Algorithme mémétique pour la détection de communautés

Olivier Gach[*,**] — **Jin-Kao Hao**[**]

[*] *LIUM & IUT*
Université du Maine
Av. O. Messiaen
72085 Le Mans, France
olivier.gach@univ-lemans.fr

[**] *LERIA*
Université d'Angers
2 Boulevard Lavoisier
49045 Angers Cedex 01, France
hao@info.univ-angers.fr

RÉSUMÉ. *La recherche de communautés est un problème important dans le domaine des réseaux complexes. La modularité est probablement la mesure fondée sur un partitionnement la plus populaire pour la détection de communautés dans un réseau complexe représenté sous la forme d'un graphe. Nous présentons un nouvel algorithme hybride mêlant un algorithme génétique à base de population de solutions et une recherche locale adaptée au problème. L'évaluation expérimentale de 11 graphes de référence dans ce domaine montre que l'algorithme proposé est capable de trouver les meilleures solutions et d'en améliorer certaines en maximisant la modularité.*

ABSTRACT. *Community detection is an important issue in the field of complex networks. Modularity is probably the most popular partition-based measure for community detection of networks represented as graphs. We present in this article a hybrid algorithm mixing a genetic approach based on population of solutions and a dedicated local search. Experimental evaluations on a collection of 11 well-known benchmark graphs show that the proposed algorithm is able to find the current best solutions and even improve some of them in terms of maximum modularity.*

MOTS-CLÉS : *heuristique, détection de communautés, graphes de terrain, réseaux complexes, partitionnement de graphe, modularité, optimisation combinatoire*

KEYWORDS: *heuristic, community detection, complex networks, graph partitioning, modularity, combinatorial optimization*

2[e] soumission à *MARAMI 2011 - Grenoble*, le 5 octobre 2011.

1. Introduction

Les réseaux complexes, aussi appelés graphes de terrain, apparaissent dans de très nombreuses applications dont les réseaux sociaux sur le Web sont une illustration paradigmatique. Contrairement aux graphes aléatoires, les réseaux complexes affichent généralement des propriétés topologiques non-triviales qui caractérisent la connectivité d'un réseau et impactent la dynamique des processus appliqués au réseau. L'analyse des réseaux complexes permet de découvrir leurs caractéristiques spécifiques et contribue à comprendre leur morphogénèse. Une littérature extrêmement abondante a été consacrée à ces analyses structurelles et métrologiques. Cependant, différents problèmes restent ouverts, et la recherche d'algorithmes performants demeure un défi pour la recherche.

Dans cet article, nous nous intéressons à la détection de communautés selon la mesure de modularité (Q) qui a fait l'objet de nombreux travaux de recherche depuis 2004 [NEW 04b]. C'est une mesure de qualité d'une partition qui pour un graphe donné permet de qualifier de "meilleure" partition, au sens de cette mesure, celle qui possède la plus grande modularité. Le problème d'optimisation de Q est NP-difficile, les approches par heuristiques deviennent indispensables à partir d'une certaine taille.

Un réseau complexe est représenté par un graphe dans lequel chaque sommet ou nœud correspond à un élément du réseau et chaque arête à un lien entre deux éléments, qui prend un sens selon le domaine d'application du réseau étudié. Notre travail se limite aux graphes non orientés et aux situations où chaque nœud appartient à une et une seule communauté. Pour un graphe pondéré $G = (V, E, w)$, où V est l'ensemble des nœuds, E l'ensemble des arêtes et $w : V \times V \longmapsto \mathbb{R}$ la fonction de poids, et pour une partition $\{C_1, C_2, ..., C_{n_c}\}$ de ce graphe, la modularité peut être définie par :

$$Q = \sum_{i=1}^{n_c} \left[\frac{l_i}{W(V,V)} - \left(\frac{d_i}{W(V,V)} \right)^2 \right] \quad (1)$$

La fonction de poids w est étendue aux communautés par la fonction nommée W, c'est-à-dire que pour une communauté C_i, $W(C_i) = \sum_{v,v' \in C_i} w(v, v')$. De plus, l_i désigne le nombre d'arêtes de la communauté, soit $l_i = W(C_i, C_i)$ et d_i est la somme des degrés des nœuds appartenant à la communauté, soit $d_i = \sum_{v \in C_i} deg(v)$. Par la suite, nous notons n le nombre de nœuds du graphe, et m son nombre d'arêtes.

De nombreuses méthodes ont été proposées dans la littérature pour l'optimisation, autrement dit la maximisation, de Q : approche gloutonne agglomérative [NEW 04a], recherche locale par déplacement de nœuds [SCH 08], recherche multi-niveau associée à la méthode gloutonne [NOA 08] ou associée aux déplacements de nœuds [BLO 08]. Les méthodes qui utilisent la technique gloutonne par fusion de communautés sont rapides mais l'optimisation de Q est médiocre. D'autres méthodes offrent un bon compromis mais ne parviennent pas aux meilleurs résultats pour tous les graphes testés. Nous présentons dans cet article une méthode hybride fondée sur un algorithme mémétique qui mêle une technique évolutionnaire avec un croisement de partitions efficace pour la diversification et une recherche locale utile à l'intensification de la

recherche. Des études d'algorithmes génétiques appliqués à la détection de communautés ont montré le potentiel de cette approche (voir par exemple [DIJ 10]). Notre objectif est de montrer qu'un algorithme hybride peut être très efficace pour optimiser la modularité, comparativement aux méthodes heuristiques dédiées à cette optimisation.

2. Algorithme

Les algorithmes mémétiques [MOS 99] ont montré leur efficacité pour apporter des solutions satisfaisantes à des problèmes d'optimisation combinatoire réputés difficiles. Ils combinent typiquement une méthode évolutionnaire à base de population de solutions et une recherche locale, la première servant à l'exploration de l'espace de recherche, la seconde à l'examen détaillé d'une zone particulière. La population initiale, généralement de taille fixe, est constituée d'individus qui sont des solutions du problème. A chaque génération, elle évolue par deux mécanismes :

– la production de nouvelles solutions par un opérateur de combinaison de solutions existantes (le croisement) ;

– la sélection d'une partie des nouvelles solutions qui prennent la place d'anciennes solutions dans la population avec deux objectifs parfois antagonistes, améliorer la qualité générale de la population et préserver sa diversité.

Ces mécanismes s'inspirent de l'évolution naturelle. Le croisement du matériel génétique des deux parents engendre une progéniture différente à la condition que les deux parents soient génétiquement différents. La pression du milieu opère la sélection d'individus bien adaptés au détriment des moins adaptés à ce milieu. Par analogie, le matériel génétique est représenté dans l'algorithme par la structure interne des solutions constituant la population, dépendante du problème d'optimisation étudié. La sélection naturelle se traduit par une fonction d'objectif, en général la fonction à optimiser, qui permet de distinguer les solutions à conserver dans la population.

L'insertion d'une progéniture dans la population est aussi soumise à une mesure de distance entre solutions qui évite l'homogénéisation de la population. Ainsi l'insertion est guidée par la fonction d'objectif à optimiser, sous contrainte d'une distance minimale à préserver entre solutions. L'algorithme enchaîne les générations jusqu'à ce qu'une condition d'arrêt soit vérifiée.

2.1. *Schéma principal*

La population est formée d'individus qui sont des solutions du problème, c'est-à-dire des partitions du graphe donné. La fonction d'objectif qui guide le processus d'évolution est la modularité à maximiser. Nous avons opté pour une population de taille fixe μ qui augmente à chaque génération d'un seul individu prenant la place du plus mauvais élément dans la population (stratégie élitiste de type $\mu + 1$). L'insertion

est soumise à une fonction de distance qui assure la diversité de la population, décrite en section 2.4. L'optimisation de la solution engendrée utilise le même algorithme que pour la génération de la population initiale.

Le schéma général de notre algorithme mémétique est le suivant :

1) génération de la population initiale par un algorithme multi-niveau très rapide (section 2.2) ;

2) choix de deux individus et croisement de ceux-ci pour produire une nouvelle solution (section 2.3) ;

3) amélioration de la progéniture par une optimisation locale adaptée au problème (section 2.3) ;

4) insertion dans la population de la progéniture selon les fonctions d'objectif et de distance (section 2.4) ;

5) boucle à l'étape 2 (nouvelle génération), si la condition d'arrêt n'est pas vérifiée.

Notre algorithme s'arrête après un nombre de générations, fixé au préalable, sans amélioration de la modularité supérieure au seuil de 10^{-6}.

2.2. *Population initiale*

Chaque solution de la population initiale est obtenue par une exécution de l'algorithme de Louvain [BLO 08] amélioré. Notre amélioration consiste à chaque niveau à optimiser Q par la méthode *vertex mover* qui déplace les nœuds vers une communauté voisine [SCH 08] jusqu'à ce que Q n'augmente plus. A chaque niveau, les nœuds sont examinés dans un ordre aléatoirement défini pour assurer la diversité des solutions obtenues.

La méthode procède d'abord par une phase de contraction où, à chaque niveau, les communautés sont regroupées en super-communautés. Le partitionnement d'un graphe de niveau est réalisé par la méthode *vertex mover*, qui démarre avec un partitionnement total dans lequel chaque nœud forme une communauté. Au premier niveau, le graphe traité est le graphe d'origine à partitionner. Ensuite, pour passer au niveau supérieur, une fois *vertex mover* exécuté, la partition obtenue est transformée en *graphe des communautés* dans lequel chaque nœud devient une communauté. Les arêtes de ce graphe sont pondérées de sorte que la modularité soit la même [ARE 07], c'est-à-dire avec le poids $W(C_i, C_j)$ si les extrémités de l'arête correspondent au niveau inférieur aux communautés C_i et C_j. De plus chaque nœud correspondant à la communauté C_i est muni d'une boucle pondérée par l_i (voir formule 1). L'algorithme s'arrête au niveau où le partitionnement total ne peut pas être amélioré (la modularité est maximale dans la situation où chaque nœud est isolé dans sa propre communauté), le passage à un niveau supérieur devenant inutile.

Dans une seconde phase d'expansion, l'algorithme procède à l'envers en partant du partitionnement de plus haut niveau. Il s'agit de la partition finale qui pour l'instant

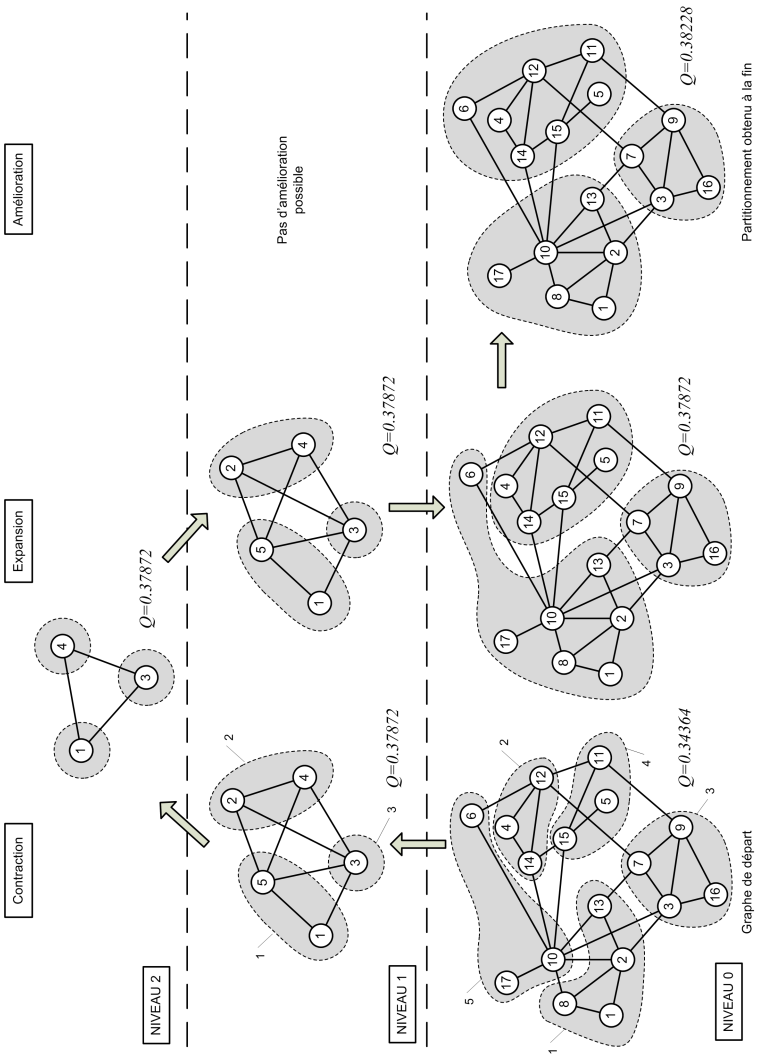

Figure 1. *Illustration de l'algorithme de Louvain amélioré par* vertex mover. *Le graphe de départ est partitionné en 5 groupes au niveau 0 qui forment au niveau supérieur un graphe de 5 nœuds (les poids des arêtes ne figurent pas sur le schéma). Au niveau 2, le partitionnement de départ où chaque nœud est isolé dans une communauté ne peut pas être amélioré en déplaçant un nœud, la première phase s'arrête donc là. Dans la phase d'expansion, le partitionnement obtenu au plus haut niveau est appliqué en descendant niveau par niveau. On constate au niveau 0 une amélioration de Q en déplaçant le nœud 6 vers une communauté voisine. Ce nœud est dès le départ mal placé du point de vue du maximum de modularité et seul un raffinement après l'expansion peut corriger cette erreur.*

contient des sous-communautés dans les niveaux inférieurs. Cette partition est appliquée au niveau immédiatement inférieur, c'est-à-dire que les sous-communautés sont supprimées, leurs nœuds étant versés dans la communauté de niveau supérieur qui les englobe. A ce stade, une application de *vertex mover* permet une amélioration de Q. Ce processus est répété jusqu'au plus bas niveau. Le mécanisme est illustré dans la figure 1.

Nous avons constaté qu'il n'est pas nécessaire de pousser le *vertex mover* au bout, jusqu'aux plus infimes améliorations, car une perte de précision dans le déplacement des nœuds dans la phase de contraction est corrigée par un *vertex mover* plus poussé en phase d'expansion. Ainsi, dans la phase de contraction, nous arrêtons la procédure *vertex mover* si une passe qui examine tous les nœuds produit une augmentation de la modularité inférieure à $\epsilon_1 = 10^{-2}$ alors que le seuil dans la phase d'expansion est fixé à $\epsilon_2 = 10^{-6}$. En fixant ϵ_1 à 10^{-2} au lieu de 10^{-6}, les performances de maximisation de Q sont identiques mais le temps d'exécution est fortement réduit.

2.3. *Croisement et amélioration locale*

A chaque génération, une progéniture est engendrée à partir de deux individus, c'est-à-dire deux partitions présentes dans la population, choisies au hasard. L'ensemble des communautés des deux parents (C_1, C_2) est numéroté 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 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 première 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 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 2).

Ce mécanisme ne fonctionne qu'à la condition que les parents ne soient pas trop proches structurellement. Ceci est assuré par une fonction de distance utilisée à la mise à jour de la population. L'individu ainsi obtenu est optimisé tout simplement par l'algorithme de Louvain amélioré décrit plus haut. A la différence de la génération de la population initiale, le *vertex mover* de la première phase de contraction est appliqué à la partition à optimiser.

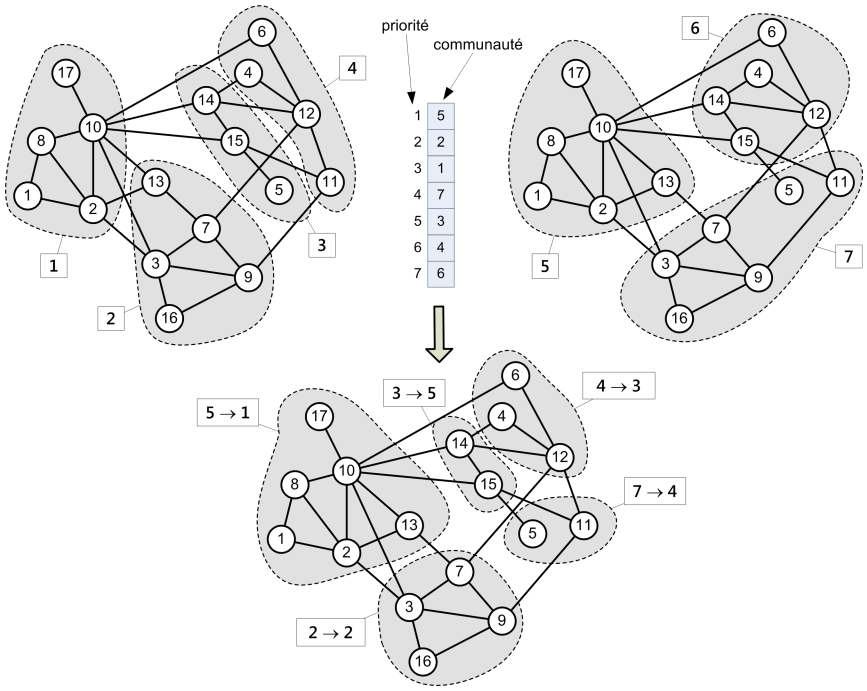

Figure 2. *Illustration de l'opérateur de croisement. La communauté n° 5 a la plus haute priorité, elle est intégralement préservée et deviendra la communauté n° 1 dans la progéniture. La communauté n° 2, qui vient du premier parent (à gauche) suit dans l'ordre des priorités. Tous ces nœuds sont conservés 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.*

Nos tests ont montré qu'un choix aléatoire des priorités est aussi efficace qu'un choix orienté par la valeur de modularité. De plus, le choix aléatoire des parents est préférable à un choix orienté.

2.4. *Mise à jour de la population*

Nous utilisons comme fonction de distance un Rand Index [RAN 71] rapide, limité 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 graphes de terrain ($m \ll n^2$). Le calcul est simple : la valeur 0 est associée à chaque arête dont les extrémités appartiennent à la même communauté dans les deux partitionnements ou appartiennent à des communautés différents dans les deux partitionnements. Si elles appartiennent à la même communauté dans un partitionnement 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 isolé dans une 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 *vertex mover* 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 S, la distance minimale d_S avec les solutions actuelles de la population est calculée. Si $d_S > \delta_{min}$, δ_{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 δ_{min} avec *tous* les individus de la population. Nos tests ont montré qu'une condition stricte allonge le temps de calcul sans améliorer la convergence.

3. Résultats expérimentaux

Nous avons soumis notre algorithme à 11 graphes réels communément utilisés pour la détection de communautés et présentés dans la table 1. La taille de population d'un algorithme mémétique est typiquement de taille réduite (quelques dizaines d'individus). Nous avons choisi $\mu = 30$ qui fournit un compromis optimisation/rapidité satisfaisant, d'après nos tests avec $\mu \in \{20, 30, 40\}$. Nous avons également testé trois valeurs de distance minimale 0.1, 0.01 et 0.001 et constaté que $\delta_{min} = 0.01$ est satisfaisant. La condition d'arrêt est une succession de 1000 générations sans amélioration significative de la modularité (supérieure à 10^{-6}).

Les résultats présentés, avec le temps d'exécution moyen et la modularité moyenne et maximale, synthétisent 20 exécutions du programme écrit en Pascal et compilé, sur un ordinateur PC muni d'un processeur Intel Core i7 870 2.93 GHz avec 8 Go de mémoire vive.

Tableau 1. *Graphes de test pour la détection de communautés, de tailles petites et moyennes, couramment utilisés dans la littérature.*

Graphe	Description	n	m	Source
Karate Club	Zachary karate club network	34	78	[ZAC 77]
Dolphins	dolphin association network	62	159	[LUS 03]
Political Books	network of co-purchased political books	105	441	[KRE 08]
College Football	network of games between college football teams	115	613	[GIR 02]
Jazz	jazz musician collaborations network	198	2742	[GLE 03]
C. elegans	metabolic network for the nematode C.Elegans	453	2025	[DUC 05]
E-mail	university e-mail network	1133	5451	[GUI 03]
Erdos	Erdös collaboration network	6927	11850	[GRO 07]
Arxiv	network of scientific papers and their citations	9377	24107	[KDD 03]
PGP	trust network of mutual signing of cryptography keys	10680	24316	[BOG 04]
Condmat2003	scientific coauthorship network in condensed-matter physics	27519	116181	[NEW 01]

3.1. *Algorithme de Louvain amélioré*

Dans le tableau 2, nous comparons l'algorithme de Louvain avec notre version améliorée (Louvain+). Nous y avons ajouté le graphe *Web nd.edu* [ALB 99] de 325729 nœuds et 1090108 arêtes. D'après ces résultats, notre amélioration de l'algorithme est pertinente car la modularité est supérieure ou égale à celle de la version d'origine pour tous les graphes. Le temps d'exécution est augmenté d'environ 30% sans que la complexité de l'algorithme ne change. Nous avons choisi cette méthode car elle offre de loin le meilleur compromis rapidité/optimisation étant entendu que l'algorithme mémétique sera moins efficace si les solutions de départ sont très mauvaises.

Une analyse rapide de ces résultats semble indiquer que la complexité de cet algorithme est en $O(m)$, comme les auteurs de l'algorithme d'origine le suggèrent dans leur article.

3.2. *Algorithme mémétique*

Nous constatons, au vu des résultats présentés en table 3, que l'algorithme Louvain+ est efficace comparativement aux méthodes existantes en atteignant ou dépassant les meilleurs résultats pour tous les graphes à l'exception de *Erdos*, en considérant des arrondis à la décimale supérieure. L'algorithme mémétique offre de très bonnes performances en dépassant les meilleurs résultats pour tous les graphes à partir de *C.*

Tableau 2. *Résultats de 20 exécutions de l'algorithme de Louvain original et de l'algorithme de Louvain amélioré par un rafinement dans la phase d'expansion. La modularité maximale ainsi que le temps d'exécution moyen en secondes pour 1000 exécutions sont fournis pour chaque algorithme.*

	Louvain		**Louvain+**	
	Q	**Tps**	**Q**	**Tps**
Karate Club	0.4198	0.0	0.4198	0.0
Dolphins	0.5278	0.8	0.5278	0.8
Political Books	0.5270	0.8	0.5273	1.6
College Football	0.6046	0.8	0.6046	2.4
Jazz	0.4452	2.3	0.4452	5.4
C. elegans	0.4442	3.9	0.4487	7.8
E-mail	0.5711	15.7	0.5806	18.7
Erdos	0.6976	37.6	0.7151	46.9
Arxiv	0.8147	85.9	0.8219	113.9
PGP	0.8840	68.6	0.8850	86.5
Condmat2003	0.8102	389.4	0.8151	536.0
Web nd.edu	0.9377	4049.1	0.9399	5343.9

elegans, c'est-à-dire pour une taille approximativement supérieure à 400 nœuds. Il est probable que l'optimum global soit déjà atteint pour les plus petits graphes, pour lesquels l'algorithme n'apporte aucune amélioration mais donne tout de même les mêmes résultats.

L'algorithme présente une constance de performance qui le qualifie pour tous les graphes testés. De plus, cette constance se retrouve à chaque instance d'exécution puisque la moyenne est de peu inférieure au maximum. D'ailleurs, pour tous les graphes, la moyenne est supérieure ou égale aux meilleurs résultats connus.

4. Conclusion

Notre travail s'est concentré sur la recherche d'un algorithme évolutionnaire qui, tout en utilisant les méthodes existantes, s'appuie sur un opérateur de croisement original pour la détection de communautés. Cet opérateur, associé à une fonction de distance simple, prouve son efficacité d'après nos résultats pour explorer plus efficacement l'espace de recherche que les méthodes classiques. Ainsi, un algorithme mémétique combinant des techniques d'algorithme à base de population et une recherche locale fournit une bonne optimisation de la modularité égale ou supérieure aux meilleurs algorithmes actuels, pour 11 graphes réels testés. Ceci est obtenu au détriment du temps d'exécution qui est plus important que pour les méthodes existantes.

Tableau 3. *Résultats de 20 exécutions de l'algorithme mémétique sur les 11 graphes réels. La colonne MRC donne les meilleurs résultats connus avec leur source (*a *pour [LIU 09],* b *pour [NOA 08],* c *pour [LU 09] et* d *pour [BLO 08]. Les colonnes Initial et Mémétique donnent la moyenne de modularité et la meilleure modularité (suivi entre parenthèses du nombre de communautés de la meilleure solution) dans la population initiale (algorithme de Louvain amélioré) et après l'exécution de l'algorithme mémétique.*

Graphe	**MRC**	**Louvain+**		**Mémétique**		
		moy Q	**max Q**	**moy Q**	**max Q**	**Tps**
Karate Club	0,4198 (a, b, c)	0,4198	0,4198 (4)	0,4198	0,4198 (4)	0,4
Dolphins	0,529 (a)	0,5283	0,5286 (5)	0,5286	0,5286 (5)	0,8
Political Books	0,527 (a)	0,5273	0,5273 (5)	0,5273	0,5273 (5)	1,4
College Football	0,605 (a)	0,6046	0,6046 (10)	0,6046	0,6046 (10)	2,1
Jazz	0,4452 (c)	0,4452	0,4452 (4)	0,4452	0,4452 (4)	8,1
C. elegans	0,452 (a)	0,4489	0,4511 (9)	0,4529	0,4533 (9)	11,9
E-mail	0,582 (a)	0,5803	0,5823 (11)	0,5829	0,5829 (10)	32,5
Erdos	0,7162 (b)	0,7151	0,7158 (34)	0,7180	0,7183 (36)	100,4
Arxiv	0,813 (d)	0,8219	0,8227 (56)	0,8252	0,8256 (54)	290,9
PGP	0,8841 (a, b)	0,8849	0,8856 (104)	0,8866	0,8867 (97)	397,9
Condmat2003	0,8146 (b)	0,8151	0,8155 (75)	0,8170	0,8174 (75)	2147,7

Des tests plus poussés, en particulier sur des graphes générés, montrent que ces nouvelles performances confortent les idées généralement admises sur la modularité, avec une tendance à créer des communautés pouvant être de taille importante en incluant les plus petites, impossibles à détecter par cette mesure [FOR 07]. Les perspectives d'amélioration de l'algorithme sont nombreuses, en introduisant par exemple une optimisation plus sophistiquée, comme la recherche tabou ou en faisant varier la distance minimale pendant l'exécution. Il reste à déterminer si la modularité offre encore un potentiel d'optimisation pertinent pour la détection de communautés.

Remerciements

Nous remercions les rapporteurs pour les suggestions et questions. Ce travail est partiellement financé par la Région Pays de la Loire dans le cadre des projets "LigeRO" (2009-2013) et "Radapop" (2009-2013).

5. Bibliographie

[ALB 99] ALBERT R., JEONG H., BARABÁSI A. L., « The diameter of the world wide web », *Nature*, vol. 401, 1999, p. 130–131.

[ARE 07] ARENAS A., DUCH J., FERNÁNDEZ A., GÓMEZ S., « Size reduction of complex networks preserving modularity », *New Journal of Physics*, vol. 9, n° 6, 2007, page 176, Institute of Physics Publishing.

[BLO 08] BLONDEL V. D., GUILLAUME J., LAMBIOTTE R., LEFEBVRE E., « Fast unfolding of communities in large networks », *Journal of Statistical Mechanics : Theory and Experiment*, vol. 10, 2008, p. 8-+.

[BOG 04] BOGUÑÁ M., PASTOR-SATORRAS R., DÍAZ-GUILERA A., ARENAS A., « Models of social networks based on social distance attachment », *Phys. Rev. E*, vol. 70, n° 5, 2004, page 056122, American Physical Society.

[DIJ 10] DI JIN DONGXIAO HE D. L., BAQUERO C., « Genetic Algorithm with Local Search for Community Mining in Complex Networks », *Tools with Artificial Intelligence, 2010. ICTAI '10. 22nd International Conference on*, IEEE, octobre 2010.

[DUC 05] DUCH J., ARENAS A., « Community detection in complex networks using extremal optimization », *Phys. Rev. E*, vol. 72, n° 2, 2005, page 027104, American Physical Society.

[FOR 07] FORTUNATO S., BARTHÉLEMY M., « Resolution limit in community detection », *Proceedings of the National Academy of Sciences*, vol. 104, n° 1, 2007, p. 36–41.

[GIR 02] GIRVAN M., NEWMAN M. E. J., « Community structure in social and biological networks », *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, n° 12, 2002, p. 7821-7826.

[GLE 03] GLEISER P., DANON L., « Community structure in social and biological networks », *Advances in Complex Systems*, vol. 6, 2003, p. 565-573.

[GRO 07] GROSSMAN J., The Erdös number project, http ://www.oakland.edu/enp/, 2007.

[GUI 03] GUIMERÀ R., DANON L., DÍAZ-GUILERA A., GIRALT F., ARENAS A., « Self-similar community structure in a network of human interactions », *Phys. Rev. E*, vol. 68, n° 6, 2003, page 065103, American Physical Society.

[KDD 03] KDD, Cornell KDD Cup, http ://www.cs.cornell.edu/projects/kddcup/, 2003.

[KRE 08] KREBS V., A network of books about recent US politics sold by the online bookseller amazon.com, http ://www.orgnet.com, 2008.

[LU 09] LÜ Z., HUANG W., « Iterated tabu search for identifying community structure in complex networks », *Phys. Rev. E*, vol. 80, n° 2, 2009, page 026130, American Physical Society.

[LIU 09] LIU X., MURATA T., « Advanced modularity-specialized label propagation algorithm for detecting communities in networks », *Physica A : Statistical Mechanics and its Applications*, , 2009.

[LUS 03] LUSSEAU D., SCHNEIDER K., BOISSEAU O. J., HAASE P., SLOOTEN E., DAWSON S. M., « The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations », *Behavioral Ecology and Sociobiology*, vol. 54, n° 4, 2003, p. 396–405.

[MOS 99] MOSCATO P., « *Memetic algorithms : a short introduction* », p. 219–234, McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[NEW 01] NEWMAN M. E. J., « The structure of scientific collaboration networks », *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, n° 2, 2001, p. 404-409.

[NEW 04a] NEWMAN M. E. J., « Fast algorithm for detecting community structure in networks », *Phys. Rev. E*, vol. 69, n° 6, 2004, page 066133, American Physical Society.

[NEW 04b] NEWMAN M. E. J., GIRVAN M., « Finding and evaluating community structure in networks », *Phys. Rev. E*, vol. 69, n° 2, 2004, page 026113, American Physical Society.

[NOA 08] NOACK A., ROTTA R., « Multi-level algorithms for modularity clustering », *ArXiv e-prints*, , 2008.

[RAN 71] RAND W. M., « Objective Criteria for the Evaluation of Clustering Methods », *Journal of the American Statistical Association*, vol. 66, n° 336, 1971, p. 846–850, American Statistical Association.

[SCH 08] SCHUETZ P., CAFLISCH A., « Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement », *Phys. Rev. E*, vol. 77, n° 4, 2008, page 046112, American Physical Society.

[ZAC 77] ZACHARY W. W., « An information flow model for conflict and fission in small groups », *Journal of Anthropological Research*, vol. 33, 1977, p. 452–473.