) + 1)(2\lambda l(C) + \lambda - 2)$ par factorisation, soit le signe de $(2\lambda l(C) + \lambda - 2)$ puisque $2l(C) + 1$ est positif. Donc, dans l'hypothèse où h est nul, le critère de fusion est faux si $(2\lambda l(C) + \lambda - 2) \geq 0$, soit $l(C) \geq 1/\lambda - 1/2$. Si maintenant h augmente (h est positif), la fonction $Ah + B$ doit être croissante pour qu'elle reste positive, donc A doit être positif, soit $l(C) \geq 1/(2\lambda) - 1/2$. Des deux conditions, $l(C) \geq 1/\lambda - 1/2$ et $l(C) \geq 1/(2\lambda) - 1/2$, la première est plus stricte, car $\lambda > 0$.

Le critère de fusion ne s'appuie que sur le nombre d'arêtes de la communauté C et non sur sa densité. Pour imposer que la fusion soit interdite et donc que le critère soit faux, si les deux communautés à fusionner sont reliées par seulement une arête, il faut que la plus petite communauté C vérifie $l(C) \geq 1/\lambda - 1/2$. Si λ s'approche de zéro, la condition n'est vraie que pour des valeurs de $l(C)$ de plus en plus grandes, ce qui n'est pas souhaitable. Si l'on souhaite interdire les fusions pour toute communauté telle que $l(C) >= K$, il faut $K \geq 1/\lambda - 1/2$, soit $\lambda \geq 2/(2K + 1)$. Une valeur raisonnable est $K = 3$ qui exige $\lambda \geq 2/7 \approx 0.285$.

La fusion est ainsi interdite si la communauté C contient au moins $K = 3$ arêtes. Il est important de noter que C n'est pas nécessairement une clique ou une quasi-clique et peut s'en éloigner, avec une densité en baisse, si le nombre de nœuds est plus important que dans une clique. Pourtant, la fusion avec une communauté C' sera toujours interdite, du moment que C et C' sont reliées par une seule arête. Cette interdiction peut être indue d'autant plus que C est peu dense. Nous touchons là un point important concernant l'implantation de cette condition de fusion dans un algorithme. Il est indispensable que les communautés soumises à ce critère de fusion soient "bien formées" avec une densité aussi forte que possible. En cela, l'application du critère à un algorithme qui procède d'abord par déplacement de nœuds (procédure VM) puis par fusion, comme Louvain, est recommandée. Alors qu'un algorithme comme FastGreedy qui procède à des fusions dès le départ, avec des communautés minuscules et faiblement définies, ne gagnera pas à utiliser la condition de fusion.

5.4.2 Inconsistance de solution

La figure 5.7 montre clairement que les solutions obtenues par l'algorithme de Louvain avec condition de fusion sont structurellement plus similaires entre elles à mesure que λ augmente.

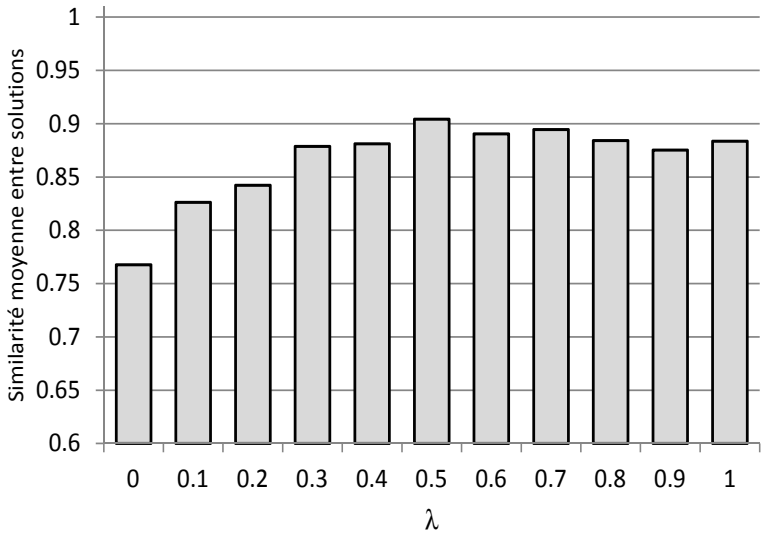


FIGURE 5.7 – Inconsistance de solution de la condition de fusion mesurée par la similarité moyenne entre solutions sur les graphes réels, selon le paramètre λ. La version stochastique de l'algorithme est exécutée 20 fois sur chaque graphe. La similarité moyenne est calculée comme la moyenne de la NID de toutes les paires distinctes de solutions. Enfin, la moyenne pour tous les graphes est représentée sur ce graphique. Ainsi une similarité proche de 1 indique des solutions structurellement proche et donc une inconsistance liée à la modularité réduite.

Pour la valeur appropriée de λ, soit environ 0.3, la similarité moyenne fondée sur la NID est de 0.879 au lieu de 0.768 pour l'algorithme Louvain+. L'inconsistance entre solutions et donc fortement réduite par l'application de la condition de fusion.

Conclusion

La condition de fusion $Q_\lambda(C, C') < 1 - \lambda$ agit efficacement pour discriminer les communautés C et C' à fusionner, dans l'objectif d'optimiser la modularité standard Q. Son application dans la phase de contraction de graphe de l'algorithme Louvain+, avec un paramètre λ compris entre 0.2 et 0.3, améliore la justesse des partitions en réduisant les effets de la limite de résolution. La taille des communautés est plus juste et en particulier les petites communautés réapparaissent. L'inconsistance de solution est également réduite avec une valeur moyenne (environ 0.88) presque aussi bonne que cela obtenue avec les algorithmes mémétiques, pour un temps d'exécution qui demeure linéaire. La principale insuffisance réside dans la taille de certaines grandes communautés, trop importante dès la première phase VM. Ces communautés demeurent dans la partition finale, car la seconde phase ne procède que par agglomération. Il est nécessaire, pour obtenir un algorithme complet, de traiter le problème des communautés de taille trop importante, constituées par l'heuristique VM.

6

Algorithme par raffinement multi-résolution

Sommaire

La condition de fusion, qui agit dans la deuxième phase de contraction de l'algorithme Louvain+, est intéressante pour s'approcher d'une taille de communauté juste, mais elle ne peut pas corriger les erreurs de communautés trop grandes résultant de la première phase VM. Nous proposons d'améliorer cette première phase en exploitant la faculté de la modularité paramétrique, comme fonction d'optimisation, de réduire la taille des communautés. Dans la version présentée dans ce chapitre, la première phase de Louvain+, au lieu de procéder à une seule exécution de VM à l'échelle de la modularité standard (facteur 1), réalise une succession de VM avec la modularité paramétrique, en allant en décroissant d'un facteur plus élevé jusqu'à 1. Cette amélioration présente de très bons résultats pour les graphes artificiels (16 graphes sur 20) et plus mitigés pour les graphes réels (baisse ou monotonie de SCM pour 6 graphes). Elle fournit des informations qui nous seront utiles pour formuler et valider les idées présentées dans le dernier chapitre.

Introduction

Les expériences d'application du paradigme de multirésolution à la détection de communautés consistent à utiliser une forme de modularité intégrant un facteur d'échelle λ qui détermine directement l'échelle de détection des communautés et indirectement la taille de celles-ci. Il est possible de déterminer au préalable une valeur d'échelle selon le graphe à partitionner (les méthodes de détermination de ce facteur sont nombreuses [99]) ou bien de tester toute une plage de valeurs et de définir un critère de validation des partition obtenues. Les auteurs de la modularité paramétrique, que nous avons utilisée pour la condition de fusion, emploient cette dernière technique en observant des plages de stabilité du nombre de communautés. Avec la modularité paramétrique, l'échelle λ joue sur la taille et donc sur le nombre des communautés que l'on observe : plus λ est faible, plus les grandes communautés sont les seules détectées et à mesure que λ augmente, des communautés de plus en plus petites apparaissent. Aux extrêmes, lorsque $\lambda = 0$ le graphe complet forme une seule communauté et lorsque $\lambda \to \infty$, chaque communauté ne contient qu'un seul nœud. La valeur particulière $\lambda = 1$ correspond à la modularité classique. Dans une autre forme de méthode multirésolution, introduite par Arenas et al.[71], des boucles de poids r sont ajoutées à chaque nœud et le facteur de résolution r a le même effet que λ pour la modularité paramétrique.

Malheureusement, bien que ses effets soient amoindris, la limite de résolution entache également ces méthodes multirésolution [100, 101]. Il n'y a pas en général une seule échelle qui capture avec justesse et complétude la structure modulaire d'un réseau complexe et qui permette de voir en même temps les petits et les grandes communautés [101].

Par ailleurs, nous avons constaté l'efficacité de l'algorithme Louvain+ avec condition de fusion, mais aussi ses faiblesses. La première phase VM souffre partiellement de la limite de résolution en détectant des communautés trop grandes. La seconde phase de contraction avec condition de fusion ne peut pas corriger ce défaut puisqu'elle prévient simplement les fusions de paires de communautés incorrectes. Notre idée, pour améliorer la première phase, est de réaliser une succession de procédure VM avec Q_λ comme fonction à maximiser, en partant d'une valeur pour λ fixée au-dessus de 1 et en la faisant décroître à chaque exécution de VM jusqu'à 1. L'objectif est de constituer des petites communautés en commençant par un λ assez élevé et d'augmenter ensuite leur taille en réduisant l'échelle λ. Ce principe s'appuie sur le fait que la taille des communautés détectées est bornée supérieurement selon la valeur d'échelle λ [102].

A chaque nouvelle exécution de VM, la partition de départ est celle résultant de l'exécution précédente de sorte que la structure incomplète trouvée précédemment s'impose aux exécutions suivantes. Nous allons montrer expérimentalement que le résultat final est meilleur qu'en appliquant uniquement VM avec $Q = Q_1$, même si nous faisons décroître λ jusqu'à 1. Nous pourrons aussi examiner l'idée de stopper le raffinement à un λ plus grand que 1 pour fournir à la seconde phase de Louvain+ des communautés plus petites.

6.1 Algorithme

L'algorithme, que nous nommons MR-Come pour *Multi resolution Conditional Merge*, est un prolongement et une amélioration de l'algorithme Louvain+ auquel la condition de fusion $Q_\lambda < 1 - \lambda$ est appliquée, décrite au chapitre précédent. La première phase VM est remplacée par une phase dans laquelle plusieurs étapes de raffinement de la partition singleton initiale s'enchaînent, chaque partition résultant d'une étape étant prise comme partition de départ de l'étape suivante. Chaque étape est une exécution de l'heuristique VM avec la modularité paramétrique Q_β comme objectif de maximisation. Nous notons cette variable β pour la distinguer du paramètre λ de l'algorithme avec condition de fusion qui utilise lui aussi la modularité pa-

ramétrique. À la première étape, β démarre à une valeur de départ à déterminer, notée β_1. Pour passer d'une étape i à la suivante, β décroît par une formule récurrente $\beta_{i+1} = k(\beta_i)$, k étant une fonction strictement décroissante telle que $k : \mathbb{R}^+ \longmapsto \mathbb{R}^+$. Enfin, la dernière étape est fixée lorsque β_i passe sous un seuil β_s à déterminer préalablement. La partition résultant de la dernière étape est ensuite utilisée par la seconde phase de contraction de l'algorithme Louvain+.

Algorithme 3 Pseudo-code de l'algorithme MR-Come

Require: Graphe $G = (V, E)$.
Ensure: Une partition $\mathcal{P}$ de G avec une modularité maximale.
1: $\beta \leftarrow \beta_1$
2: $\mathcal{P} \leftarrow$ PartitionnerEnSingleton(G) /* Un nœud par communauté */
3: **repeat**
4: $\quad \mathcal{P} \leftarrow$ VertexMover$(\mathcal{P}, \beta)$
5: $\quad \beta \leftarrow \beta - 1$ /* version Mr-Come/de, $\beta/2$ pour Mr-Come/ch */
6: **until** $\beta < \beta_s$

L'idée, en démarrant à une grande échelle, est de déterminer des petites structures communautaires fortes qui, en réduisant progressivement l'échelle, vont grossir sous la contrainte de leurs sous-structures denses préalablement établies. Les très grandes communautés qui se forment facilement avec la modularité standard, d'autant plus que la structure communautaire est mal définie ou que le graphe est très peu dense, auront plus de difficulté à se former à partir de plus petites communautés déjà établies. Nous avons constaté empiriquement que l'application de la procédure VM avec Q_1 conduit à une structure différente si elle directement appliquée à la partition singleton ou si elle est appliquée à une partition issue par exemple des raffinements successifs avec Q_4, puis Q_3 et enfin Q_2. Les tests qui suivent ont pour objectif de mesurer l'efficacité de cette méthode pour réduire les effets de la limite de résolution.

6.2 Validation expérimentale

L'algorithme MR-Come possède 3 paramètres à fixer préalablement à toute exécution, λ le seuil de la condition de fusion, β_1 et β_s les bornes de variation de l'échelle des raffinements successifs. La valeur du paramètre λ est fixée pour tous les tests à 0.285, conformément aux conclusions du chapitre précédent. Nous devons également déterminer de quelle façon l'échelle β décroît pour passer de β_1 à β_s. Nous avons adopté deux fonctions simples de décroissance, la première, décrémentale, consistant à enlever 1 à chaque étape, la seconde, dichotomique, consistant à diviser β par 2 à chaque étape. Les versions d'algorithme sont nommées MR-Come/de pour la décroissance décrémentale de β, et MR-Come/ch pour la décroissance dichotomique. Cet algorithme s'appuie sur l'algorithme Louvain+ avec condition de fusion, que nous nommons Come.

6.2.1 Test de paramètre

L'objectif des tests réduits présentés ici est de fixer une valeur pour chaque paramètre et chaque version, utilisée ensuite dans les tests de performance détaillés. Ces tests réduits portent uniquement sur les graphes artificiels, qui permettent une validation par la mesure de similarité avec les solutions, en exécutant 20 instances de la version stochastique de l'algorithme MR-Come.

Nos tests de paramètres utilisent les versions d'algorithme suivantes :

- la version nommée MR-Come/de de l'algorithme Come avec une décroissance decrémentale de β en première phase, testée d'une part avec $\beta_1 \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ et d'autre part avec $\beta_s \in \{1, 2, 3, 4\}$;

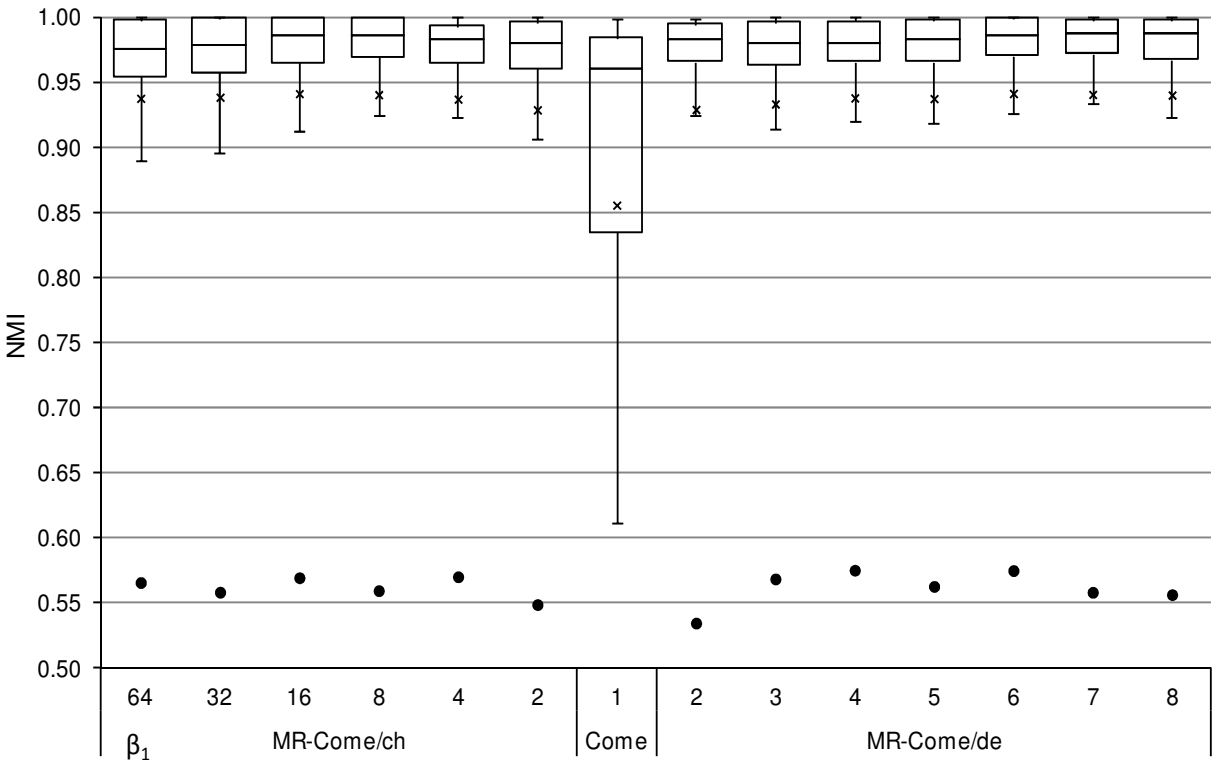


FIGURE 6.1 – Boîte à moustaches de MR-Come/ch et MR-Come/de exécutés sur les graphes artificiels pour différentes valeurs de β_1. Chaque boîte montre la dispersion des 20 valeurs de NMI moyenne associée à chaque graphe. Pour un graphe, la moyenne est calculée sur l'ensemble des 20 exécutions. La valeur atypique basse de $\beta_1 = 1$ est environ 0.29 et ne figure pas sur le graphique.

- la version nommée MR-Come/ch de l'algorithme Come avec une décroissance dichotomique de β en première phase, testée d'une part avec $\beta_1 \in \{1, 2, 4, 8, 16, 32, 64\}$ et d'autre part avec $\beta_s \in \{1, 2, 3, 4\}$.

Nous comparons ces versions avec l'algorithme Come (Louvain+ avec condition de fusion). Tous ces algorithmes utilisent le paramètre $\lambda = 0.285$. Nous avons choisi de tester en première le paramètre β_1 qui est sans doute le plus important, indépendamment du paramètre β_s, moins crucial.

Tests généraux

La figure 6.1 montre les résultats d'exécution de MR-Come/de et MR-Come/ch avec $\beta_s = 1$ et différentes valeurs de β_1. L'apport des raffinements successifs à différentes échelles est net dès $\beta_1 \geq 2$. Les performances de la version décrémentale sont légèrement meilleures que celles de la version dichotomique, surtout pour les grandes valeurs de β_1. Pourtant, il est difficile de juger que de plus grandes valeurs de β_1 donnent de meilleures partitions pour la version MR-Come/de, tellement les caractéristiques statistiques de chaque échantillon, entre $\beta_1 = 2$ et $\beta_1 = 8$, sont proches. Pour MR-Come/ch, la version dichotomique, les performances paraissent moins bonnes avec pour $\beta_1 = 16$ et au-delà.

La figure 6.2 représente les ratios d'erreur k, $min|C_i|$ et $max|C_i|$ et le score SCM, dont la valeur idéale est 1, pour trois valeurs de β_s. Les comportements de MR-Come/de et MR-Come/ch sont très proches pour toutes ces mesures de performance. Examinons tout d'abord

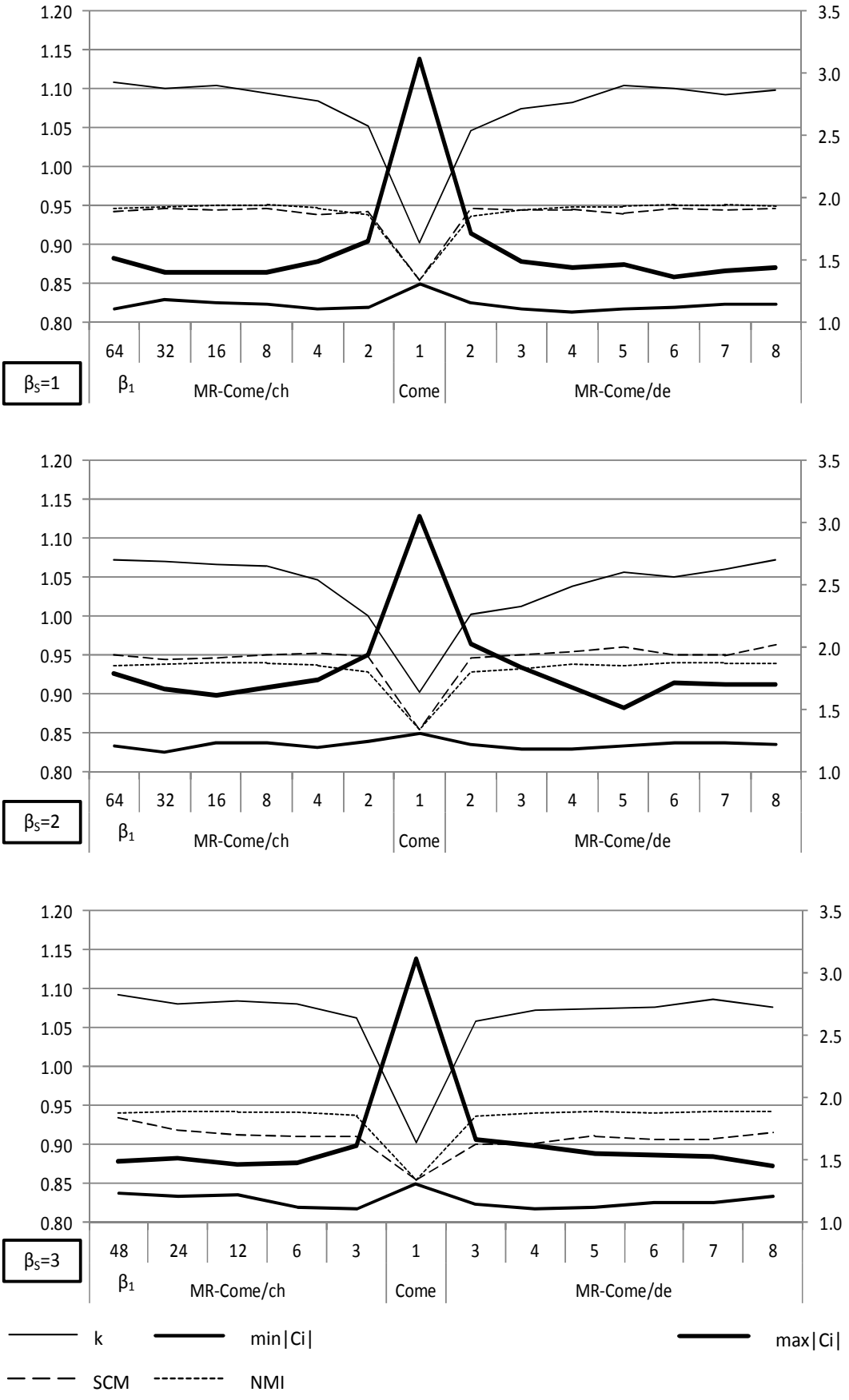


FIGURE 6.2 – Ratios d'erreur et score SCM de MR-Come/ch et MR-Come/de selon β_1 et β_s. La moyenne de chaque ratio ou score est calculée sur les 20 instances et l'ensemble des graphes, pour différentes valeurs de β_1 (représentées en abscisse) et pour trois valeurs de β_s (un graphique pour chacune). L'axe des ordonnées de droite se rapporte au ratio d'erreur $max|C_i|$ et l'axe de gauche à toutes les autres valeurs.

$\beta_s = 1$ avec le premier graphique. Le ratio du nombre de communautés à 0.9 pour Come, croit avec β_1 jusqu'à environ 1.07. La valeur parfaite est atteinte pour $\beta_1 = 2$. La taille de plus petite communauté est peu impactée par les raffinements multirésolutions. Elle se dégrade légèrement avec ce raffinement par rapport à Come, quel que soit la valeur de β_1, en passant de 0.849 à 0.825 au pire. L'impact majeur est visible sur la taille de plus grande communauté, dont le ratio dépasse 3 avec Come et descend jusqu'à près de 1.6 avec MR-Come/ch et $\beta_1 = 16$ et 1.5 avec MR-Come/ch associé à $\beta_1 = 5$. Les mesures de NMI et SCM semblent corrélées et montrent une amélioration significative avec environ 0.95, pour toutes les valeurs de β_1, par rapport à la référence Come qui donne 0.85.

Dans l'ensemble, avec un ratio k très légèrement dégradé et des ratios $min|C_i|$ et $max|C_i|$ optimaux, un bon compromis est obtenu avec $8 \leq \beta_1 \leq 16$ pour MR-Come/ch et $4 \leq \beta_1 \leq 6$ pour MR-Come/de.

Les deux derniers graphiques de la figure 6.2 montrent l'impact d'une dernière valeur de résolution β_s supérieure à 1. Les communautés sont globalement plus petites ce que traduit la baisse des ratios $min|C_i|$ et $max|C_i|$ et l'augmentation du nombre de communautés. Le score de communauté minimal SCM est impacté négativement avec $\beta_s = 3$. Ces résultats tendent à montrer qu'une valeur supérieure de β_s a un effet négatif sur la structure des partitions détectées. Nous pouvons conjecturer que ce phénomène est dû aux communautés mal formées qui résultent de la première phase VM avec raffinements successifs et qui perturbent l'application de la condition de fusion en seconde phase, selon nos conclusions concernant l'algorithme Come.

Ces résultats qui agrègent des indicateurs pour l'ensemble des graphes peuvent cacher des disparités selon la structure et la difficulté de partitionnement de chaque graphe. Ils masquent également les disparités de résultats entre instances d'exécution pour un même graphe. Il est préférable d'affiner nos conclusions en observant en détail certains graphes sélectionnés pour leurs propriétés.

Graphes

Nous souhaitons détailler nos tests avec des graphes représentatifs des caractéristiques particulières des réseaux complexes ou présentant simplement une difficulté de partitionnement. Nous avons choisi les six graphes suivants :

- #5 : petit graphe dont les communautés sont faiblement définies (paramètre μ du modèle LFR égal à 0.6) ;
- #7 : graphe très peu dense ayant une large étendue des degrés de nœud et des tailles de communauté et qui met en échec l'algorithme de Louvain (NMI moyenne d'environ 0.44) ;
- #10 : graphe qui possède uniquement des communautés de grande taille (entre 510 et 988 nœuds par communauté) ;
- #12 : graphe très dense (degré de nœud entre 29 et 100, 50 en moyenne) mettant en échec l'algorithme de Louvain (NMI moyenne d'environ 0.47) ;
- #15 : graphe aux communautés très faiblement définies ($\mu = 0.8$) ;
- #16 : graphe aux communautés faiblement définies ($\mu = 0.6$) et ayant une grande queue de distribution de taille de communautés qui donne un grand nombre de petites communautés.

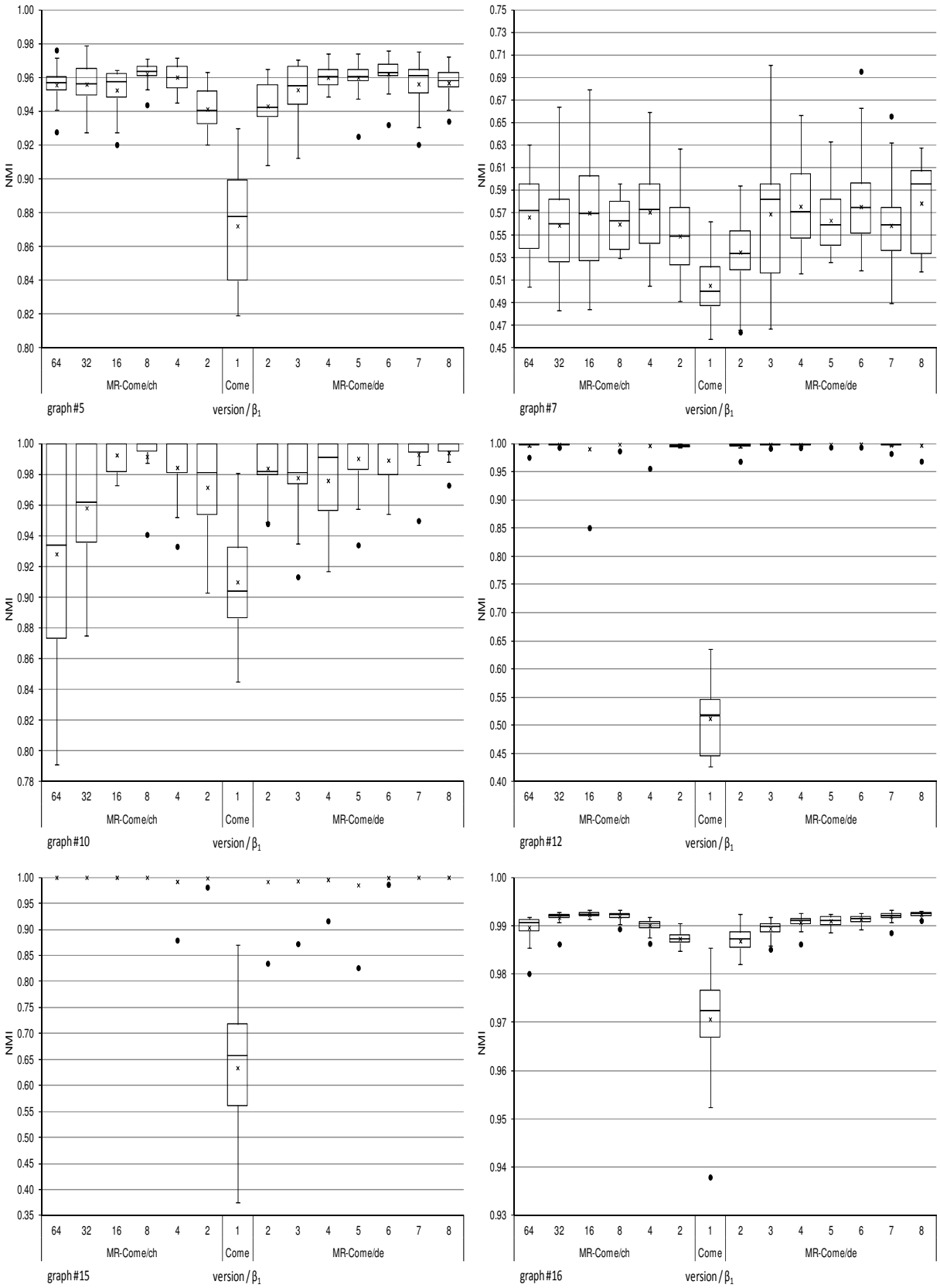


FIGURE 6.3 – Boîte à moustaches de NMI pour 6 graphes représentatifs, traités par MR-Come/ch et MR-Come/de. Les boîtes permettent de comparer la répartition de NMI moyenne calculée sur 20 exécutions pour les trois algorithmes Come, MR-Come/ch et MR-Come/de et pour différentes valeurs de β_1.

Les résultats sont présentés sur la figure 6.3. Pour le graphe $\#5$, l'amélioration est nette dès $\beta_1 = 2$, quel que soit la version, et encore meilleur à partir de β_1 égal à 3 ou 4. En $\beta_1 = 8$ pour MR-Come/ch et $\beta_1 = 6$ pour MR-Come/de, la distribution de NMI est la plus étroite et la plus haute de sorte que toutes les valeurs sont au-dessus du maximum pris par Come. La situation se dégrade pour les valeurs supérieures de β_1 tout en restant meilleure que pour $\beta_1 = 2$. Les résultats de $\#7$ sont similaires bien que l'amélioration soit moins nette, car les meilleures distributions chevauchent celle de Come. Toutefois, en valeur absolue de NMI, nous constatons que les meilleures partitions trouvées sont loin d'être justes avec un maximum de 0.7 et une valeur médiane inférieure à 0.6. Les résultats de $\#16$ sont semblables à ceux de $\#5$ et $\#7$ bien que l'optimum de MR-Come/ch soit $\beta_1 = 16$ et celui de MR-Come/de, $\beta_1 = 8$.

La situation du graphe $\#10$ est intéressante, car si la justesse des partitions augmente pour les valeurs croissantes de β_1 avec MR-Come/ch, elle se dégrade fortement pour $\beta_1 = 32$ et plus encore pour $\beta_1 = 64$, alors que l'amélioration se maintient pour les grandes valeurs de β_1 avec MR-Come/de. La méthode n'est sans doute pas en cause, mais plutôt les grandes valeurs de β_1 qui révèlent des communautés très petites alors que ce graphe n'en contient pas. Notons toutefois que dès la valeur $\beta_1 = 2$, quel que soit la version, la solution est trouvée au moins une fois parmi les 20 instances, avec une NMI de 1.

Enfin, les graphes $\#12$ et $\#15$ montrent des distributions quasi parfaites avec une étendue nulle et pratiquement toutes les valeurs de NMI à 1. Quelques valeurs atypiques subsistent, mais sont juste en-dessous du maximum obtenu par Come.

Ces résultats détaillés sur 6 graphes représentatifs montrent que l'algorithme par raffinement multirésolution n'est pas en général fortement sensible au paramètre β_1, mais peut l'être d'autant plus que les communautés sont grandes, car les grandes valeurs de ce paramètre ont tendance à faire apparaître des petites communautés pas nécessairement pertinentes. L'amélioration de la justesse des partitions est nette pour $\beta_1 = 2$ et augmente généralement avec β_1 jusqu'à un optimum aux alentours de 8 pour MR-Come/ch et 6 pour MR-Come/de. De plus, en augmentant β_1 jusqu'à cet optimum, la distribution de NMI se réduit.

En conclusion pour les paramètres, $\beta_1 = 8$ convient bien à MR-Come/ch et $\beta_1 = 6$ à MR-Come/de, avec pour les deux $\beta_s = 1$. La succession des échelles de raffinement préconisées pour les graphes artificiels testés est $\{8, 4, 2, 1\}$ pour MR-Come/ch et $\{6, 5, 4, 3, 2, 1\}$ pour MR-Come/de. Il est très difficile de départager les deux variantes, mais les résultats globaux de la figure 6.2 avec les valeurs de paramètres choisies donnent un léger avantage à MR-Come/de.

Temps d'exécution

Le temps d'exécution moyen selon la version d'algorithme et selon le paramètre β_1 est représenté en figure 6.4. Assez logiquement, l'heuristique VM ayant généralement une complexité en temps linéaire, le temps d'exécution est proportionnel aux nombres de raffinements, quel que soit la version MR-Come/ch ou MR-Come/de.

6.2.2 Graphes artificiels

A ce stade de notre exposé, nous souhaitons entrer dans le détail des mesures de chaque graphe, pour comparer les performances selon les caractéristiques des graphes, mais aussi pour apprécier la dispersion des valeurs mesurées sur les 100 instances du protocole d'expérimentation.

Tous les résultats qui suivent dans ce chapitre portent uniquement sur la version MR-Come/de et sont obtenus avec le seuil de condition de fusion $\lambda = 0.285$, et la séquence de résolutions $\{6, 5, 4, 3, 2, 1\}$ pour la première phase de raffinements successifs.

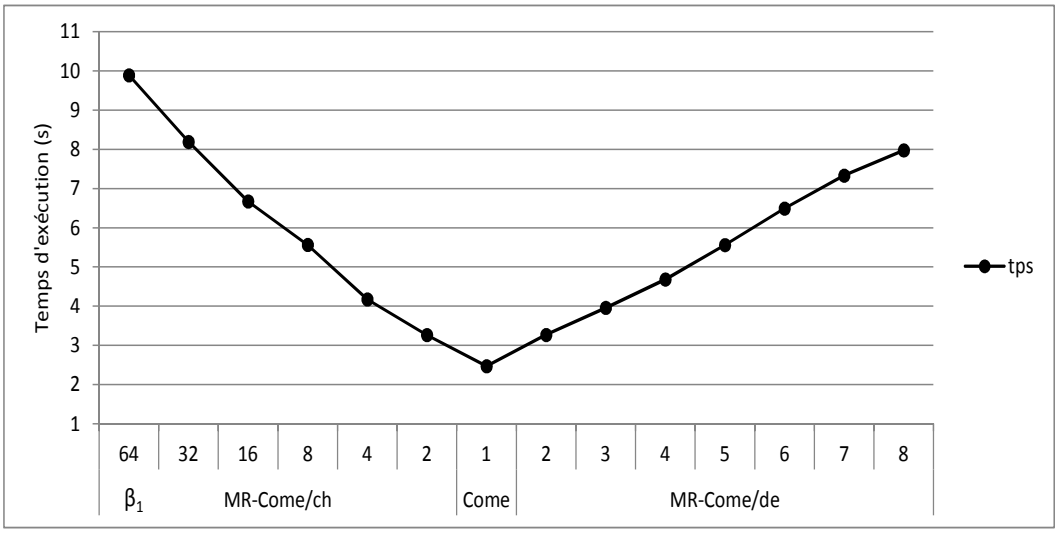


FIGURE 6.4 – Temps d'exécution moyen calculé sur 20 instances d'exécution et sur l'ensemble des graphes artificiels, pour différentes valeurs de β_1 et pour les versions MR-Come/ch et MR-Come/de. La valeur $\beta_1 = 1$, quel que soit la version, correspond à l'algorithme Come.

Algorithme Louvain+ vs Come

La figure 6.5 compare la distribution de NMI entre Louvain+ et Come pour chacun des 20 graphes artificiels. L'amélioration dans la justesse des partitions apportée par Come est nette, avec une distribution obtenue par Come au-dessus de celle de Louvain, hors valeurs atypiques, pour tous les graphes à l'exception des plus difficiles que sont les numéros #5, #7, #10, #12, #15 et #18. Parmi ceux-ci, seuls #10 et #15 ne voient aucune amélioration avec la condition de fusion de Come, les autres ayant une distribution plus haute, mais chevauchant celle de Louvain+. Pour ces 6 graphes difficiles, l'étendue de la distribution de NMI n'est pas réduite par Come.

Ces graphes ont en commun, à l'exception de #10 qui a une forme très particulière, d'avoir un paramètre μ élevé signifiant une appartenance faible des nœuds aux communautés. Dans cette situation, la limite de résolution se manifeste amplement dès la première phase de l'algorithme de Louvain provoquant des erreurs que la condition de fusion ne peut pas corriger et justifiant le recours aux raffinements successifs de l'algorithme MR-Come/de. Le graphe 10 ne contient que des grandes communautés, de tailles comprises entre 510 et 988. Dans ce cas également, nous avons relevé des erreurs dès la première phase VM dues à la limite de résolution. L'explication d'une NMI entre 0.8 et 1, non améliorée par Come, tient au fait que l'impact de ces erreurs est amplifié par la taille des communautés, deux communautés pouvant être fusionnées dans la phase VM par déplacement de tous les nœuds de l'une d'entre elles vers l'autre.

Algorithme Come vs MR-Come/de

La figure 6.6 compare la distribution de NMI entre Come et MR-Come/de pour chacun des 20 graphes artificiels. Parmi les graphes qui résistent à Come, soit #5, #7, #10, #12, #15 et #18, seuls #7 et #18 ont une NMI faible avec MR-Come/de, bien que celle-ci soit améliorée par rapport à Come. Tous les autres graphes, à l'exception de #11 qui atteint 0.8, ont une NMI comprise entre 0.9 et 1 en faisant abstraction des valeurs atypiques (seul #15 a une valeur atypique entre 0.85 et 0.9).

Pour expliquer la résistance des graphes #7 et #18 face à l'algorithme MR-Come/de, nous pouvons avancer l'hypothèse, selon nos observations de distribution de tailles de communauté,

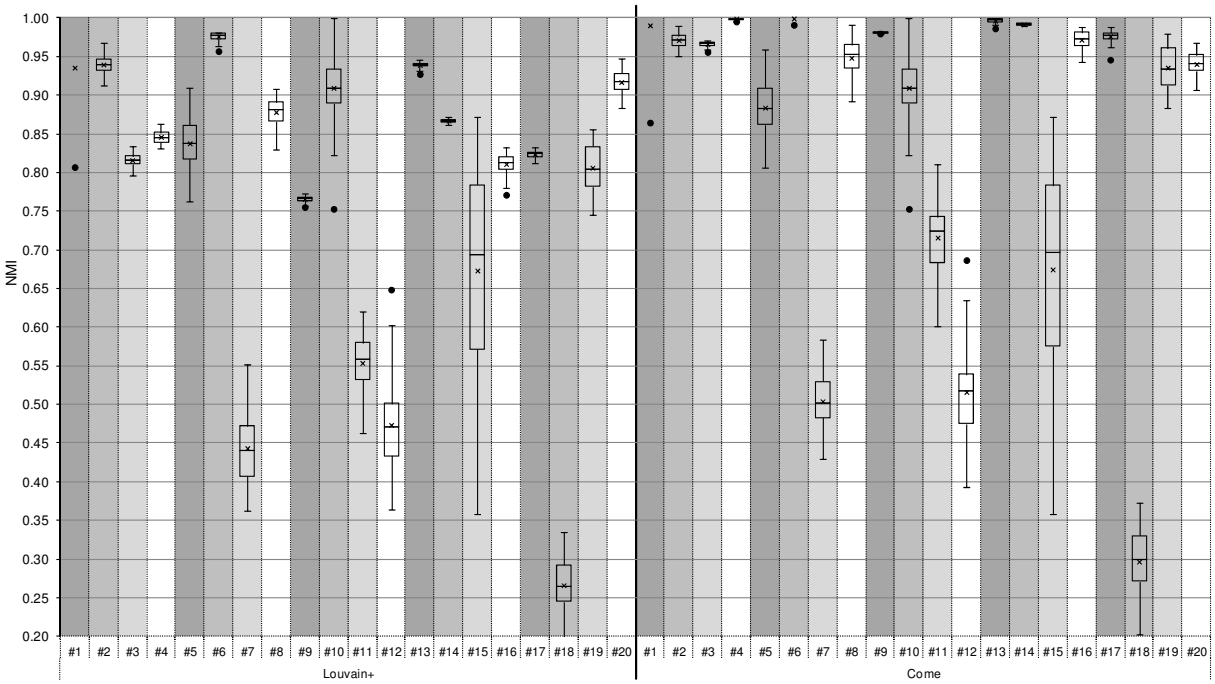


FIGURE 6.5 – Boîtes à moustaches de la NMI comparée entre Louvain+ et Come, pour les 20 graphes artificiels. Les paramètres sont $\epsilon_c = \epsilon_r = 10^{-5}$ et $\lambda = 0.285$. Le fond alterne quatre couleurs pour faciliter la lecture et les comparaisons graphe à graphe.

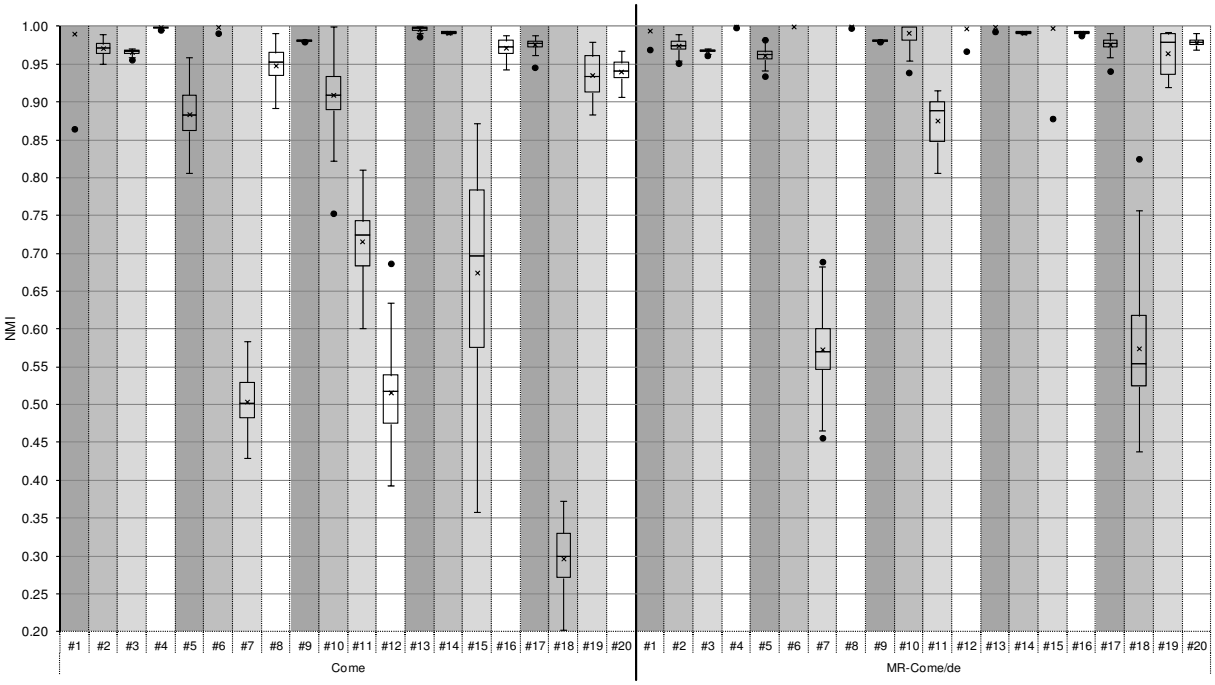


FIGURE 6.6 – Boîtes à moustaches de la NMI comparée entre Come et MR-Come/de, pour les 20 graphes artificiels. Les paramètres communs sont $\epsilon_c = \epsilon_r = 10^{-5}$. Les paramètres spécifiques de MR-Come/de sont $\beta_1 = 6$, $\beta_s = 1$. Le fond alterne quatre couleurs pour faciliter la lecture et les comparaisons graphe à graphe.

que les raffinements successifs ne suffisent pas à écarter la limite de résolution de la phase VM, du fait d'une appartenance aux communautés trop faible, qui conduit à la constitution de communautés trop grandes par un grand nombre de déplacements de nœuds. Toujours selon nos observations, ce phénomène concerne un grand nombre de communautés pour #18, car l'appartenance est originellement très faible ($\mu = 0.85$) et seulement quelques communautés pour #7, mais de grande taille, couvrant plus de 20 % des nœuds.

6.2.3 Graphes réels

Nous avons précédemment établi que l'optimisation de la modularité, par exemple avec l'algorithme mémétique MA-COM, améliore le score SN. Inversement, avec la condition de fusion qui dégrade légèrement la modularité, mais améliore la justesse des partitions en réduisant les effets de la limite de résolution, les scores SN et SC sont dégradés. Par ailleurs, les caractéristiques des graphes artificiels présentées dans le tableau 2.3 montrent que ces scores peuvent être très dégradés pour des graphes possédant néanmoins les caractéristiques de réseaux complexes et ayant en particulier une structure communautaire. Pour ces deux raisons, nous pensons qu'il est nécessaire à ce stade d'abandonner ces deux mesures pour les graphes réels en ne conservant que la mesure SCM d'existence minimale des communautés. Tous les graphes artificiels ont un score SCM compris ente 0.99 et 1.

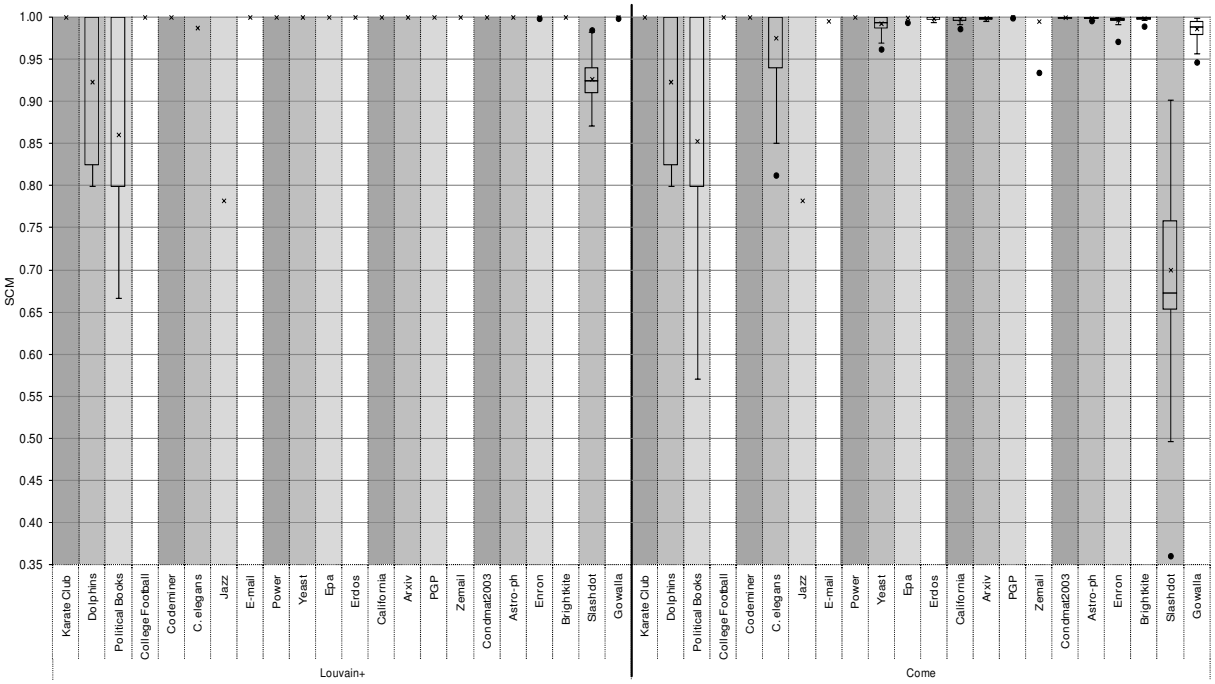


FIGURE 6.7 – Boîtes à moustaches de distribution, sur les 100 instances du protocole d'expérimentation, de la mesure SCM pour chaque graphe réel et pour les algorithmes Louvain+ et Come. Les paramètres communs sont $\epsilon_c = \epsilon_r = 10^{-5}$ et celui spécifique de Come est $\lambda = 0.285$.

Les figures 6.7 et 6.8 présentent l'étendu de la mesure SCM pour les algorithmes Louvain+, Come et MR-Come/de. Nous remarquons, entre Louvain+ et Come, une dégradation du score pour les graphes *C. elegans*, *Yeast* et *Gowalla*, non préjudiciable, car il demeure pour chaque graphe des instances qui atteignent 1. Le résultat de *Slashdot* est plus problématique, car il passe d'un peu moins de 1 pour Louvain+ à une plage entre 0.5 et 0.9 pour Come. Par principe, toute valeur inférieure à 1 pose problème, car idéalement toutes les communautés doivent être bien

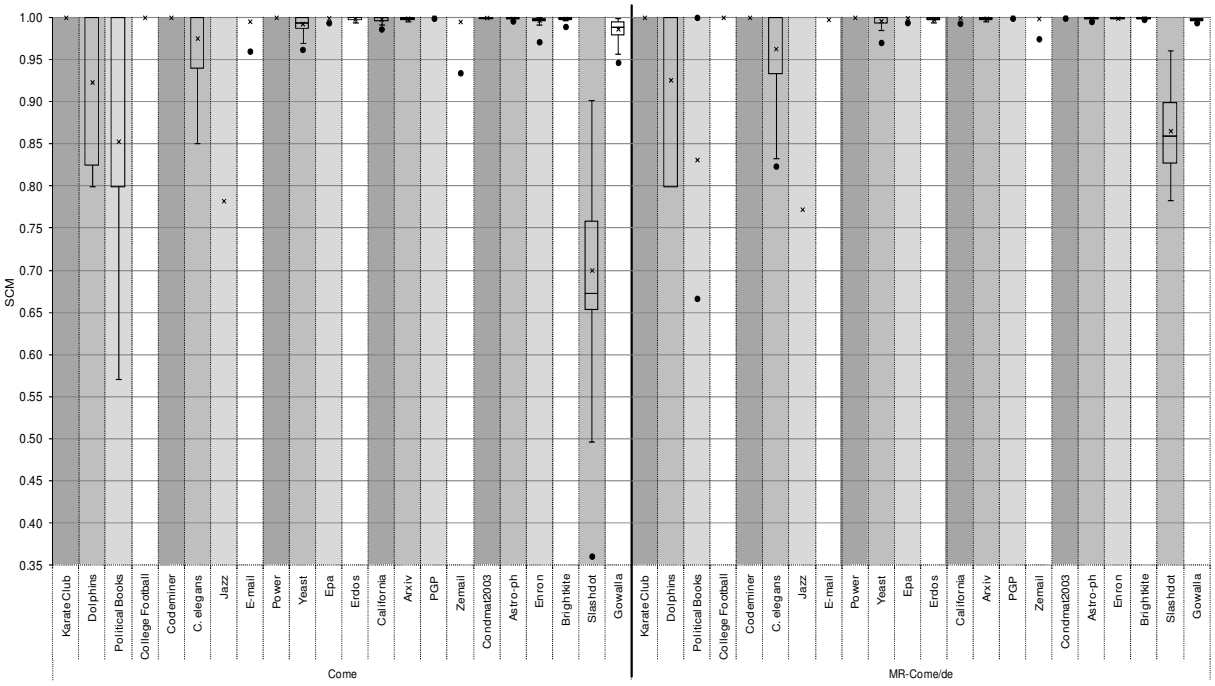


FIGURE 6.8 – Boîtes à moustaches de distribution, sur les 100 instances du protocole d'expérimentation, de la mesure SCM pour chaque graphe réels et pour les algorithmes Come et MR-Come/de. Les paramètres communs sont $\epsilon_c = \epsilon_r = 10^{-5}$ et $\lambda = 0.285$. Pour MR-Come/de, les paramètres sont $\beta_1 = 6$ et $\beta_s = 1$.

définies, sauf si le partitionnement résultant de l'optimisation de Q produit comme score maximal une valeur inférieure à 1. Cela se produit pour le graphe *Jazz* dénotant peut être un problème de structure communautaire pour ce graphe ou un échec de l'optimisation de la modularité pour partitionner ce graphe. Pour le cas du graphe *Slashdot*, Louvain+ est pratiquement à 1. La baisse de performance de l'algorithme Come doit nous inciter à relativiser la valeur du paramètre λ pour les graphes réels, une valeur plus proche de zéro étant préférable pour ce graphe.

Comme nous l'avons établi au chapitre précédent avec la condition de fusion, les densités minimale et surtout moyenne sont révélatrices d'une amélioration de partitionnement et d'une réduction de la limite de résolution. La modularité et la densité moyenne ont des croissances inversées lorsque le paramètre λ augmente (graphique 5.5). L'idée sous-tendue par ce résultat est simple : une communauté est plus forte si elle se rapproche d'une clique et donc que sa densité augmente. Cet objectif à lui seul ne peut suffire, car il conduit à des communautés singleton de densité 1, mais il prend tout son sens combiné avec la modularité qui intègre l'objectif de liens faibles entre communautés. L'association de ces deux objectifs fait l'objet du chapitre suivant. Faisons le constat pour l'instant que l'algorithme Come augmente dans l'ensemble la densité moyenne avec la figure 6.9. L'algorithme MR-Come/de a un impact neutre sur la densité moyenne (figure 6.10) avec des baisses et des hausses cantonnées dans l'intervalle $[-0.1; 0.1]$.

Les graphes réels sont sans conteste de structure plus diversifiée et plus complexe que les graphes artificiels de type LFR. Les résultats que nous présentons sur les graphes réels montrent pour la plupart des graphes un maintien du score SCM proche de 1 et une augmentation de la densité moyenne en passant de Louvain+ à Come et MR-Come, mais ils révèlent aussi pour d'autres graphes une dégradation de ces mesures. Nous formulons l'hypothèse que cette contre-performance est due en partie à des valeurs de paramètres inadéquates pour ces graphes. En

particulier, la valeur de paramètre λ (la dégradation la plus forte est due à l'algorithme Come) est plus sensible pour les graphes réels que pour les graphes artificiels. Cette hypothèse est vérifiée au chapitre suivant.

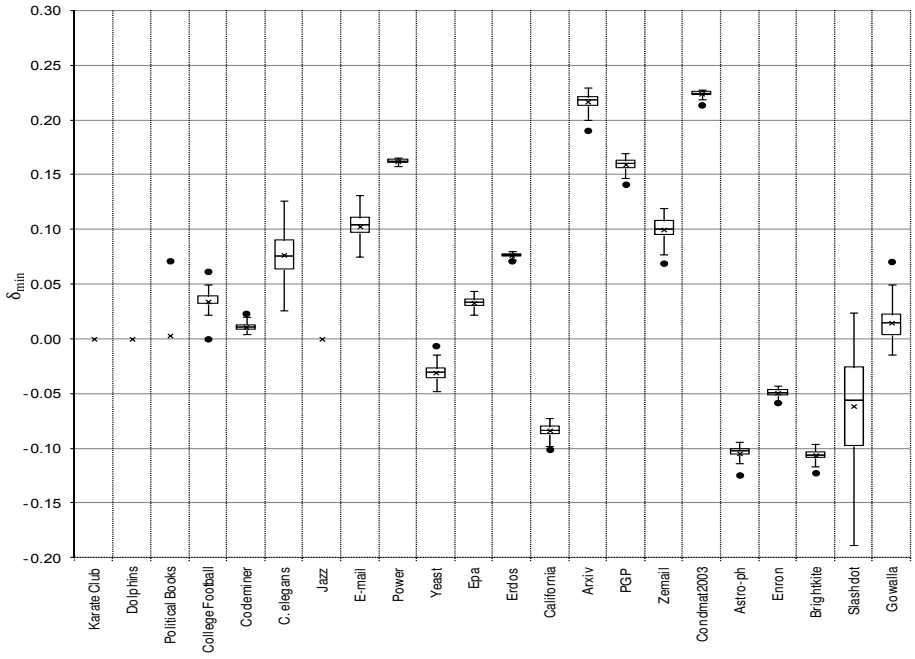


FIGURE 6.9 – Boîtes à moustaches de distribution, sur les 100 instances des graphes réels, de la différence de densité minimale, calculée d'instance à instance, entre l'algorithme Come avec $\lambda = 0.285$ et Louvain+ avec $\epsilon_c = \epsilon_r = 10^{-5}$.

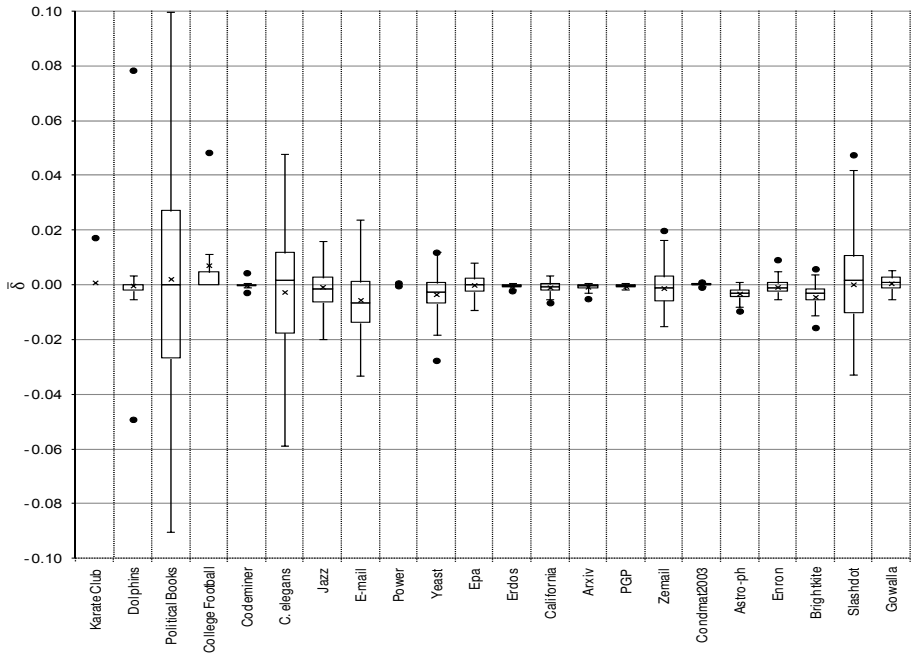


FIGURE 6.10 – Boîtes à moustaches de distribution, sur les 100 instances des graphes réels, de la différence de densité moyenne, calculée d'instance à instance, entre l'algorithme MR-Come/de $\beta_1 = 6$, $\beta_s = 1$ et $\lambda = 0.285$ et Come avec $\epsilon_c = \epsilon_r = 10^{-5}$ et $\lambda = 0.285$.

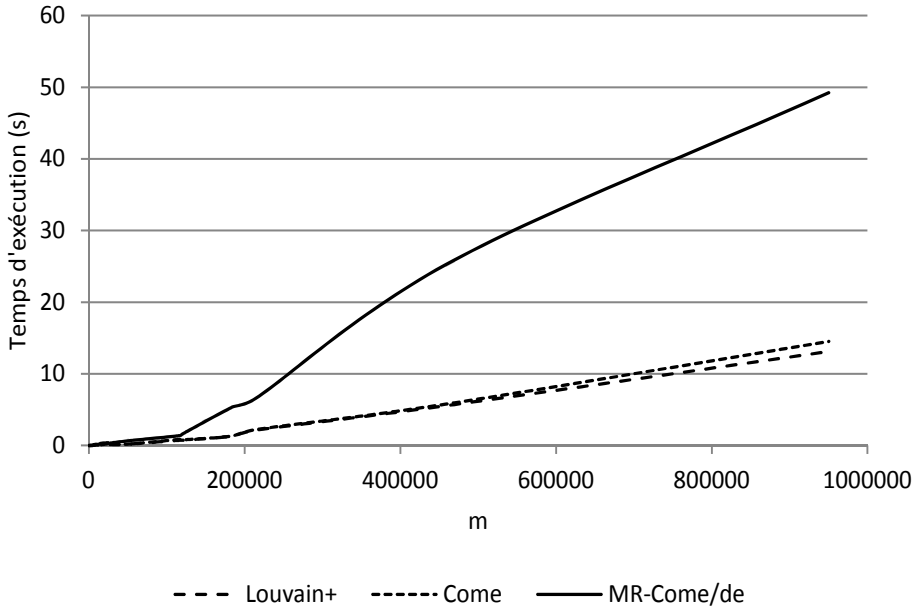


FIGURE 6.11 – Temps d'exécution moyen comparé entre Louvain+, Come et MR-Come/de sur les graphes réels. Les paramètres sont $\lambda = 0.285$, $\epsilon_c = \epsilon_r = 10^{-5}$, $\beta_1 = 6$ et $\beta_s = 1$.

6.2.4 Temps d'exécution et complexité

Les temps d'exécution des algorithmes Louvain+, Come et MR-Come/de en fonction du nombre d'arêtes m sont représentés sur la figure 6.11. Ce graphique montre une préservation de la quasi-linéarité de l'algorithme Louvain+ constatée et discutée au Chapitre 3, en toute logique, car la procédure VM est répétée dans MR-Come un nombre fixe de fois et l'appréciation de la condition de fusion dans Come est réalisée en temps constant.

6.3 Impact sur les défauts de la modularité

6.3.1 Limite de résolution

Les résultats sur la NMI présentés plus haut tendent à montrer que la limite de résolution est amoindrie par les algorithmes Come et MR-Come, mais une observation du nombre et de l'étendue des tailles de communautés offre une démonstration plus convaincante et précise.

Algorithme Louvain+ vs Come

Avec les résultats de la figure 6.12, portant sur le nombre de communautés, Louvain+ est systématiquement dans l'erreur avec un ratio inférieur à 1, c'est-à-dire un nombre de communautés trop faible. Ceci est l'une des manifestations de la limite de résolution. L'amélioration de ce score apportée par Come est manifeste pour tous les graphes à l'exception de #10 et #15. Pour les graphes #2, #3, #4, #6, #9, #13, #14, #16 et #19, toutes les instances sont pratiquement à un score de 1, la valeur idéale.

Cet impact décisif sur la limite de résolution est évidemment le mieux illustré par l'erreur sur la taille de la plus petite communauté, sur la figure 6.13. Les erreurs de Louvain+ peuvent être très importantes, jusqu'à une taille presque 64 fois trop grande. Avec l'algorithme Come, la taille de plus petite communauté ne dépasse pas le double de celle de la solution, mais devient trop faible pour 6 graphes (#8, #12, #15, #17, #18 et #20).

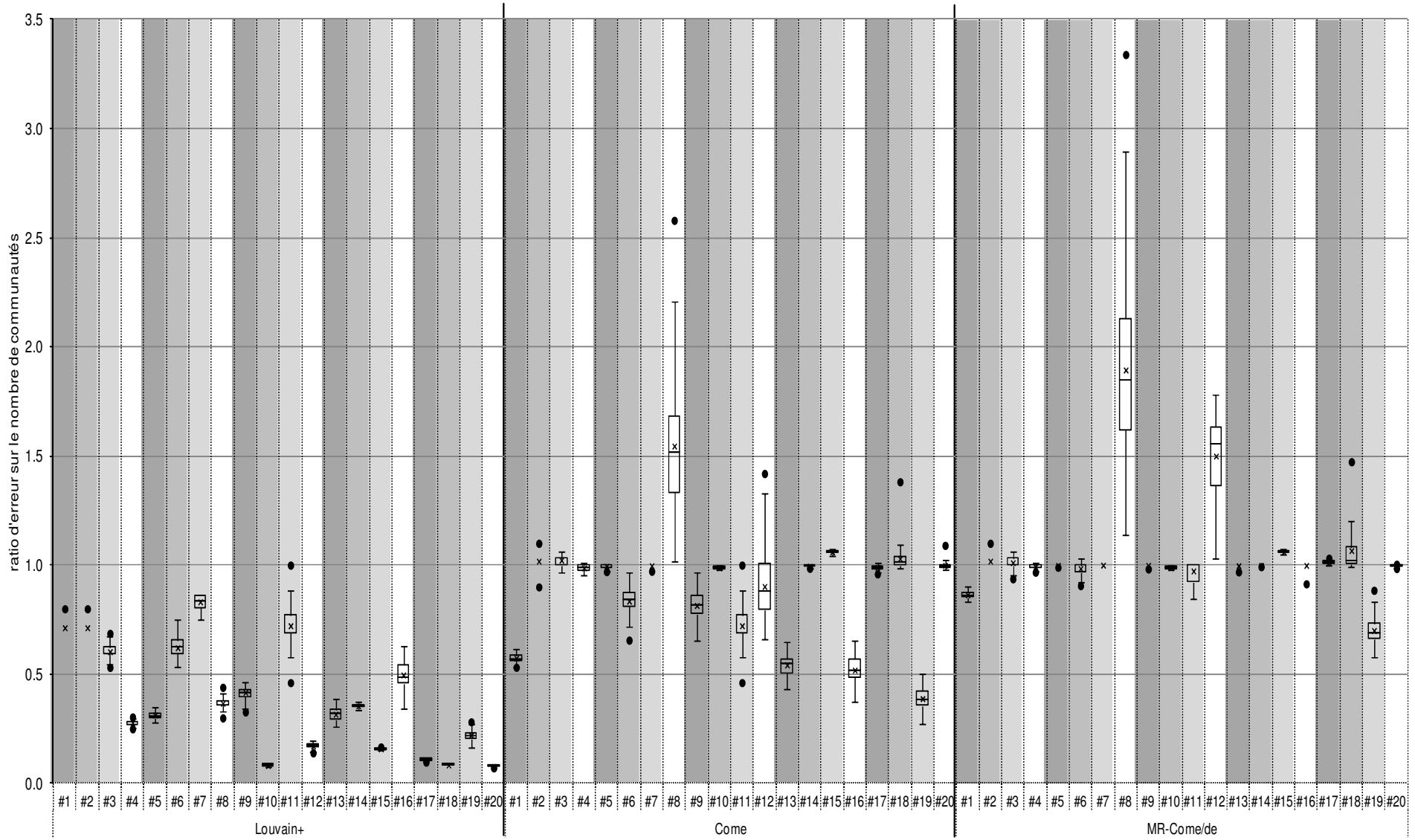


FIGURE 6.12 – Boîtes à moustaches de distribution du ratio d'erreur sur le nombre de communautés détecté par Louvain+, Come et MR-Come/de. La distribution porte sur les 100 instances des graphes artificiels. Les paramètres sont $\epsilon_c = \epsilon_r = 10^{-5}$, $\lambda = 0.285$, $\beta_1 = 6$ et $\beta_s = 1$.

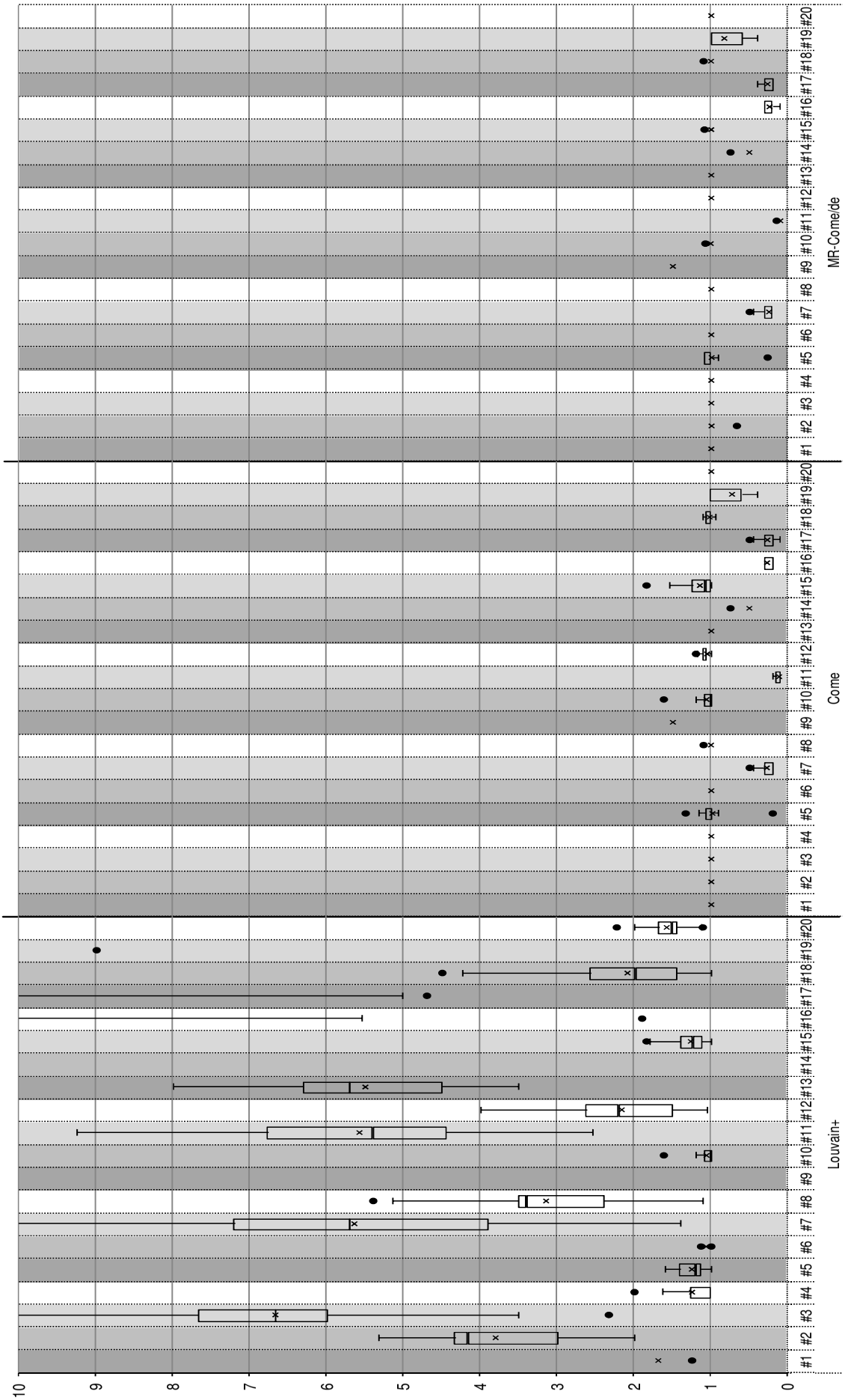


FIGURE 6.13 – Boîtes à moustaches de distribution du ratio d'erreur sur la taille de plus petite communauté détectée par Louvain+, Come et MR-Come/de. La distribution porte sur les 100 instances des graphes artificiels. Les paramètres sont $\epsilon_c = \epsilon_r = 10^{-5}$, $\lambda = 0.285$, $\beta_1 = 6$ et $\beta_s = 1$.

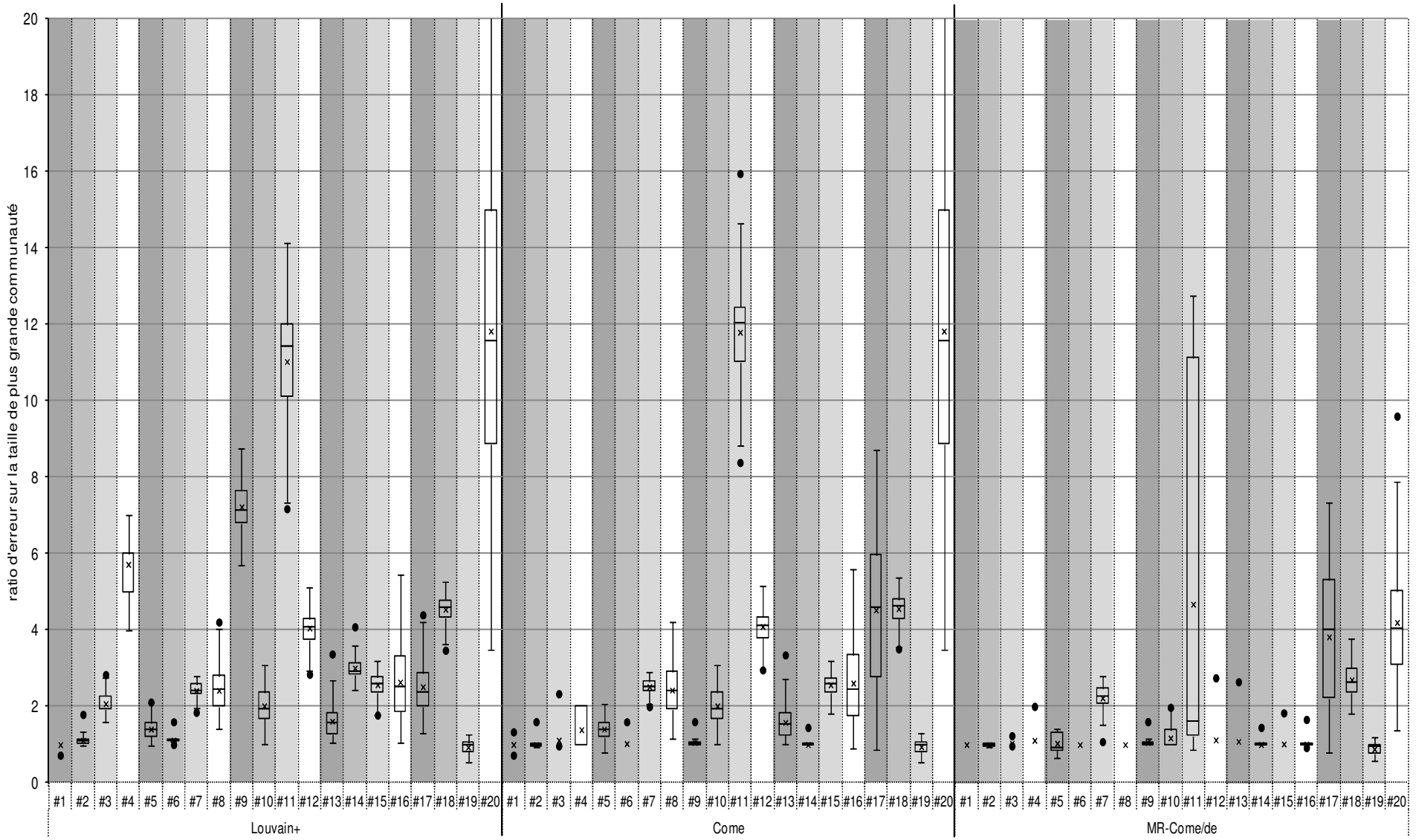


FIGURE 6.14 – Boîtes à moustaches de distribution du ratio d'erreur sur la taille de plus grande communauté détectée par Louvain+, Come et MR-Come/de. La distribution porte sur les 100 instances des graphes artificiels. Les paramètres sont $\epsilon_c = \epsilon_r = 10^{-5}$, $\lambda = 0.285$, $\beta_1 = 6$ et $\beta_s = 1$.

Enfin, concernant l'algorithme Come, nous observons que son impact sur la taille de plus grande communauté est quasi nul (figure 6.14). Ce constat rejoint les conclusions du chapitre précédent, relative à la condition de fusion, qui ont justifié de chercher à réduire la taille des communautés dès la première phase VM, par l'algorithme MR-Come.

Algorithme Come vs MR-Come/de

Nous avons précédemment constaté une amélioration très significative de la NMI par l'algorithme MR-Come par rapport à l'algorithme Come. Les figures 6.12, 6.13 et 6.14 précisent et confirment ce constat. La distribution du ratio d'erreur sur le nombre de communautés est très resserrée autour de 1 pour 16 graphes artificiels sur les 20 testés. Les graphes #7, #11, #17 et #18 demeurent en échec avec une légère aggravation de l'erreur pour les deux premiers.

L'algorithme MR-Come agit modérément sur la taille de plus petite communauté avec toutefois quelques améliorations remarquables comme pour le graphe #15 qui monte à 1.5 avec Come et est contenu à 1 avec MR-Come, hormis une valeur atypique à 1.1. L'action principale de cet algorithme est sur la taille de plus grande communauté. L'erreur importante sur les graphes #8, #11, #12, #13, #15, #16, #17, #18 et #20, constatée avec Come, est fortement réduite grâce à MR-Come. Il demeure une erreur relativement importante, dépassant le ratio de 2, pour 5 graphes.

Dans l'ensemble, nous constatons un effet très positif sur la limite de résolution des graphes artificiels. La taille de plus petite communauté est correctement trouvée pour 13 graphes. Les tests utilisent les mêmes paramètres pour les 20 graphes et leurs résultats seraient sans doute améliorés en ajustant les paramètres β_1 et λ pour les graphes posant problème.

Concernant les graphes réels, la réduction de la limite de résolution peut être appréciée en observant l'augmentation globale de la densité de communauté moyenne (voir les résultats sur les graphes réels). La technique consistant à partitionner le sous-graphe d'une communauté pour vérifier si des sous-communautés existent et donc si la résolution est mauvaise, ne nous paraît pas pertinente, car le sous-graphe fait abstraction des liens externes avec les autres communautés.

6.3.2 Inconsistance de solution

D'après la figure 6.15, les solutions générées par Come sont nettement plus proches entre elles que celles générées par Louvain+, pour tous les graphes réels testés. En moyenne, la mesure de similarité NID passe de 0.767 avec Louvain+ à 0.868 avec Come. Avec MR-Come, la similarité augmente pour 19 graphes sur 22 et atteint en moyenne 0.892. La condition de fusion et les raffinements à échelle décroissante réduisent significativement la diversité des solutions possibles pour un même graphe, donnant ainsi plus de sens à ces solutions.

Conclusion

Le raffinement par échelle décroissante préalable à des fusions sous condition est efficace sur 17 graphes artificiels testés qui donnent un score moyen de NMI calculée sur 100 instances compris entre 0.9 et 1. Cet algorithme, avec des valeurs de paramètres appropriées, réduit les effets néfastes de la modularité, en particulier la limite de résolution, de sorte que pour 16 graphes artificiels le nombre de communautés trouvé est égal au nombre attendu à +/- 20 %, mais aussi l'inconsistance de solution avec pour résultat, sur les graphes réels, une similarité passant de moins de 0.8 avec Louvain+ à presque 0.9 avec l'algorithme présenté. Parmi les graphes réels, nous observons un accroissement de la densité moyenne de communauté pour 16 graphes sur 22 qui dénote aussi une diminution des effets néfastes de la limite de résolution. Néanmoins,

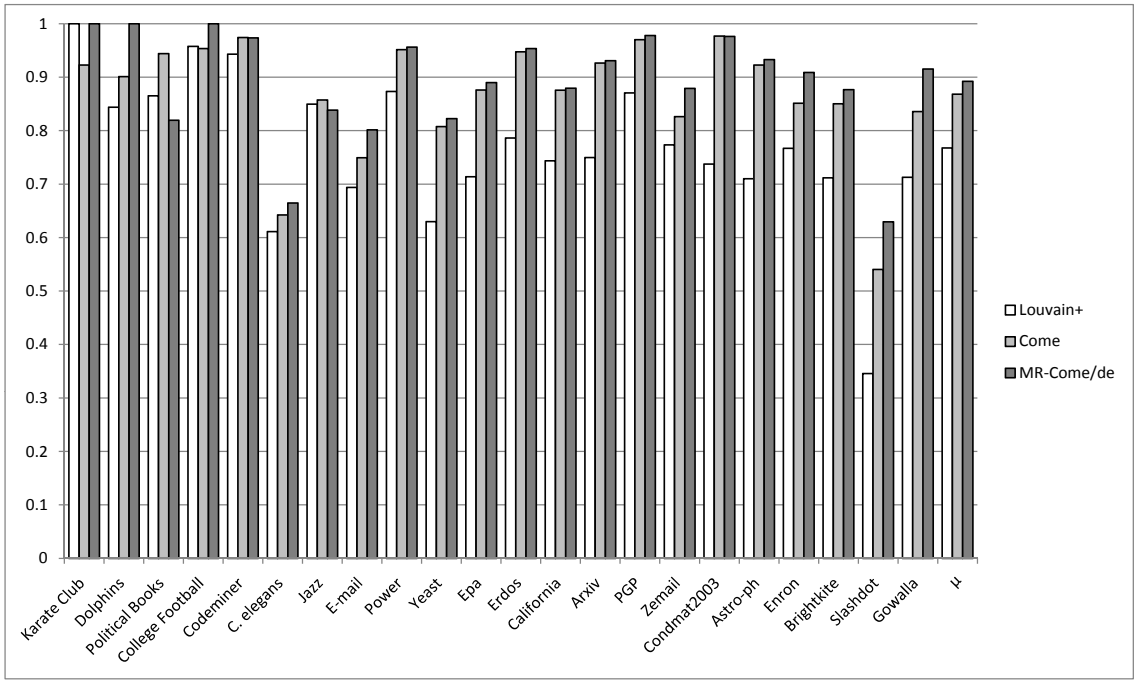


FIGURE 6.15 – Inconsistance de solution de MR-Come, mesurée par la similarité moyenne entre solutions sur les graphes réels. La version stochastique de l'algorithme est exécutée 20 fois sur chaque graphe. La similarité moyenne est calculée comme la moyenne de la NID de toutes les paires distinctes de solutions. Une similarité proche de 1 indique des solutions structurellement proche et donc une inconsistance liée à la modularité réduite.

les partitions de 3 graphes artificiels demeurent éloignées de la solution, bien que cette différence se réduise, avec une similarité inférieure à 0.7 pour deux d'entre eux. De plus le score d'existence minimale de communauté (SCM) s'aggrave ou reste inchangé pour 6 graphes réels par rapport aux résultats de l'algorithme Louvain+. Ces mauvais résultats peuvent s'expliquer soit par une difficulté de certains graphes rédhibitoire pour cet algorithme, soit par un mauvais ajustement des paramètres pour ces graphes en particulier. Nous avons conjecturé que la densité moyenne ou minimale pourrait fournir un moyen de sélectionner certaines partitions obtenues par exemple avec différentes valeurs de paramètres. L'idée générale est de combiner deux fonctions d'objectif, la modularité employée sous condition de fusion et la densité, pour produire et sélectionner des partitions en s'affranchissant du délicat problème du choix des valeurs de paramètre. La formulation, les modalités et l'expérimentation de cette idée font l'objet du dernier chapitre.

7

Algorithme mémétique biobjectif

Sommaire

Le dernier algorithme original que nous présentons fait la synthèse entre l'algorithme mémétique, qui explore efficacement l'espace de recherche, et la condition de fusion, qui limite les effets négatifs de la limite de résolution. En préambule, nous vérifions que la densité minimale fournit un critère pertinent pour discriminer un grand nombre de partitions obtenues par la condition de fusion avec diverses valeurs de paramètres. Ce critère est introduit dans l'algorithme mémétique, comme fonction d'adaptation, c'est-à-dire d'objectif à optimiser pour sélectionner les individus. L'algorithme y gagne en généralité et se comporte mieux avec les graphes réels. Finalement, les résultats sur les graphes artificiels sont excellents avec une similarité moyenne de partition par rapport à la solution de 0.978, très proche du maximum 1. Avec ces tests, la limite de résolution est quasi inexistante, les petites communautés n'étant pas absorbées par les plus grandes. Enfin, l'inconsistance de solution atteint un niveau très faible. Cependant, l'algorithme a un temps d'exécution bien supérieur à Louvain+.

Introduction

Nous avons montré l'efficacité de la condition de fusion pour atténuer la limite de résolution et produire des partitions plus justes. Cette condition est appliquée en deuxième phase de contraction de l'algorithme de Louvain et améliore grandement les performances de celui-ci. Malheureusement, la première phase VM, même améliorée par les raffinements successifs du chapitre précédent, est susceptible de produire des communautés trop grandes d'autant plus que l'appartenance des nœuds aux communautés est faible. La deuxième phase qui procède à des fusions est incapable de corriger ce problème et laisse, au mieux, ces communautés intactes.

Par ailleurs, notre algorithme mémétique est très efficace non seulement pour optimiser la modularité, mais aussi pour explorer l'espace de recherche en trouvant précisément des partitions, très peu nombreuses, dont la modularité dépasse les innombrables optimums locaux atteints par l'algorithme de Louvain. Notre idée est d'associer l'algorithme mémétique à la condition de fusion afin d'exploiter l'opérateur de croisement pour "casser" les communautés trop grandes. En effet, le croisement a tendance à augmenter le nombre de communautés en mêlant les communautés des deux parents. En utilisant l'algorithme Louvain+ avec condition de fusion comme procédure d'optimisation locale à appliquer après le croisement, nous avons l'espoir que les communautés trop grandes, scindées par le croisement, ne se reformeront pas si la condition de fusion l'interdit.

Dans cette version hybride entre les algorithmes MA-COM (mémétique) et Come (Louvain+ avec condition de fusion), la population initiale est générée avec Come et l'optimisation après croisement est également réalisée avec Come. Ainsi, tous les individus de la population sont soumis à la condition de fusion. Les tests préliminaires ont montré des résultats très intéressants sur les graphes artificiels, mais aucune amélioration de la population initiale sur les graphes réels. Ce phénomène paraît anormal, car la complexité des graphes réels, a priori supérieure à celle des graphes artificiels, devrait plutôt conduire au comportement inverse. Nous incriminons la modularité en tant que fonction objectif de l'algorithme mémétique pour sélectionner les meilleurs individus. Cette sélection s'oppose au principe de la condition de fusion qui veut qu'une modularité forte, mais pas optimale soit privilégiée, car sinon l'optimisation trop forte conduit aux erreurs de la limite de résolution. Il est nécessaire d'abandonner la modularité comme objectif de la sélection pour choisir un autre critère.

Ces interrogations relèvent de la question plus générale d'une discrimination entre partitions obtenues par l'algorithme Come ou MR-Come. Les différentes partitions obtenues par les versions stochastiques d'algorithme, pour le même graphe, sont certes plus proches entre elles en utilisant Come ou MR-Come plutôt qu'en utilisant Louvain+, mais demeurent tout de même distantes, avec une similarité moyenne comprise entre 0.85 et 0.9 (pour $\lambda = 0.3$, voir la figure 5.7). Nous pouvons proposer aux utilisateurs de ces algorithmes d'appliquer leur propre critère structurel et dépendant de leur discipline, pour choisir une partition en particulier. Nous devons aussi nous interroger sur l'existence d'un critère général qui, parmi toutes ces partitions proches, oriente vers un nombre limité de partitions jugées plus pertinentes.

7.1 Critère de sélection de partitions

Dans le chapitre précédent consacré à l'algorithme MR-Come, qui utilise des raffinements successifs en première phase (paramètres β_1 et β_s) et la condition de fusion en seconde phase (paramètre λ), le problème d'un choix unique de valeurs de paramètre, quel que soit le graphe, s'est posé. Nous avons établi que la valeur $\lambda = 0.285$ était théoriquement idéale pour contenir les effets de la limite de résolution, ce que les résultats d'expérimentation confirment. Pour les paramètres propres à MR-Come, $\beta_1 = 6$ et $\beta_s = 1$ semblent convenir à la grande majorité des graphes artificiels testés, à l'exception de quelques graphes pour lesquels d'autres valeurs de β_1

donnent de meilleurs résultats. Nous avons enfin établi que β_1 n'est pas un paramètre crucial et que la valeur 1 convient parfaitement. Ainsi, nous souhaitons rechercher un discriminant de partition parmi un vaste ensemble de partitions obtenues pour différentes valeurs de paramètres.

7.1.1 Principe de l'expérimentation

Notre expérience consiste à exécuter l'algorithme MR-Come/de, une fois pour chaque graphe artificiel, avec chaque valeur de β_1 prise entre 0.05 et 0.5 par pas de 0.05 (soit 10 valeurs) croisée avec chaque valeur de λ prise entre 0 et 1 par pas de 1 (soit 10 valeurs). Le paramètre β_s est fixé à 1. Nous obtenons 100 partitions par graphe et devons établir un critère qui discrimine les meilleurs d'entre elles.

Le premier point est de vérifier une définition correcte des communautés. Nous savons que les grandeurs SN et SC, qui correspondent aux critères fort et faible d'existence des communautés, sont fortement liées à la modularité et que par conséquent les meilleurs scores ne sont pas synonymes de partitions idéales. En revanche, le critère d'existence le plus faible, traduit par la valeur SCM, nous est apparu comme essentiel. En effet, d'une part les graphes artificiels montrent un score de 1, la valeur idéale, avec la partition solution (excepté pour le graphe #19 qui obtient 0.9967, voir le tableau 2.3) et d'autre part, ce score demeure très proche de 1 avec l'algorithme Come exécuté avec un paramètre λ compris entre 0 et 0.4. Il est difficile de concevoir une communauté bien formée qui aurait un degré externe avec une autre communauté supérieur à son degré interne. Pour cette raison, nous appliquons ce critère comme premier filtre discriminant, en éliminant les partitions qui ne vérifient pas SCM $= 1$, ou à tout le moins qui n'ont pas le score maximum[1].

Vérifiant le critère d'existence des communautés, il reste bon nombre de partitions de structures différentes, qu'il faut départager par un critère plus discriminant. Nous souhaitons tester plusieurs critères, comme la modularité, la densité moyenne ou minimale, et vérifier si, dans l'ensemble des partitions obtenues pour un graphe donné G, la partition qui a la valeur optimale de ce critère est aussi la plus proche de la solution et a donc une NMI maximale. Pour être plus précis, notons C_G l'ensemble des partitions candidates qui vérifient le premier critère de valeur SCM maximale, pour le graphe G. Notons A_G le sous-ensemble de C_G des partitions dont la valeur du critère discriminant à tester est maximale. Toutes ces partitions ayant la même valeur de critère discriminant constituent l'ensemble des partitions sélectionnées. Pour vérifier que celles-ci sont bien les meilleures, nous calculons la NMI de chacune par rapport à la solution, puis la moyenne de toutes ces NMI notée $\overline{NMI}_G$. Cette dernière valeur représente le "score" absolu des partitions sélectionnées par le critère. Finalement, pour permettre une comparaison entre graphes et pour relativiser ce score, nous calculons la NMI normalisée, entre 0 et 1, selon l'étendue des valeurs de NMI des partitions de A_G, soit :

$$\overline{NMI}^n_G = \frac{\overline{NMI}_G - \min_{\mathcal{C}\in A_G} NMI(\mathcal{C})}{\max_{\mathcal{C}\in A_G} NMI(\mathcal{C}) - \min_{\mathcal{C}\in A_G} NMI(\mathcal{C})} \tag{7.1}$$

Résultats pour le choix d'un critère discriminant

Les résultats de cette expérience sont résumés sur la figure 7.1 montrant la répartition de $\overline{NMI}^n_G$ pour cinq critères de sélection pris parmi les grandeurs définies dans le protocole d'expérimentation, soit la modularité Q, le score de nœuds bien définis (SN), le score de communautés bien définies (SC), le score de communautés au sens le plus faible (SCM) et les densités moyenne

[1] Heureusement, dans notre expérience, sur les 100 instances obtenues, tous les graphes ont au moins une partition avec SCM=1, y compris #19.

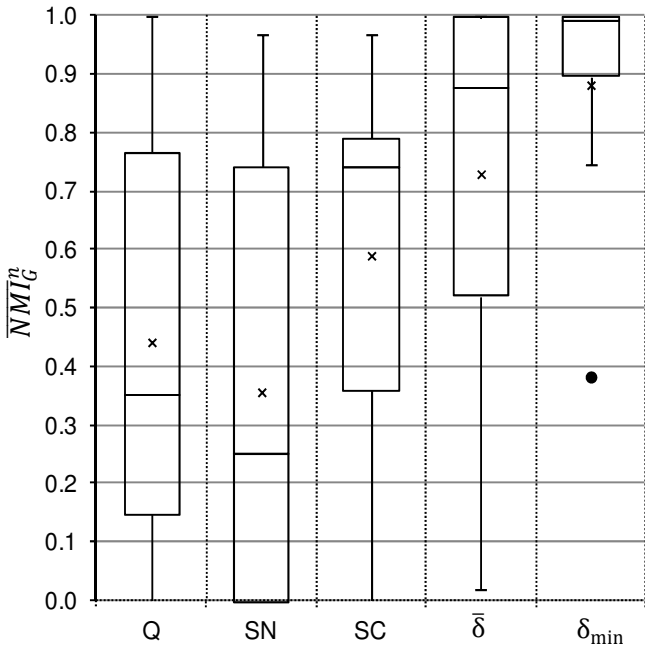


FIGURE 7.1 – Boîtes à moustaches de distribution du score de "meilleures" communautés sélectionnées selon différents critères figurant en abscisse.

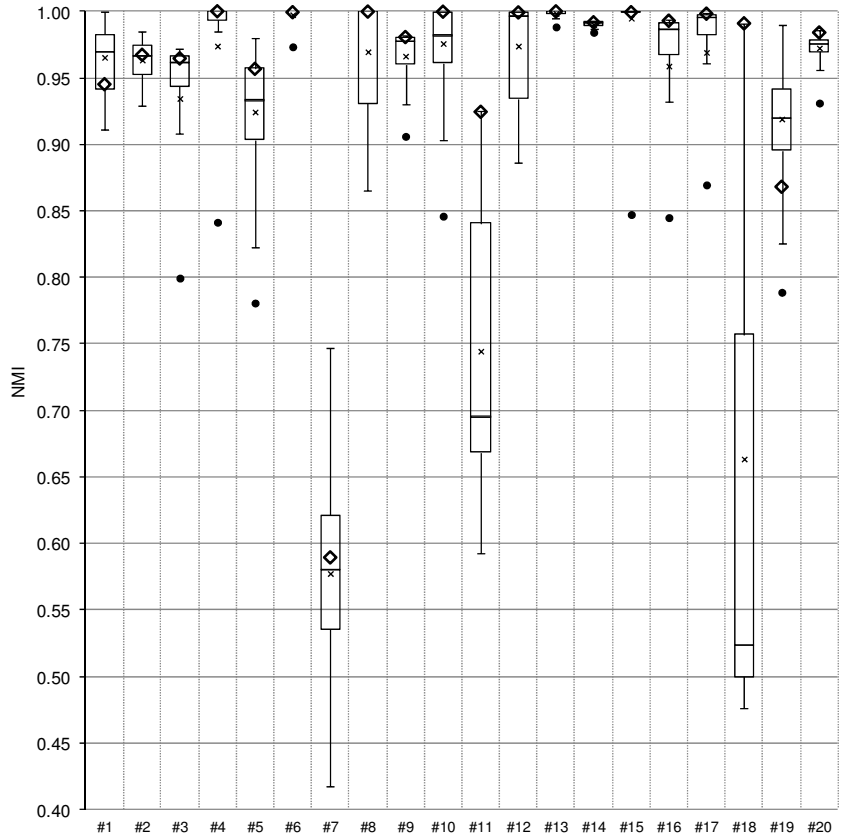


FIGURE 7.2 – Boîtes à moustaches de distribution de la NMI des 100 instances (10 valeurs de β_1 croisées avec 10 valeurs de λ) pour chaque graphe artificiel. La moyenne de NMI notée $\overline{NMI}_G$ et calculée sur les partitions sélectionnées par le critère de la densité minimale est représentée par un losange.

et minimale $\bar{\delta}$ et δ_{min} (moyenne des densités de communautés et plus petite densité de communauté). De manière très claire, nous constatons que la densité minimale produit le meilleur résultat avec une NMI moyenne normalisée entre 0.75 et 1 si l'on excepte une valeur atypique proche de 0.4 pour le graphe #19. Les résultats de Q et SN sont très mauvais couvrant toute la plage possible de NMI, confirmant ainsi nos précédentes conclusions sur la corrélation entre ces deux grandeurs. La densité moyenne offre une performance correcte, mais inférieure à la densité minimale, sans doute parce qu'elle résulte d'une somme de valeurs qui traduit un compromis et favorise la limite de résolution. Au contraire, la densité minimale renseigne sur la plus mauvaise communauté qui est souvent la plus grande. Nous avons remarqué qu'en poussant l'optimisation de la modularité jusqu'aux valeurs qui aggravent la limite de résolution, les trop grandes communautés apparaissent et la densité de celles-ci décroît. Donc, le phénomène de disparition des petites communautés est corrélé à la baisse de densité minimale.

Résultats détaillés de la densité minimale

Pour appuyer et affiner ces résultats, observons la répartition de NMI de notre expérimentation (100 instances obtenues par croisement de 10 valeurs de β_1 et de 10 valeurs de λ), pour chaque graphe, sur la figure 7.2. Ce graphique montre également le niveau de NMI moyen des partitions sélectionnées par le critère de densité minimale (valeur $\overline{NMI}_G$). En comparant ce résultat avec celui de la figure 6.6, nous pouvons premièrement constater, en observant simplement la distribution de NMI, que cette expérience avec une large palette de valeurs de paramètre ne conduit pas à un meilleur résultat que le simple MR-Come/de avec $\beta_1 = 6$ et $\lambda = 0.285$. Les trois graphes qui posent problème, soit #7, #11 et #18 ont d'aussi mauvais résultats. De plus, l'étendue de NMI est plus large avec des valeurs minimales plus basses. Le deuxième constat concerne notre critère de sélection qui s'avère très bon dans ce contexte avec les situations suivantes de $\overline{NMI}_G$ par rapport à la distribution de NMI : 7 valeurs sont dans le dernier quartile et 7 autres sont dans le troisième quartile (donc au-dessus de la médiane). Dans l'absolue, indépendamment de la distribution de NMI, 12 valeurs de $\overline{NMI}_G$ sur 20 sont entre 0.98 et 1 et 18 entre 0.94 et 1. Parmi les graphes problématiques, #18 est bien traité avec une $\overline{NMI}_G$ à 0.998, mais #19 est légèrement moins bien traité avec 0.868.

Ces résultats nous confortent dans l'idée que la densité minimale peut être combinée à la modularité pour éviter les problèmes de limite de résolution et départager un grand nombre de valeurs de paramètres possibles avec des méthodes comme Come ou MR-Come. Cette expérience pourrait fournir un algorithme opérationnel, à condition d'exécuter un nombre significatif d'instances avec des valeurs de paramètres variées. Cependant, notre propos est plutôt de justifier l'utilisation de ce critère de sélection de partitions dans un algorithme mémétique.

7.2 Description de l'algorithme

Le dernier algorithme que nous souhaitons présenter fait la synthèse entre l'algorithme mémétique du chapitre 4 et la condition de fusion du chapitre 5, les deux s'appuyant sur l'amélioration de l'algorithme de Louvain décrite au chapitre 4. L'algorithme, que nous appellerons MemCM reprend intégralement les mécanismes de l'algorithme mémétique avec croisement par intersection de communautés (MA-COM/IC) en substituant la densité minimale à la modularité comme fonction objectif pour la sélection des individus et l'algorithme Come à l'algorithme Louvain+, utilisé pour initier la population et optimiser les individus issus des croisements.

Le mécanisme général devient celui-ci :

1. Génération d'un nombre déterminé de partitions en exécutant la version stochastique de Come (algorithme Louvain+ avec condition de fusion de paramètre λ) et placement de ces partitions dans la population P ;

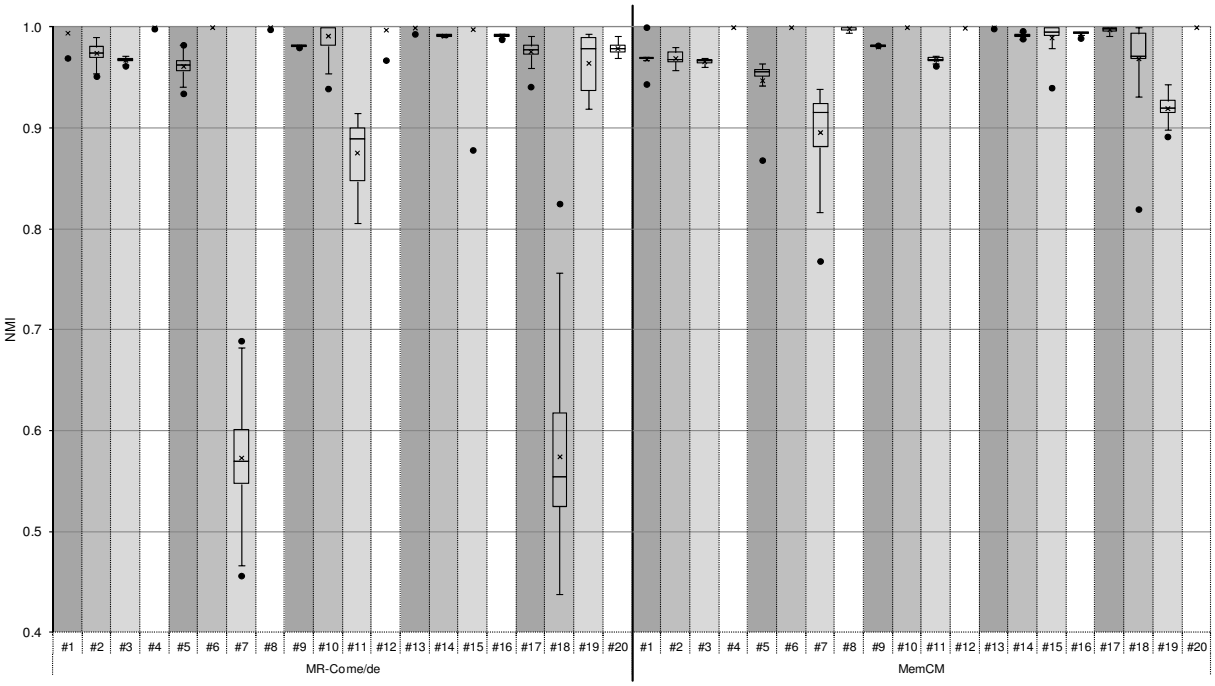


FIGURE 7.3 – Boîtes à moustaches de la NMI comparée entre MR-Come/de et MemCM, pour les graphes artificiels. Les résultats de MR-Come/de portent sur la version déterministe exécutée pour les 100 instances de graphes du protocole d'expérimentation avec les paramètres $\beta_1 = 6$, $\beta_s = 1$. Pour MemCM, la version stochastique est exécutée 20 fois. Les paramètres communs sont $\epsilon_c = \epsilon_r = 10^{-5}$ et $\lambda = 0.285$.

2. Croisement de deux partitions de P choisies au hasard, par la méthode d'intersection de communautés, pour obtenir une nouvelle partition I ;

3. Application de Come avec le même paramètre λ à la partition I pour optimiser sa modularité ;

4. Mise à jour de la population avec la partition I : I prend la place de l'individu de plus faible densité minimale si sa distance avec I est supérieure ou égale à d_{min} ou de l'individu le plus proche de I dans le cas contraire, à condition dans tous les cas que I ait une densité minimale supérieure à celle de l'individu remplacé ;

5. Répétition des étapes 2 à 4 jusqu'à ce que cette boucle soit exécutée t_{max} fois sans amélioration de la population[2].

Ainsi, les paramètres de l'algorithme sont λ, le seuil de condition de fusion, $|P|$, la taille de population , t_{max}, le nombre maximum d'itération sans amélioration de la population et enfin d_{min}, la distance minimale entre deux partitions dans la population.

7.3 Validation expérimentale

L'algorithme mémétique avec condition de fusion, nommé MemCM, est comparé au meilleur algorithme obtenu jusqu'ici, MR-Come/de. Du fait de son caractère évolutionnaire, comme

[2]Une amélioration de la population se produit si la densité minimale d'une progéniture I est supérieure à la plus forte densité minimale dans la population avant son insertion.

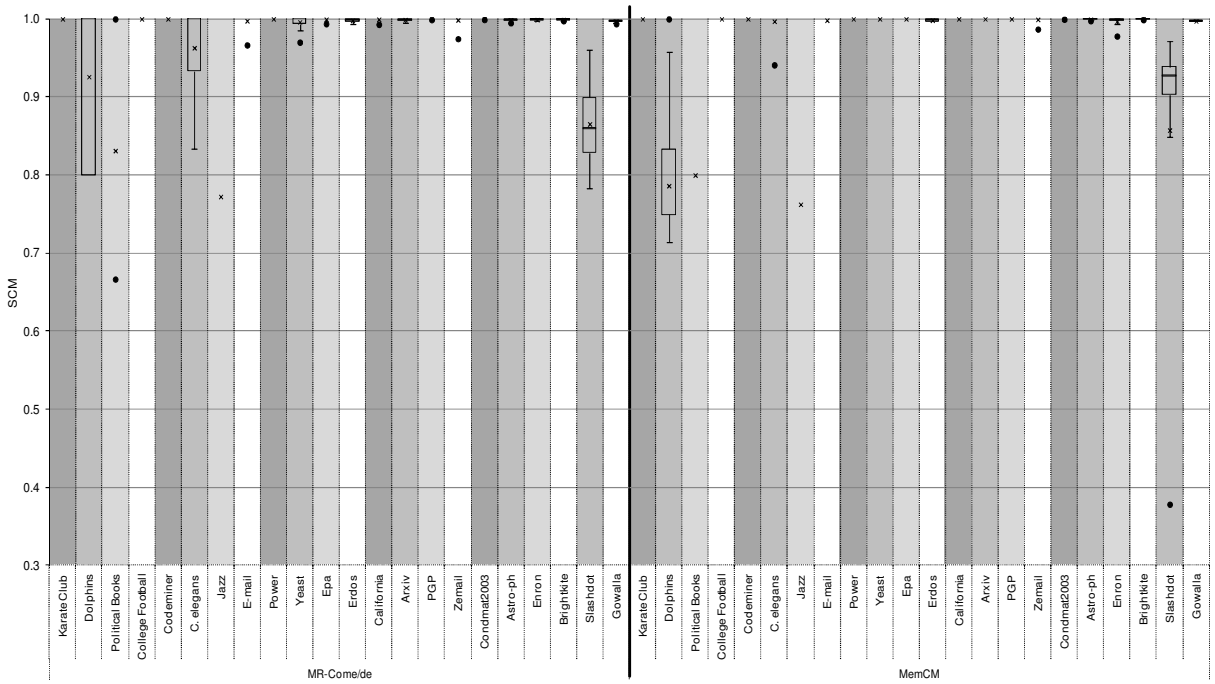


FIGURE 7.4 – Boîtes à moustaches du score SCM comparé entre MR-Come/de et MemCM, pour les graphes réels. Les résultats de MR-Come/de portent sur la version déterministe exécutée pour les 100 instances de graphes du protocole d'expérimentation avec les paramètres $\beta_1 = 6$, $\beta_s = 1$. Pour MemCM, la version stochastique est exécutée 20 fois. Les paramètres communs sont $\epsilon_c = \epsilon_r = 10^{-5}$ et $\lambda = 0.285$.

pour l'algorithme mémétique du chapitre 5, nous devons tester cet algorithme dans une version nécessairement stochastique avec des instances de graphes ayant un ordre aléatoire de nœuds. Tous les tests que nous présentons utilisent une population de taille $|P| = 10$, qui nous paraît suffisante d'après les résultats du chapitre 5. De plus, t_{max} est fixé à 100, d_{min} à 10^{-3} et λ à 0.285. Les résultats de MR-Come sont ceux présentés au chapitre précédent, avec la version déterministe exécutée sur les 100 instances du protocole d'expérimentation, et avec $\beta_1 = 6$, $\beta_s = 1$ et $\lambda = 0.285$.

7.3.1 Graphes artificiels

Sur la figure 7.3, l'apport de MemCM apparaît clairement avec une NMI au-dessus de 0.75 pour tous les graphes. Parmi les trois graphes qui posaient encore problème à MR-Come/de, seul #7 montre toujours des difficultés avec MemCM, qui améliore toutefois légèrement l'étendue de NMI pour ce graphe (médiane qui passe au-dessus de 0.9). Nous entrons plus dans les détails structurels dans la partie consacrée à la limite de résolution.

7.3.2 Graphes réels

La situation des graphes réels du point de vue du score SCM est pratiquement inchangée comme l'attestent les résultats présentés sur la figure 7.4, avec toutefois une amélioration nette de *C. elegans* et une légère dégradation de *Dolphins*. La distribution du graphe *Slashdot* remonte avec une médiane passant au-dessus de 0.9, mais une valeur atypique apparaît en dessous de 0.4. Dans l'ensemble, les conclusions relatives à MR-Come sont valables pour MemCM, certains

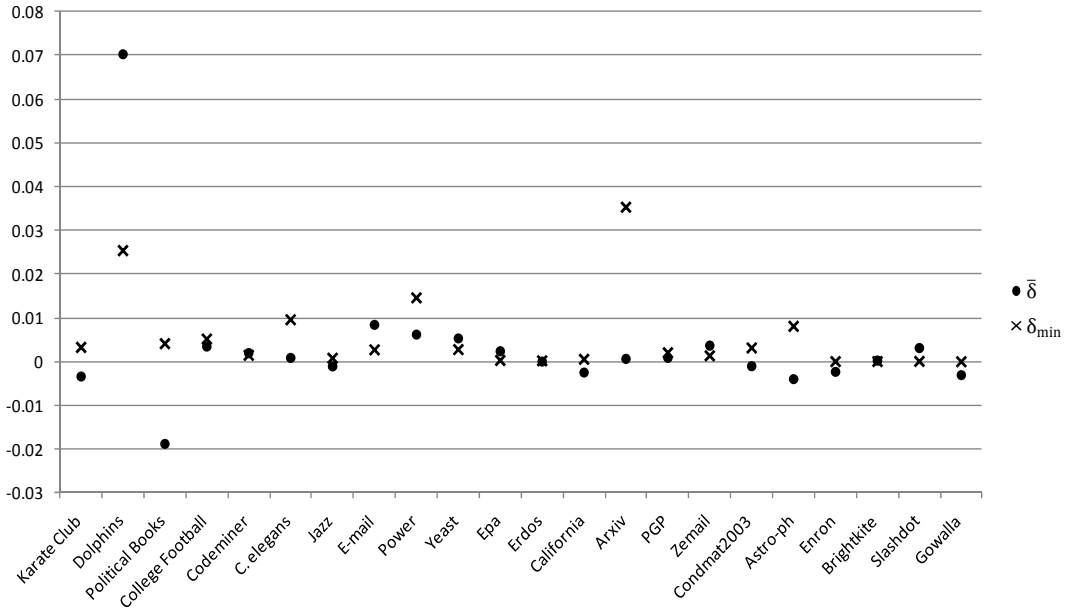


FIGURE 7.5 – Différence, entre MemCM et MR-Come/de, des moyennes de densité moyenne et de densité minimale, pour les graphes réels. La moyenne est calculée sur les 100 instances de MR-Come et les 20 instances de MemCM dans les mêmes conditions que pour les graphiques précédents. Chaque valeur reportée est la densité de MemCM moins la densité de MR-Come (moyenne ou minimale selon le symbole).

graphes réels ayant une structure communautaire difficile à mettre en évidence.

Contrairement à MR-Come comparé à Come, tous deux fondés sur l'algorithme de Louvain, nous ne pouvons pas calculer une différence de densité instance à instance, car l'expérimentation de MR-Come diffère de celle de MemCM. De plus, la distribution de densité sur les instances d'exécution est très étroite, aussi bien pour MR-Come que pour MemCM, ce qui ne justifie pas le recours à une boîte à moustache. Pour ces raisons, nous présentons simplement la différence de grandeur moyenne entre MemCM et MR-Come, en mesurant comme grandeur la densité moyenne et la densité minimale de communauté (figure 7.5). Comme attendu, la différence de densité minimale est toujours positive, ce qui signifie une densité meilleure pour MemCM que pour MR-Come, car elle constitue l'objectif d'optimisation de l'algorithme mémétique MemCM. L'amélioration n'est pas négligeable pouvant atteindre 0.035. La différence de densité moyenne, globalement légèrement à l'avantage de MemCM, est fluctuante, tantôt positive, tantôt négative.

7.3.3 Temps d'exécution et complexité

Comme cela est prévisible avec un algorithme évolutionnaire, le temps d'exécution est beaucoup plus important que celui nécessaire à générer un seul individu contenu dans la population. L'arrêt de l'algorithme MemCM est imprévisible et dépend de l'efficacité du croisement combiné à l'optimisation locale, mais aussi bien sûr de la condition d'arrêt. Il est délicat de conjecturer une complexité pour cet algorithme même si la courbe suggère une croissance linéaire, comme Louvain+, Come et MR-Come, mais avec une pente très élevée.

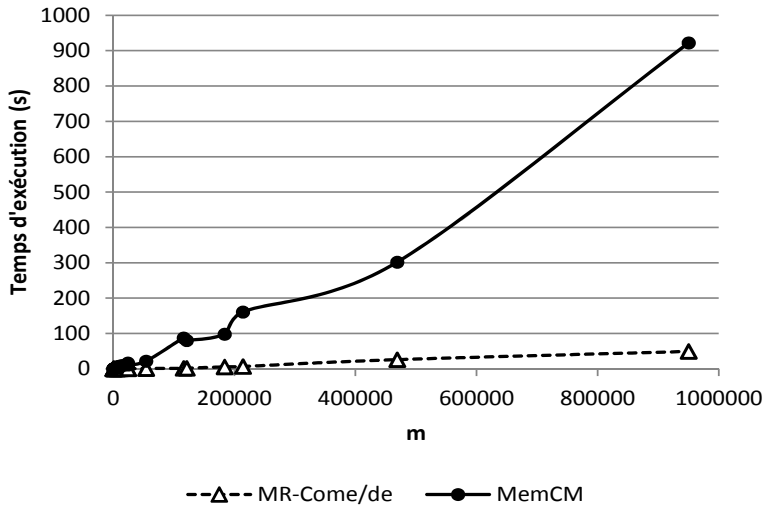


FIGURE 7.6 – Temps d'exécution moyens comparés entre MR-Come/de et MemCM dans les mêmes conditions que pour les graphiques précédents.

7.4 Impact sur les défauts de la modularité

7.4.1 Limite de résolution

Pour confirmer et affiner les résultats de NMI exposés plus haut, les figures 7.7, 7.8 et 7.9 comparent les ratios d'erreur structurels entre MR-Come/de et MemCM. Le comportement de l'algorithme est remarquable avec une amélioration significative des graphes #7, #10, #11, #17 et #18, sur le nombre de communautés, qui posaient problème avec MR-Come. Pour ce ratio, 12 graphes sont pratiquement à 1 sur toute la distribution, 7 autres ont un ratio entre 0.8 et 1.2, en incluant les quelques valeurs atypiques, et seul #7 présente une erreur significative, qui reste tout de même inférieure à 1.6. Ce graphe est très peu dense avec une large étendue de taille de communauté et une définition des communautés faible ($\mu = 0.5$ dans le modèle LFR), le rendant particulièrement difficile à partitionner. Il y a tout de même une instance très proche du ratio 1, c'est-à-dire une partition très proche de la solution.

Pour la taille de plus petite communauté, l'amélioration apportée par MemCM est plus discrète, avec trois graphes, #7, #14 et #17 qui présentent une erreur importante, entre 0.2 et 0.8. Ces graphes, peu denses, ont des petites communautés de taille inférieure ou égale à 10. L'erreur mesurée par un rapport en est d'autant plus amplifiée. Les partitions proposées par MemCM ont toutes une taille de plus petite communauté inférieure ou égale à celle de la solution. C'est un point essentiel tendant à établir l'affaiblissement des effets néfastes de la limite de résolution. Les petites communautés ne sont pas englouties par les plus grandes, mais certaines sont détectées indûment. Cette erreur est beaucoup moins préjudiciable que celle résultant de la limite de résolution, car il suffit d'observer ces très petites communautés pour décider d'une fusion ou d'un versement des nœuds dans d'autres communautés voisines.

Enfin, l'amélioration la plus remarquable porte sur la taille des grandes communautés. Le ratio, excepté les valeurs atypiques, est très proche de 1, la valeur idéale, pour 17 graphes sur 20. Seul #18 présente une erreur préjudiciable (supérieure à 1) avec des tailles jusqu'au double de la solution (hors valeur atypique). Ce graphe est très mal défini ($\mu = 0.85$) et présente indubitablement plusieurs structures communautaires différentes.

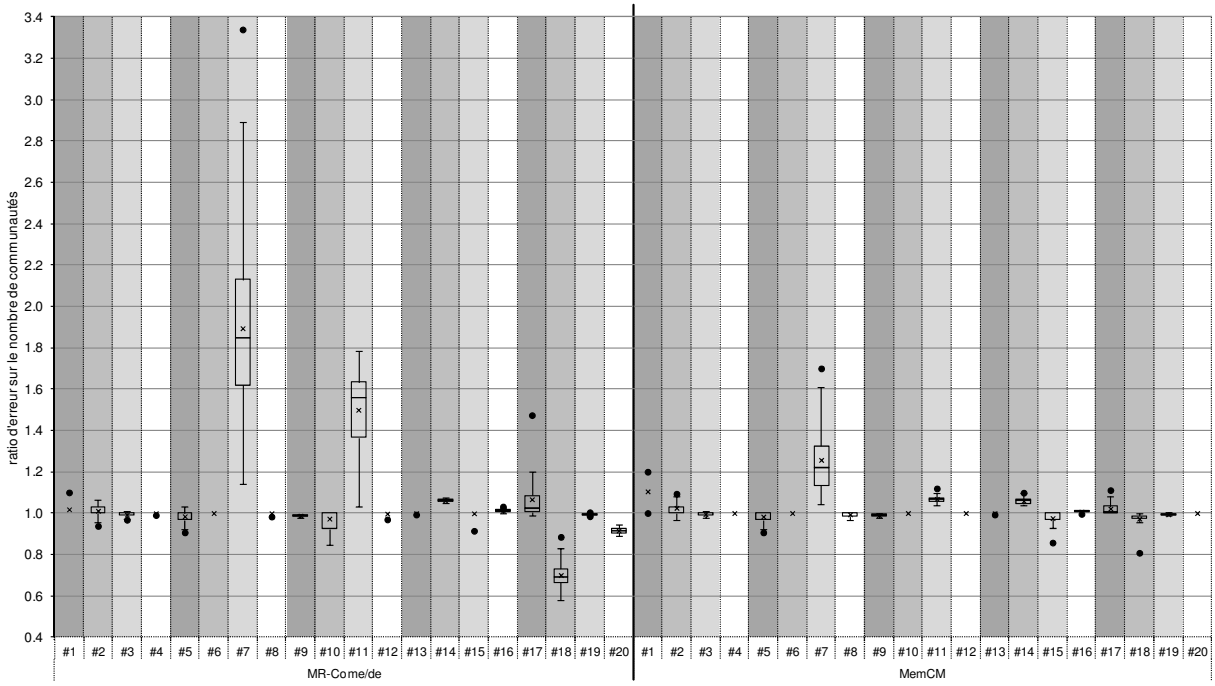


FIGURE 7.7 – Boîtes à moustaches de distribution du ratio d'erreur sur le nombre de communautés des graphes artificiels, comparé entre MR-Come/de et MemCM. L'expérience est réalisée dans les mêmes conditions que les graphiques précédents.

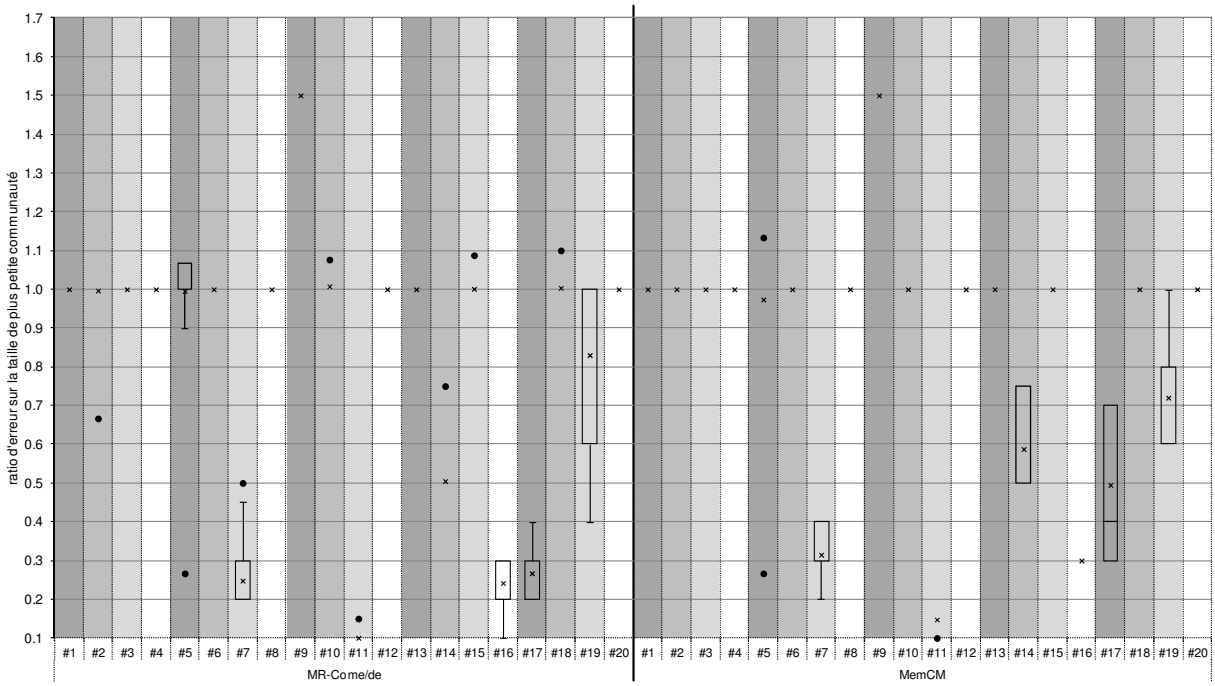


FIGURE 7.8 – Boîtes à moustaches de distribution du ratio d'erreur sur la taille de plus petite communauté des graphes artificiels, comparé entre MR-Come/de et MemCM. L'expérience est réalisée dans les mêmes conditions que les graphiques précédents.

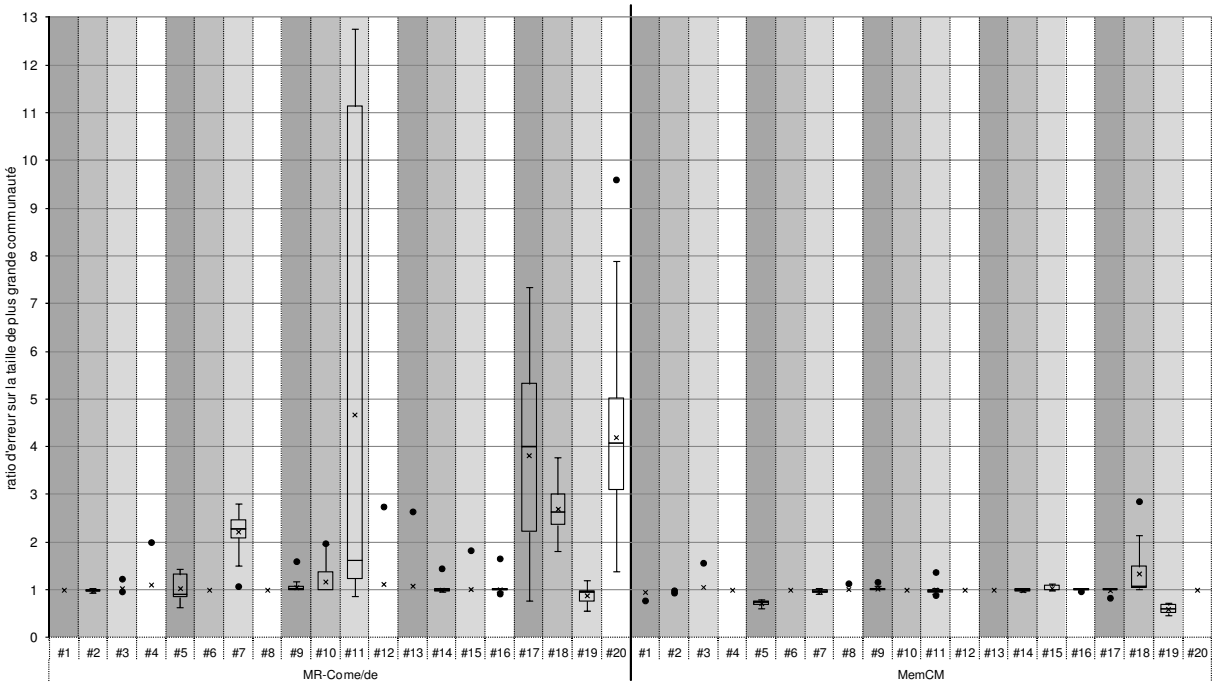


FIGURE 7.9 – Boîtes à moustaches de distribution du ratio d'erreur sur la taille de plus grande communauté des graphes artificiels, comparé entre MR-Come/de et MemCM. L'expérience est réalisée dans les mêmes conditions que les graphiques précédents.

7.4.2 Inconsistance de solution

L'inconsistance de solution comparée entre MR-Come/de et MemCM, à partir de la moyenne de distance NID entre solutions générées prises deux à deux, montre sur la figure 7.10 que l'algorithme MemCM atteint les niveaux de l'algorithme mémétique classique (en comparant les 15 graphes testés pour cet algorithme), en apportant en général une légère amélioration par rapport à MR-Come. Ainsi la moyenne globale dépasse 0.9 ce qui constitue un résultat remarquable et dénote des solutions très proches entre elles. À noter une amélioration importante pour le graphe *C. elegans* et toujours une diversité entre solutions importantes pour *Slashdot*.

Conclusion

Nous avons proposé un algorithme mémétique à double objectif pour la détection de communauté, la modularité guidant l'optimisation locale qui améliore les individus engendrés par croisement et la densité minimale servant de fonction d'adaptation de la population. De plus, la condition de fusion est appliquée à la génération de la population initiale et à l'optimisation des individus. La combinaison d'un croisement qui casse certaines communautés en préservant celles qui sont les mieux adaptées et une condition de fusion qui regroupe de façon pertinente les communautés rend cet algorithme particulièrement adapté au partitionnement des graphes artificiels avec une mesure de similarité moyenne entre partition trouvée et solution de 0.978, valeur remarquable, l'idéal étant 1. Pour les graphes réels, le critère d'existence minimal de communauté est très bien rempli, pour ainsi dire spontanément alors qu'il n'est pas introduit dans l'algorithme. La densité minimale augmente de façon significative ce qui constitue un gage de communautés plus cohésives, la densité minimale renseignant sur le niveau de cohésion interne de la plus mauvaise des communautés. Enfin, l'inconsistance de solution atteint un

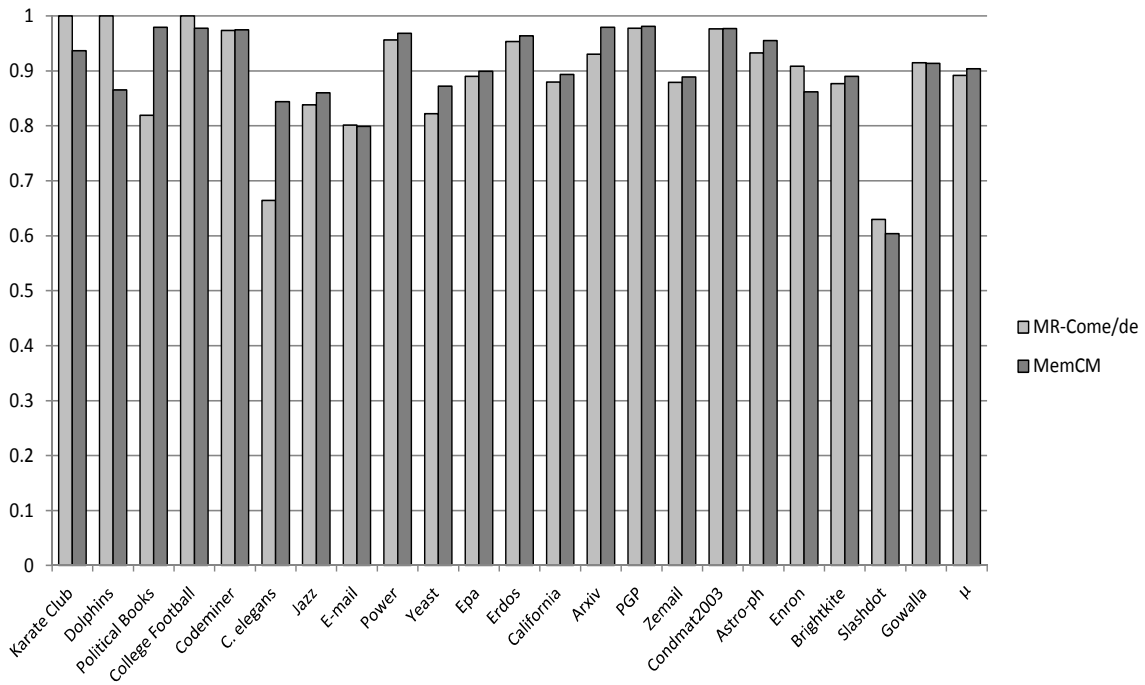


FIGURE 7.10 – Inconsistance de solution de MemCM mesurée par la similarité moyenne entre solutions sur les graphes réels. La version stochastique de l'algorithme est exécutée 20 fois sur chaque graphe. La similarité moyenne est calculée comme la moyenne de la NID de toutes les paires distinctes de solutions. Une similarité proche de 1 indique des solutions structurellement proches et donc une inconsistance liée à la modularité réduite.

niveau très satisfaisant avec une moyenne de similarité au-dessus de 0.9. Ainsi, deux solutions quelconques fournies par l'algorithme pour un même graphe sont très proches, renforçant le sens que l'on peut attribuer au partitionnement et rendant inutiles des exécutions multiples pour sélectionner une partition parmi un grand nombre.

Conclusion générale

Contexte et objectif

La modularité est une mesure de la qualité d'un partitionnement en communautés de grands graphes de terrain, très largement utilisée dans des algorithmes opérationnels, et présentant deux défauts majeurs, dont l'impact sur des graphes réalistes est mal connu, la limite de résolution et l'inconsistance de solution. D'une part les petites communautés sont d'autant plus indétectables par l'optimisation de la modularité qu'elles côtoient de grandes communautés et que le graphe est grand. D'autre part, pour un même graphe, les méthodes fondées sur la modularité sont sensibles à l'ordre d'exploration des nœuds du graphe de sorte qu'avec deux ordonnancements différents, les partitions proposées sont structurellement différentes, suffisamment pour gêner leur interprétation. Notre objectif était de conserver les avantages de la modularité sous une forme qui en diminue les impacts négatifs. Pour y parvenir, nous avons développé quatre nouveaux algorithmes à partir de l'algorithme de Louvain qui offre un bon compromis entre la performance et la vitesse de calcul, testés par un jeu de graphes réels et un jeu de graphes artificiels de caractéristiques variées.

Constat de faiblesse de l'algorithme de Louvain

À partir d'un des algorithmes les plus efficaces pour optimiser la modularité, l'algorithme de Louvain, nous avons établi que la pertinence des partitions trouvées est très insatisfaisante avec une mesure moyenne de similarité de 0.76 environ, pour les graphes artificiels, l'idéal étant de 1. Le nombre de communautés détectées est 2.7 fois plus faible que le nombre attendu. Pour les graphes réels, nous avons fait le constat que la densité minimale de communauté était très faible, du fait de communautés trop grandes, et pouvait sans doute être augmentée sans compromettre l'existence des communautés, appréciée selon le critère le plus faible.

Algorithmes proposés

Notre première approche a consisté à pousser l'optimisation de la modularité à des niveaux jamais atteints jusqu'ici, d'une part en développant l'algorithme Louvain+ qui améliore l'algorithme de Louvain par une phase de raffinement fondée sur le paradigme multiniveau et d'autre part en concevant un algorithme mémétique qui croise efficacement les partitions et les améliore grâce à Louvain+.

Dans une deuxième approche, à la suite du constat que la fusion de communautés pratiquée par l'algorithme de Louvain était à l'origine d'une très grande majorité des erreurs de partitionnement, nous avons formulé une condition de fusion fondée sur la modularité paramétrique et qui élimine une bonne partie des fusions erronées. Pour réduire la limite de résolution présente aussi dans la première phase de l'algorithme de Louvain, nous avons développé une phase par raffinements successifs à échelle décroissante, qui utilise aussi la modularité paramétrique.

Enfin, l'association de l'algorithme mémétique avec l'algorithme de Louvain sous condition de fusion produit d'excellents résultats, en prenant soin de modifier la fonction d'adaptation (objectif de l'algorithme mémétique) qui devient la densité minimale de communauté. L'objectif de maximisation de la modularité n'est conservé que dans l'optimisation locale réalisée par l'algorithme de Louvain sous condition de fusion. Dans cet algorithme biobjectif, les raffinements successifs sont abandonnés.

Résultats

L'algorithme mémétique produit des partitions plus justes, mais il souffre de la limite de résolution d'une fait d'une optimisation pure. Cet algorithme obtient toutefois les meilleurs résultats d'optimisation pure de la modularité, au prix d'un temps de calcul beaucoup plus important que celui de Louvain. La limite de résolution est nettement réduite par l'algorithme sous condition de fusion et raffinements successifs et disparaît pratiquement avec l'algorithme biobjectif (similarité moyenne de 0.98).

Le défaut d'instabilité des partitions est très fortement réduit, dans les mêmes proportions par les algorithmes mémétiques que par la condition de fusion, avec une similarité qui passe d'environ 0.8 pour Louvain à 0.9 pour ces algorithmes. Les partitions proposées à partir d'un même graphe sont ainsi beaucoup plus proches structurellement entre elles.

Notre démonstration repose sur un modèle de génération de graphe artificiel qui reproduit de manière réaliste les caractéristiques des réseaux complexes, sans pour autant capturer toute leur complexité. L'efficacité du partitionnement sur les grands graphes réels est plus délicate à établir, mais nous démontrons tout de même que les communautés sont bien définies et que les solutions proposées possèdent une densité minimale de communauté élevée et donc une meilleure cohésion interne pour la plus "faible" des communautés, en général la plus grande en taille.

Perspectives

La modularité présente des inconvénients difficiles à atténuer sans pour autant gêner dans la pratique les experts des domaines d'application des réseaux complexes, qui utilisent, avec précaution toutefois, les algorithmes fondés sur cette mesure. La limite de résolution est consubstantielle à la modularité du fait de son caractère additif qui constitue par ailleurs son avantage calculatoire. Pour autant, cette mesure n'est pas condamnée si son utilisation est amendée selon trois modalités que nous distinguons et qui devraient faire l'objet de recherches futures. La première que nous avons employée consiste à conditionner les opérateurs de recherche (déplacement de nœud, fusion de communautés) tout en préservant l'objectif d'optimisation de la modularité. La deuxième passe par une recherche multi objectifs en associant la modularité à une fonction qui contrecarre la limite de résolution, la densité par exemple. La troisième, encore inexploitée, utilise de manière indirecte les informations produites par un algorithme d'optimisation de la modularité pour produire une partition. Cette voie est prometteuse, car elle permet de rechercher un recouvrement, dans lequel un nœud peut appartenir à plusieurs communautés, un problème difficile qui constitue encore un défi.

Liste des publications

Congrès internationaux avec comité de sélection

- O. Gach and J.-K. Hao. A memetic algorithm for community detection in complex networks. In C. Coello Coello et al. (Eds.) : 12th International Conference on Parallel Problem Solving from Nature (PPSN XII), Lecture Notes in Computer Science 7492 : 327-336, 2012. Springer.
- O. Gach and J.-K. Hao. Improving the Louvain algorithm for community detection with modularity maximization. In P. Legrand et al. (Eds) : 11th International Conference on Artificial Evolution (EA-2013), Lecture Notes in Computer Science, Springer (A paraître).

Congrès nationaux avec comité de sélection

- O. Gach and J.-K. Hao. Algorithme mémétique pour la détection de communautés. Conférence sur les Modèles et l'Analyse des Réseaux : Approches Mathématiques et Informatique (MARAMI), 2011. (article long)
- O. Gach and J.-K. Hao. Maximisation de la modularité sous condition de fusion. Conférence sur les Modèles et l'Analyse des Réseaux : Approches Mathématiques et Informatique (MARAMI), 2013. (article long)

Article de revue en soumission

- O. Gach and J.-K. Hao. Combined neighborhood tabu search for community detection in complex networks.

Liste des tableaux

Table des figures

Références bibliographiques

[1] A.-L. Barabási, *Linked : The New Science of Networks*, vol. 71. Perseus Publishing, 2002. 14, 16

[2] D. J. Watts, "The new science of networks," *Annual Review of Sociology*, vol. 30, no. 1, pp. 243–270, 2004. 14

[3] J. L. Moreno, *Who Shall Survive ? A New Approach to the Problem of Human Interrelations*, vol. 58. Nervous and Mental Disease Publishing, 1934. 15

[4] J. F. Padgett and C. K. Ansell, "Robust action and the rise of the medici, 1400-1434," *American Journal of Sociology*, vol. 98, no. 6, p. 1259, 1993. 15

[5] M. Mizruchi, *The American corporate network, 1904-1974*. Sage library of social research, Sage Publications, 1982. 15

[6] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003. 15, 37

[7] S. Milgram, "The small-world problem," *Psychology Today*, vol. 1, no. 1, pp. 60–67, 1967. 15

[8] M. E. J. Newman, "Citebase - who is the best connected scientist ? a study of scientific coauthorship networks," 2004. 15

[9] M. Steyvers and J. Tenenbaum, "The large-scale structure of semantic networks : statistical analyses and a model of semantic growth.," *Cognitive Science*, vol. 29, no. 1, pp. 41–78, 2005. 15

[10] R. Albert, H. Jeong, and A.-L. Barabasi, "The diameter of the world wide web," *Nature*, vol. 401, no. September, p. 5, 1999. 15

[11] V. Latora and M. Marchiori, "Is the boston subway system a small-world network," *Physica A*, vol. 314, pp. 109–113, 2002. 15

[12] Q. C. Q. Chen, H. C. H. Chang, R. Govindan, and S. Jamin, "The origin of power laws in internet topologies revisited," 2002. 15

[13] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000. 15

[14] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki, "A comprehensive two-hybrid analysis to explore the yeast protein interactome," *Proceedings of the National Academy of Sciences*, vol. 98, no. 8, pp. 4569–4574, 2001. 15

[15] N. Guelzim, S. Bottani, P. Bourgine, and F. Képès, "Topological and causal structure of the yeast transcriptional regulatory network.," *Nature Genetics*, vol. 31, no. 1, pp. 60–63, 2002. 15

[16] R. Albert, "Scale-free networks in cell biology," *Journal of cell science*, vol. 118, no. 21, pp. 4947–4957, 2005. 15

[17] X. Zhu, M. Gerstein, and M. Snyder, "Getting connected : analysis and principles of biological networks.," *Genes & Development*, vol. 21, no. 9, pp. 1010–1024, 2007. 15

[18] A. Arenas, A. Díaz-Guilera, and C. J. Pérez-Vicente, "Synchronization reveals topological scales in complex networks," *Phys. Rev. Lett.*, vol. 96, p. 114102, Mar 2006. 16

[19] A. Aleksiejuk, J. A. Hołyst, and D. Stauffer, "Ferromagnetic phase transition in barabási–albert networks," *Physica A : Statistical Mechanics and its Applications*, vol. 310, no. 1, pp. 260–266, 2002. 16

[20] J. P. K. Doye, "Network topology of a potential energy landscape : A static scale-free network," *Phys. Rev. Lett.*, vol. 88, p. 238701, May 2002. 16

[21] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, p. 58, 2003. 16

[22] M. E. J. Newman, *Networks : An Introduction*. Oxford University Press, 2010. 16

[23] A. Lesne, "Complex networks : from graph theory to biology," *Letters in Mathematical Physics*, vol. 78, no. 3, pp. 235–262, 2006. 16

[24] A. Barrat, M. Barthélemy, and A. Vespignani, "Réseaux complexes et physique statistique," *Images de la Physique*, pp. 47–53, 2006. 16

[25] A. S. Klovdahl, J. J. Potterat, D. E. Woodhouse, J. B. Muth, S. Q. Muth, and W. W. Darrow, "Social networks and infectious disease : the colorado springs study.," *Social science medicine*, vol. 38, no. 1, pp. 79–88, 1994. 16

[26] D. J. Watts and S. H. Strogatz, "Collective dynamics of ?small-world ? networks.," *Nature*, vol. 393, no. 6684, pp. 440–2, 1998. 17, 18

[27] P. Erdős and A. Rényi, "On the evolution of random graphs," *Evolution*, vol. 5, no. 1, pp. 17–61, 1960. 18

[28] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 9, pp. 2658–2663, 2004. 19, 27

[29] Y. Hu, H. Chen, P. Zhang, M. Li, Z. Di, and Y. Fan, "Comparative definition of community and corresponding identifying algorithm," *Physical Review E*, vol. 78, no. 2, p. 026121, 2008. 19, 85

[30] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010. 22

[31] A. D. Medus and C. O. Dorso, "Alternative approach to community detection in networks," *Physical Review E*, vol. 79, no. 6, p. 066111, 2009. 22

[32] R. Ghosh and K. Lerman, "Community detection using a measure of global influence," *Advances in Social Network Mining and Analysis*, pp. 20–35, 2010. 22

[33] R. Aldecoa and I. Marín, "Deciphering network community structure by surprise," *PloS one*, vol. 6, no. 9, p. e24195, 2011. 22

[34] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004. 22, 27

[35] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence," *Random Structures & Algorithms*, vol. 6, no. 2-3, pp. 161–180, 1995. 23

[36] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral, "Modularity from fluctuations in random graphs and complex networks," *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, vol. 70, no. 2 Pt 2, p. 4, 2004. 23

[37] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, no. 1, pp. 1–14, 2006. 23, 31

[38] J. Reichardt and S. Bornholdt, "Partitioning and modularity of graphs with arbitrary degree distribution," *Physical Review E*, vol. 76, no. 1, p. 015102, 2007. 23

[39] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, pp. 36–41, Jan. 2007. 23

[40] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 78, no. 4, 2008. 24, 36

[41] A. Lancichinetti and S. Fortunato, "Community detection algorithms : a comparative analysis.," *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, vol. 80, no. 5 Pt 2, p. 056117, 2009. 24, 28

[42] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips, "Tolerating the community detection resolution limit with edge weighting," *Physical Review E*, vol. 83, no. 5, p. 056119, 2011. 24

[43] Z. Li, S. Zhang, R.-S. Wang, X.-S. Zhang, and L. Chen, "Quantitative function for community detection," *Physical review E*, vol. 77, no. 3, p. 036109, 2008. 24

[44] B. H. Good, Y.-A. De Montjoye, and A. Clauset, "Performance of modularity maximization in practical contexts.," *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, vol. 81, no. 4 Pt 2, p. 20, 2010. 24

[45] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell system technical journal*, vol. 49, no. 1, pp. 291–307, 1970. 25

[46] E. R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM Journal on Algebraic Discrete Methods*, vol. 3, no. 4, pp. 541–550, 1982. 25

[47] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, p. 14, California, USA, 1967. 26

[48] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007. 26

[49] L. Donetti and M. A. Munoz, "Detecting network communities : a new systematic and efficient algorithm," *Journal of Statistical Mechanics : Theory and Experiment*, vol. 2004, no. 10, p. P10012, 2004. 26

[50] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002. 27, 37

[51] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics : Theory and Experiment*, vol. 2005, no. 09, p. P09008, 2005. 28

[52] H. Zhou and R. Lipowsky, "Network brownian motion : A new method to measure vertex-vertex proximity and to identify communities and subcommunities," *Computational Science-ICCS 2004*, pp. 1062–1069, 2004. 28

[53] P. Pons and M. Latapy, "Computing communities in large networks using random walks," *Computer and Information Sciences-ISCIS 2005*, pp. 284–293, 2005. 28

[54] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007. 28

[55] X. Liu and T. Murata, "Advanced modularity-specialized label propagation algorithm for detecting communities in networks," *Physica A : Statistical Mechanics and its Applications*, December 2009. 28, 71

[56] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," 2008. 28

[57] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, p. 066133, Jun 2004. 28

[58] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, p. 066111, Dec 2004. 29

[59] P. Schuetz and A. Caflisch, "Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement," *Phys. Rev. E*, vol. 77, p. 046112, Apr 2008. 29

[60] S. Kirkpatrick, M. Vecchi, *et al.*, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983. 30

[61] J. H. Holland, *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992. 31

[62] C. Pizzuti, "Community detection in social networks with genetic algorithms," *Proceedings of the 10th annual conference on Genetic and evolutionary computation GECCO 08*, p. 1137, 2008. 31, 62

[63] C. Pizzuti, "Ga-net : A genetic algorithm for community detection in social networks," *Parallel Problem Solving from Nature–PPSN X*, pp. 1081–1090, 2008. 31, 62

[64] M. Tasgin, A. Herdagdelen, and H. Bingol, "Community detection in complex networks using genetic algorithms," *arXiv preprint arXiv :0711.0491*, 2007. 31, 62

[65] C. Shi, Y. Wang, B. Wu, and C. Zhong, "A new genetic algorithm for community detection," *Complex Sciences*, pp. 1298–1309, 2009. 31

[66] Y. Park and M. Song, "A genetic algorithm for clustering problems," in *Proceedings of the Third Annual Conference on Genetic Programming*, pp. 568–575, 1998. 31

[67] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics : Theory and Experiment*, vol. 10, pp. 8–+, Oct. 2008. 31, 32, 53

[68] A. Noack and R. Rotta, "Multi-level algorithms for modularity clustering," *ArXiv e-prints*, Dec. 2008. 31, 32, 71

[69] Z. Lü and W. Huang, "Iterated tabu search for identifying community structure in complex networks," *Phys. Rev. E*, vol. 80, p. 026130, Aug 2009. 31

[70] M. Gong, B. Fu, L. Jiao, and H. Du, "Memetic algorithm for community detection in networks," *Physical Review E*, vol. 84, no. 5, p. 056101, 2011. 31, 62

[71] A. Arenas, J. Duch, A. Fernández, and S. Gómez, "Size reduction of complex networks preserving modularity," *New Journal of Physics*, vol. 9, p. 176, June 2007. 31, 94

[72] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," tech. rep., Citeseer, 1993. 31

[73] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998. 31

[74] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977. 37

[75] V. Krebs, "A network of books about recent us politics sold by the online bookseller amazon.com," http ://www.orgnet.com, 2008. 37

[76] J. Heymann S., Palmier, "Source code structure of a java program," http ://wiki.gephi.org/index.php/Datasets. 37

[77] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Phys. Rev. E*, vol. 72, p. 027104, Aug 2005. 37

[78] P. Gleiser and L. Danon, "Community structure in social and biological networks," *Advances in Complex Systems*, vol. 6, pp. 565–573, 2003. 37

[79] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Phys. Rev. E*, vol. 68, p. 065103, Dec 2003. 37

[80] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks.," *Nature*, vol. 393, no. 6684, pp. 440–2, 1998. 37

[81] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, and N. Zhang, "Topological structure analysis of the protein-protein interaction network in budding yeast.," *Nucleic Acids Research*, vol. 31, no. 9, pp. 2443–2450, 2003. 37

[82] J. Kleinberg, “A network of pages linking www.epa.gov in a search engine,” http ://www.cs.cornell.edu/courses/cs685/2002fa/. 37

[83] J. Grossman, “The erdös number project,” http ://www.oakland.edu/enp/, 2007. 37

[84] J. Kleinberg, “A network of pages matching the query "california" in a search engine,” http ://www.cs.cornell.edu/courses/cs685/2002fa/. 37

[85] KDD, “Cornell kdd cup,” http ://www.cs.cornell.edu/projects/kddcup/, 2003. 37

[86] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, “Models of social networks based on social distance attachment,” *Phys. Rev. E*, vol. 70, p. 056122, Nov 2004. 37

[87] R. Rotta, “Email network,” http ://studiy.tu-cottbus.de/ rrotta/. 37

[88] M. E. J. Newman, “The structure of scientific collaboration networks,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 2, pp. 404–409, 2001. 37

[89] J. Leskovec, J. Kleinberg, and C. Faloutsos, “Graph evolution : Densification and shrinking diameters,” *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, pp. 2–es, 2006. 37

[90] J. Leskovec, K. J. Lang, A. Dasgupta, and M. Mahoney, “Community structure in large networks : Natural cluster sizes and the absence of large well-defined clusters,” *Internet Mathematics*, vol. 6, no. 1, p. 66, 2008. 37

[91] E. Cho, S. A. Myers, and J. Leskovec, *Friendship and mobility*, p. 1082. ACM Press, 2011. 37

[92] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971. 42, 66

[93] D. J. C. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003. 42

[94] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, “Comparing community structure identification,” *Journal of Statistical Mechanics : Theory and Experiment*, vol. 2005, p. P09008, Sept. 2005. 43

[95] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison : Variants, properties, normalization and correction for chance,” *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010. 43

[96] F. Neri, C. Cotta, and P. Moscato, *Handbook of memetic algorithms*, vol. 379. Springer, 2011. 62

[97] Q. Wu and J.-K. Hao, “Memetic search for the max-bisection problem,” *Computers & Operations Research*, vol. 40, no. 1, pp. 166–179, 2013. 64

[98] U. Benlic and J.-K. Hao, “An effective multilevel tabu search approach for balanced graph partitioning,” *Computers & Operations Research*, vol. 38, no. 7, pp. 1066–1075, 2011. 64

[99] V. A. Traag, P. Van Dooren, and Y. Nesterov, "Narrow scope for resolution-free community detection," *Physical Review*, vol. 84, no. 1, p. 016114, 2011. 94

[100] J. M. Kumpula, J. Saramaki, K. Kaski, and J. Kertesz, "Limited resolution in complex network community detection with potts model approach," *European Physical Journal B*, vol. 56, no. 1, p. 5, 2006. 94

[101] A. Lancichinetti and S. Fortunato, "Limits of modularity maximization in community detection," *Arxiv preprint arXiv11071155*, pp. 1–7, 2011. 94

[102] G. Krings and V. D. Blondel, "An upper bound on community size in scalable community detection," *Compare A Journal Of Comparative Education*, p. 4, 2011. 94

l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

Université du Maine

Thèse de Doctorat

Olivier GACH

Algorithmes mémétiques de détection de communautés dans les réseaux complexes

Techniques palliatives de la limite de résolution

Memetic algorithm for community detection in Complex Network

Mitigation techniques to the resolution limit, the main weakness of modularity

Résumé

Les réseaux complexes, issus de relevés de terrain d'origines très variées, en biologie, science de l'information ou sociologie, présentent une caractéristique remarquable dénommée structure communautaire. Des groupes, ou communautés, à l'intérieur du réseau, ont une cohésion interne forte et des liens entre eux plus faibles. Sans connaissance a priori du nombre de communautés, la difficulté réside dans la caractérisation d'un bon partitionnement en communautés. La modularité est une mesure globale de qualité de partitionnement très utilisée qui capture les contraintes de cohésion interne forte et de liens externes faibles. Elle transforme le problème de détection de communautés en problème d'optimisation NP-difficile. Elle souffre d'un défaut, la limite de résolution, qui tend à rendre indétectables les très petites communautés d'autant plus que le réseau est grand. L'algorithme le plus efficace pour optimiser la modularité, dit de Louvain, procède par fusion de communautés. Cette thèse s'attache à modifier cet algorithme pour qu'il réalise majoritairement des fusions pertinentes, qui n'aggravent pas la limite de résolution, en utilisant une condition de fusion. De plus, en l'associant à un algorithme mémétique, les partitions proposées sont très proches des partitions attendues pour des graphes générés par un modèle qui reproduit les caractéristiques des réseaux complexes. Enfin, cet algorithme mémétique réduit fortement l'inconsistance de solution, défaut de la modularité selon lequel deux partitions trouvées à partir d'un examen des nœuds dans un ordre aléatoire, pour le même graphe, peuvent être structurellement très différentes, rendant leur interprétation délicate.

Mots clés

réseaux complexes, graphes de terrain, détection de communautés, modularité, algorithme mémétique, limite de résolution.

Abstract

From various applications, in sociology or biology for instance, complex networks exhib the remarquable property of community structure. Groups, sometimes called communities, has a strong internal cohesion and poor links between them. Whithout prior knowledge of the number of communities, the difficulty lies in the characterization of a good clustering. Modularity is an overall measure of clustering quality widely used to capture the double constraint, internal and external, of well formed communities. The problem became a NP-hard optimization problem. The main weak of modularity is the resolution limit, which tends to make undetectable very small communities especially as the network is large. The algorithm of Louvain, one of the most efficient one to optimize modularity, proceeds by merging communities. This thesis attempts to modify the algorithm so that it mainly produces relevant merges that do not make worse the effects of resolution limit, using a merge condition. In addition, by combining it with a memetic algorithm, proposed clusterings are very close to the expected ones for graphs generated by a model that reproduces the characteristics of complex networks. Finally, the memetic algorithm greatly reduces the inconsistency of solution, another weakness of modularity such that, for the same graph, two partitions found from an exploration of nodes in a random order can be structurally very different, making them difficult to interpret.

Key Words

complex network, community detection, clustering, modularity, memetic algorithm, resolution limit.