MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ MOULOUD MAMMERI DE TIZI OUZOU
FACULTÉ DE GÉNIE ÉLECTRIQUE ET D'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

THÈSE DE DOCTORAT

SPECIALITÉ : INFORMATIQUE

Présentée par

Fatiha SOUAM - AÏT EL HADJ

Sujet :

Approche de détection de communautés chevauchantes dans des réseaux bipartis

Devant le jury d'examen composé de :

-Mr Lalam Mustapha ;	Professeur,	UMMTO	Président
-Mr Baba-Ali Ahmed Riadh ;	MC (A),	USTHB	Directeur de thèse
-Mr Lebbah Mustapha ;	MC HDR,	PARIS 13	Examinateur
-Mr Ahmed Ouamer Rachid ;	Professeur,	UMMTO	Examinateur
-Mr Mezghiche Mohamed ;	Professeur,	UMBB	Examinateur
-Mr Hamadouche Djamel ;	Professeur,	UMMTO	Examinateur

Soutenue le : 20/10/2013

A la mémoire de mes chers parents

A toute ma famille....

Remerciements

Gloire et Louange à Dieu, le tout Puissant, de m'avoir donné courage et persévérance pour finaliser cette thèse malgré les circonstances pénibles qui ont précédées la disparition de mon cher et regretté père, que Dieu ait son âme et l'accueille en son vaste paradis.

Je remercie très respectueusement Mr Mustapha Lalam, Professeur au département d'informatique (FGEI, UMMTO), d'avoir accepté d'être président de ce jury.

Je voudrais adresser mes vifs remerciements à Mr Mustapha Lebbah, Maître de conférences HDR à l'université Paris 13 (France), d'avoir accepté très gentiment de faire partie de ce jury. Je le remercie particulièrement pour m'avoir permis de travailler à ses côtés, en acceptant de m'accueillir au sein de l'équipe A3 durant les différents stages que j'ai effectués au LIPN de Paris 13.

Mes sincères remerciements s'adressent aussi à Mr Mohamed Mezghiche, Professeur à l'UMBB (Boumerdes) d'avoir répondu favorablement pour renforcer la composante de ce jury.

Je remercie également Mr Rachid Ahmed Ouamer, Professeur au département d'informatique (FGEI, UMMTO), d'avoir accepté d'être membre de ce jury.

Je remercie chaleureusement Mr Djamel Hamadouche, Professeur à la faculté des sciences de l'UMMTO d'avoir accepté gentiment, d'être examinateur de cette thèse.

De même, je dirais un grand merci à mon directeur de thèse, Mr Ahmed Riadh Baba-ali, Maitre de conférences à l'USTHB (Alger) d'avoir accepté de m'encadrer. Je le remercie également de m'avoir fait découvrir les algorithmes évolutionnaires, et particulièrement, pour m'avoir laissé une large marge de manœuvre dans mes investigations.

Enfin, je ne saurais clore ce volet sans adresser mes remerciements chaleureux aux membres de ma famille, en particulier :

Mon Mari Mr Aïtelhadj Ali, Maître de Conférences (UMMTO), de m'avoir fait bénéficier de son expérience et aidée tout au long de cette thèse.

Mon frère Morad Souam, Consultant Infrastructure auprès de Grands Comptes Européens, pour l'accueil qu'il m'a toujours réservé durant mes séjours à Paris.

Mon frère Rabah Souam, Chercheur au CNRS, pour m'avoir accueillie en 2009 au sein de son équipe de l'institut de mathématique de Jussieu, Université Paris 7.

Mon frère Mohand Said Souam, Professeur agrégé en sciences économiques, à l'université Paris Ouest Nanterre La Défense, pour son aide précieuse et pour l'intérêt constant qu'il n'a cessé de témoigner à mon égard durant ces années de thèse. Je le remercie également pour avoir participé à l'amélioration de la rédaction de ce manuscrit en dépit de ses nombreuses occupations.

Je remercie affectueusement, ma fille et mon fils, source de mon inspiration, pour leur compréhension durant ces deux dernières années.

Résumé

La perception actuelle de la notion de réseau a permis de réaliser des progrès significatifs pour la compréhension des systèmes complexes. L'une des caractéristiques les plus pertinentes des réseaux représentant des systèmes réels est l'existence de zones plus densément connectées que d'autres. Ces zones sont habituellement appelées communautés. De nombreux algorithmes ont été conçus pour découvrir les structures des communautés dans des réseaux. Ces algorithmes sont pour la plupart dédiés pour détecter des communautés disjointes. Très peu d'entre eux sont destinés à découvrir des communautés chevauchantes, particulièrement les réseaux bipartis, n'ont presque pas été explorés pour la détection de ce type de communautés. Dans cette thèse, nous présentons une approche qui consiste à former des communautés chevauchantes mixtes significatives dans des réseaux bipartis. Cette approche est fondée sur une double optimisation de la modularité réseau. À ce titre, nous proposons deux algorithmes. Le premier est un algorithme évolutionnaire dédié à une optimisation globale de la modularité de Newman sur le graphe-arête. Le second algorithme est appliqué sur la décomposition de nœuds résultant du processus évolutionnaire. Ce deuxième algorithme est destiné à optimiser localement la modularité de Mancoridis en prenant en considération l'aspect sémantique. La prise en compte de la sémantique contribue à détecter des communautés cohérentes et significatives. Notre approche ne nécessite aucune connaissance préalable du nombre de communautés recherchées dans le réseau. Nous validons notre approche sur deux ensembles de données : un groupe de réseaux synthétiques et un réseau du monde réel.

Mots clés: Data mining, algorithme évolutionnaire, détection de communautés chevauchantes, optimisation globale et locale de la modularité, réseaux bipartis.

Abstract

The modern science of networks has brought significant advances to our understanding of complex systems. One of the most relevant features of networks representing real systems is the existence of areas more densely connected than other areas. These areas are usually called communities. Many algorithms have been designed to discover community structure in networks. These algorithms are mostly dedicated to detecting disjoint communities. Very few of them are intended to discover overlapping communities, particularly the bipartite networks, have hardly been explored for the detection of such communities. In this thesis, we describe a new approach which consists in forming significant overlapping mixed communities in a bipartite network based on dual optimization of modularity. To this end, we propose two algorithms. The first one is an evolutionary algorithm dedicated for global optimization of the Newman's modularity on the line graph. The second one is an algorithm that locally optimizes the graph Mancoridis modularity, we have adapted to a bipartite graph. Specifically, this second algorithm is applied to the decomposition of nodes, resulting from the evolutionary process, and also characterizes the overlapping communities taking into account their semantic aspect. Taking into account the semantic aspect contributes to detect consistent and significant communities. Our approach requires a priori no knowledge about the number of communities searched in the network. We show its interest on two datasets, namely, a group of synthetic networks and real-world network.

Keywords: Data mining, evolutionary algorithm, detection of overlapping communities, global and local modularity optimization, bipartite networks.

Table des matières

Chapitre 3

Chapitre 4

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction générale

Depuis une vingtaine d'années, nous assistons à une explosion d'activités autour de la représentation, de la modélisation et de l'analyse de grands volumes de données d'interactions. Ces données ne touchent pas seulement un domaine particulier mais s'étendent à tout système pouvant être décrit par un réseau au sens mathématique (graphe). Par ailleurs, l'utilisation généralisée des technologies de l'information et de la communication, la progression des moyens de recueil et de stockage des données rend la taille de ces réseaux de plus en plus croissante. Néanmoins, les performances des ordinateurs actuels permettent la manipulation de volumes d'informations de tailles dépassant de très loin celles des volumes d'informations d'il y a seulement une décennie. Ainsi, le développement de méthodes permettant la représentation, l'analyse et la compréhension de ces réseaux est aujourd'hui considéré comme un domaine de recherche des plus actifs et en pleine effervescence, à la frontière entre plusieurs disciplines comme les mathématiques, les statistiques, l'informatique, etc.

Contexte de travail

Une des caractéristiques commune que l'on retrouve dans de nombreux réseaux réside dans l'existence de zones plus densément connectées que d'autres. Ces zones sont habituellement appelées communautés et correspondent intuitivement à des groupes de nœuds plus fortement connectés entre eux qu'avec les autres nœuds du réseau. L'identification de ce type de structure est intéressante à plus d'un titre. En effet, cette structure existe dans de nombreux réseaux réels, et la plupart du temps, elle a une signification concrète en termes d'organisation. Ces communautés permettent de ce fait de donner un point de vue macroscopique sur la structure des réseaux. Elles peuvent avoir des interprétations différentes suivant le type de réseau considéré. Ainsi, elles correspondent, dans les réseaux sociaux à des ensembles d'individus ayant des points communs et dont les liens sociaux sont certainement plus forts, par exemple, des groupes d'individus avec des intérêts communs, des activités communes, etc. De manière totalement différente, dans les réseaux

biologiques d'interactions protéine-protéine, les communautés correspondent communément à des ensembles de protéines qui collaborent à une même fonction cellulaire. Les propriétés topologiques des réseaux d'interactions protéine-protéine sont les clés de la compréhension des maladies et devraient permettre aux biologistes de trouver des objectifs thérapeutiques précis.
Selon l'usage que l'on veut en faire, les communautés peuvent être disjointes ou non. En biologie par exemple, l'analyse des réseaux d'interactions protéine-protéine permet de prédire les fonctions des protéines. Ainsi, de nombreuses protéines peuvent avoir plusieurs fonctions. Dans ce cas, il est logique de rechercher non pas une partition, mais un *recouvrement*, c'est-à-dire un système *de communautés chevauchantes*. Il en est de même dans les réseaux sociaux, où les individus peuvent dépendre de plusieurs groupes. Les travaux de cette thèse s'inscrivent dans ce contexte interdisciplinaire, en se concentrant sur la question algorithmique de détection de communautés. Plus précisément, c'est dans la détection de communautés chevauchantes, que se situe notre travail. Nous nous sommes intéressés principalement aux réseaux bipartis.

Problématique

La détection de zones dites communautaires est un domaine de recherche proche des problématiques classiques de clustering de données en fouille de données [Sou et al. 13, Ait et al. 12] et/ou de partitionnement de graphes en mathématiques. Ainsi, la détection de communautés est un problème NP-difficile. De ce fait, l'optimum est impossible à calculer en un temps raisonnable dès lors que les graphes sont de grande taille. On fait donc appel à des méthodes d'optimisation approchées conduisant, en temps et espace polynomiaux, à des partitions que l'on espère proches de l'optimum. Malgré les efforts déployés dans ce sens, comme en témoigne le rapport de Fortunato [Fortuna 10], le problème de détection de communautés est non encore résolu de manière satisfaisante.

Un réseau biparti, par opposition au réseau monoparti, est une catégorie spéciale de réseau, où les nœuds peuvent être divisés en deux sous-ensembles disjoints, tels qu'il ne peut y avoir de nœuds joints dans le même sous-ensemble. Les réseaux auteur-publication, les réseaux acteur-film, les réseaux produit-consommation, etc., sont des exemples concrets dans la catégorie des réseaux bipartis.

La détection de communautés dans des réseaux bipartis donne un aperçu sur l'interaction entre les entités correspondantes. Récemment, certains auteurs ont abordé la détection de communautés dans des réseaux bipartis. Une première problématique dans de tels réseaux réside dans la définition de la notion de communauté. En effet, ce concept n'a pas été abordé de la même manière. En conséquence, certains auteurs considèrent une communauté d'un réseau biparti comme composée de nœuds de même type. Par exemple dans un réseau acteur-événement, les acteurs forment les communautés acteur et les événements forment les communautés événement. Dans ce cas on cherche à détecter uniquement des communautés de nœuds d'un même type à la fois. En revanche, d'autres auteurs considèrent une communauté de réseaux bipartis comme un groupe de nœuds mixtes densément connectés, qui est le point de vue traditionnel dans un réseau monoparti.

Dans les réseaux bipartis, la détection de communautés chevauchantes n'a pas reçu beaucoup d'attention. Généralement, la recherche de telles communautés se fait d'abord en projetant le réseau sur chaque mode, ensuite on essaie de localiser les communautés dans chacune des projections. Ceci est souvent moins efficace que la détection des communautés dans le réseau d'origine biparti, car une partie de l'information est perdue dans la projection. Un autre désavantage vient du fait qu'ordinairement les communautés détectées dans chacune des projections ne sont pas cohérentes entre elles, ce qui constitue une problématique qui n'a pas encore été abordée. Nous nous proposons dans ce contexte d'explorer cette piste, selon une approche permettant de détecter des communautés mixtes chevauchantes significatives dans des réseaux bipartis non orientés et non pondérés. L'interprétation de l'organisation des communautés dans les réseaux est une problématique dans le domaine de l'analyse de ces structures. Ainsi, notre approche peut aider à mieux comprendre la structure communautaire découverte.

Principales contributions

Bien que l'optimisation ne soit pas l'objectif principal dans cette thèse, elle peut être néanmoins considérée comme une problématique engendrée par la tâche de détection de communautés. Rappelons que la détection de communautés qui est l'objectif principal dans cette thèse est un problème NP-difficile. Dans cette optique, nous proposons une approche d'optimisation hybride pour la détection de communautés chevauchantes dans des réseaux bipartis. Cette approche exploite la

capacité d'exploration d'un algorithme d'optimisation globale et l'efficacité d'intensification d'un algorithme d'optimisation locale. Ces deux algorithmes sont complémentaires et agissent successivement l'un après l'autre à des phases différentes du processus d'optimisation. L'algorithme d'optimisation globale permet de faire alors "disparaître" la combinatoire du problème, laissant ainsi le champ libre à l'algorithme d'optimisation locale. Plus précisément, nous avons proposé d'appliquer après un algorithme évolutionnaire, un algorithme d'optimisation locale sur le meilleur élément trouvé de façon à approcher au mieux la valeur exacte de l'optimum.
Plus précisément, le problème de détection de communautés chevauchantes dans un réseau biparti est transformé dans cette approche en deux sous problèmes :

- Détection de communautés disjointes dans le graphe-arête (line-graph) du réseau biparti. Le graphe-arête du réseau biparti est en fait un graphe monoparti.
- Détermination des communautés chevauchantes à partir du recouvrement des groupes de nœuds obtenus précédemment.

Notre première contribution consiste en la proposition d'un algorithme d'optimisation globale dédié à la détection de communautés disjointes dans un réseau monoparti. Il s'agit d'un algorithme génétique, pour lequel nous avons conçu deux nouveaux opérateurs génétiques adaptés au problème de détection de communautés. Ces opérateurs sont spécialement crées pour faire varier le nombre de communautés, qui est un paramètre inconnu dans la détection. Cet algorithme sert précisément à partitionner le graphe-arête du réseau biparti. Ainsi, la structure communautaire trouvée dans le graphe-arête définit une décomposition de nœuds (des groupes chevauchants de nœuds) du graphe biparti.

La deuxième contribution dans cette thèse est la proposition d'un algorithme d'optimisation locale. Il est utilisé pour améliorer le score de qualité, associé à la décomposition de nœuds obtenue précédemment. Une certaine sémantique y est aussi intégrée pour donner un sens aux communautés recherchées. Cette sémantique est introduite sous forme d'une équation qui exprime l'appartenance des nœuds chevauchants à une certaine communauté. Ainsi, cet algorithme permet d'améliorer la décomposition des nœuds en vue d'avoir des communautés mixtes chevauchantes significatives dans des réseaux bipartis.

Notre dernière contribution consiste en l'adaptation d'une mesure de qualité. Celle-ci est, à l'origine utilisée pour quantifier un découpage en communautés

chevauchantes dans un réseau monoparti. Nous l'avons adaptée pour quantifier des communautés chevauchantes dans un réseau biparti donné.

Pour valider notre approche et vérifier la faisabilité de ces propositions, nous avons conduit nos expérimentations sur deux fronts.

- Dans le premier, nos tests consistent d'abord à montrer la performance de l'algorithme génétique à trouver des communautés disjointes sur deux ensembles de données, à savoir, plusieurs réseaux réels connus et un groupe de réseaux synthétiques. Ensuite, cet algorithme a été comparé aux principaux algorithmes génétiques existants pour la détection de communautés disjointes.
- Dans le second, nos tests visent à montrer l'efficacité de l'approche proposée à découvrir des communautés chevauchantes mixtes significatives dans un réseau biparti. À cet effet, nous avons également utilisé un groupe de réseaux synthétiques et un réseau réel dont la structure est connue pour être difficile à comprendre.

Organisation du document

Afin de présenter les travaux de cette thèse et le domaine dans lequel elle s'inscrit, nous avons retenu une organisation qui s'articule autour de cinq chapitres, d'une introduction générale et d'une conclusion.

Dans l'*introduction générale*, nous avons délimité le contexte de notre étude et identifié les différentes problématiques associées. Nous avons dans ce contexte tracé quelques brins de pistes qui donnent une idée sur l'objectif principal que nous nous sommes assigné. Nous avons au passage donné un aperçu général sur nos contributions dont la démarche est détaillée au chapitre 4. Nous avons également tracé les grandes lignes de la mise en œuvre et des expérimentations réalisées pour valider notre approche. Enfin, nous terminons par le plan du rapport de thèse.

Nous abordons dans le premier chapitre "*les Algorithmes Évolutionnaires*" en premier lieu, le problème d'optimisation et nous résumons à l'occasion les méthodes de résolution de tels problèmes. Nous décrivons aussi les fondements biologiques qui ont inspiré les algorithmes évolutionnaires. Mais avant de décrire le modèle concernant ces algorithmes, nous donnons également certaines définitions utiles. Nous donnons un aperçu sur le fonctionnement de ces algorithmes en s'appuyant sur la mise en œuvre des algorithmes génétiques, vu leur utilisation intensive dans le cadre de notre approche. Outre les algorithmes génétiques, nous nous intéressons également

aux autres voies des algorithmes évolutionnaire: la programmation génétique, les stratégies d'évolution, la programmation évolutionnaire. Enfin, nous présentons quelques approches hybrides utilisant les algorithmes génétiques qui constituent aujourd'hui une nouvelle voie de recherche.

Le chapitre deux "*Notions sur la détection de communautés dans les réseaux*" a pour objectif principal de donner un aperçu général sur le domaine de la détection de communautés dans les réseaux. Il rapporte en premier lieu des cas de modélisation par des graphes, ainsi que certains concepts et définitions utiles de la théorie des graphes. Ensuite, il présente les diverses définitions relatives à la notion de communauté. Enfin, il termine par la définition formelle et l'intérêt de la détection de communautés dans les réseaux.

Le troisième chapitre intitulé "*Détection de communautés : un état des lieux"*, vise à présenter un état de l'art sur la détection de communautés. Pour donner une vue d'ensemble des méthodes proposées, et illustrer leur diversité, nous nous sommes limités aux approches qui ont reçu le plus d'attention de la part de la communauté scientifique. Ainsi, pour une meilleure présentation, nous les avons rassemblées en fonction du type de communautés recherchées (disjointes ou chevauchantes), du type de réseaux pour lesquels elles sont conçues (monopartis ou bipartis) et du type d'approche utilisée.

Quant au quatrième chapitre "*Double optimisation de la modularité pour détection de communautés chevauchantes*", il est entièrement consacré à la description de l'approche que nous proposons dans cette thèse et qui a fait l'objet d'une publication dans la revue "Knowledge and Information Systems", éditée par Springer-Verlag. Nous présentons tout d'abord le schéma général de l'approche de détection de communautés mixtes chevauchantes dans des réseaux bipartis, qui constitue notre principale proposition. Nous détaillons par la suite les quatre parties constituant notre approche, en commençant par l'algorithme évolutionnaire que nous avons mis en œuvre pour la détection de communautés disjointes. Nous décrivons également le passage d'un graphe biparti au graphe-arête correspondant, et nous détaillons notre algorithme d'optimisation locale pour obtenir des communautés chevauchantes mixtes significatives. Nous montrons, au passage, l'adaptation que nous avons réalisée sur la fonction de qualité pour des réseaux bipartis.

Le cinquième chapitre "*Expérimentation*" est totalement dédié à l'expérimentation des différentes propositions concernant notre apport dans cette thèse. Nos premiers tests montrent la performance de l'algorithme évolutionnaire à trouver des communautés disjointes. Ces tests sont effectués sur deux ensembles de données, à savoir, plusieurs réseaux réels et un groupe de réseaux synthétiques. D'autres tests situant notre algorithme évolutionnaire par rapport aux algorithmes existants sont proposés. La seconde série de tests vise à montrer l'efficacité de l'approche proposée à découvrir des communautés chevauchantes mixtes dans des réseaux bipartis. Ces derniers tests sont également utilisés sur un groupe de réseaux synthétiques et un réseau réel.

Enfin, une dernière partie "*Conclusion et perspectives*" récapitule nos contributions et propose des perspectives de recherche future.

1

Les Algorithmes Évolutionnaires

1.1 Introduction

Les algorithmes évolutionnaires (AEs) sont des méthodes stochastiques inspirées du modèle biologiste sélectionniste introduit par Charles Darwin. Ils sont essentiellement basés sur le principe du processus d'évolution naturelle [DeJ Spe 93, Bâc Sch 93, Sch Mic 97]. Ces algorithmes se sont vite imposés comme méthodes d'optimisation globale, permettant de trouver des solutions approchées pour des problèmes très variés et non triviaux, particulièrement dans le traitement des problèmes de grande taille.

Dans ce chapitre, nous commençons d'abord par rappeler brièvement certains concepts et notions relatifs au domaine de l'optimisation combinatoire. Nous présentons ensuite les fondements biologiques qui ont inspiré les algorithmes évolutionnaires. Mais, avant de décrire le modèle évolutionnaire, nous introduisons certaines définitions utiles. Nous donnons un aperçu sur le fonctionnement de ces algorithmes en s'appuyant sur la mise en œuvre des algorithmes génétiques étant donné leur utilisation intensive dans le cadre de notre approche (cf. chapitre 4). Enfin, nous décrivons succinctement les autres algorithmes évolutionnaires : la programmation génétique, les stratégies d'évolution, la programmation évolutionnaire ainsi que différentes approches hybrides. Nous clôturons ce chapitre par une conclusion.

1.2 Optimisation combinatoire : définition et résumé des méthodes

Les scientifiques ont toujours été confrontés à des problèmes de complexité croissante qui surgissent dans des secteurs techniques très divers. Ces problèmes peuvent être formulés sous forme de problème d'optimisation combinatoire [Rib Mac 94]. Bien que les problèmes d'optimisation combinatoire soient habituellement faciles à décrire, leur résolution n'est pas toujours aisée. D'ailleurs, la majorité de ces problèmes sont classés comme étant des problèmes NP-difficiles. Cependant, aucun algorithme polynomial de résolution n'a été trouvé pour de tels problèmes. De ce fait, l'utilisation de méthodes d'optimisation permettant de trouver une solution acceptable en un temps raisonnable, est généralement la mieux indiquée.

1.2.1 Définitions

Problème d'optimisation

Un problème d'optimisation est défini communément par un ensemble de solutions possibles S, dont la qualité peut être décrite par une fonction objectif f dite aussi fonction de qualité. On cherche alors à trouver la solution $s*$ présentant la meilleure qualité $f(s*)$.

Optimisation combinatoire

L'optimisation combinatoire est une discipline utilisant conjointement différentes techniques des mathématiques discrètes, de la recherche opérationnelle et de l'informatique, afin de résoudre des problèmes d'optimisation dont la structure sous-jacente est discrète (généralement un graphe).

Optimum global, optimum local

Soit un problème d'optimisation combinatoire, une fonction objectif f et l'ensemble des solutions admissibles du problème S_a.

Soit $\check{x} \in S_a$, si l'on peut prouver que $\forall\ x \in S_a\ \ f(\check{x}) \leq f(x)$, alors on dira que $\check{x}$ est l'optimum (minimum) global du problème.

S'il existe un ensemble $V \subset S_a$, contenant $\check{x}$ et au moins deux autres éléments, tel que $\forall x \in V, f(\check{x}) \leq f(x)$, on dira alors que $\check{x}$ est un optimum (minimum) local du problème.

1.2.2 Résumé des méthodes de résolution existantes

Un très grand nombre de méthodes de résolution existent pour l'optimisation combinatoire. Ces méthodes se déclinent en deux variantes : les méthodes exactes (complètes) qui assurent la complétude de la résolution et les méthodes approchées (incomplètes) qui perdent en complétude mais gagnent en efficacité.

1. Le principe essentiel d'une méthode *exacte* consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Les méthodes exactes sont alors susceptibles de trouver des solutions optimales pour des problèmes de taille raisonnable.

2. Le but des méthodes *approchées* est de trouver une solution de bonne qualité en un temps raisonnable sans garantir l'optimalité de la solution obtenue. Les méthodes approchées sont basées principalement, sur diverses *heuristiques*, souvent spécifiques à un type de problème et sur une classe de méthode dites *méta-heuristiques*. Les *méta-heuristiques*, interviennent dans toutes les situations où l'on ne connaît pas d'heuristique efficace pour résoudre un problème donné. Elles ont en commun les particularités suivantes :

   - Elles introduisent en général un principe stochastique qui permet de faire face à l'explosion combinatoire des possibilités ;
   - Elles sont inspirées par des similitudes avec la physique comme le recuit simulé, avec la biologie comme les algorithmes évolutionnaires et avec l'éthologie[1] comme les colonies de fourmis ;
   - Elles sont en mesure de guider, dans une tâche précise, une autre méthode de recherche spécialisée (par exemple, une autre heuristique, ou une méthode d'exploration locale).

Les méta-heuristiques n'étant pas, à priori, spécifiques à la résolution de tel ou tel type de problème, leur classification reste assez arbitraire. On peut cependant distinguer deux familles [Dré et al. 03]: les méta-heuristiques de voisinage et celles dites à population.

1. Les méta-heuristiques de "*voisinage*" sont les plus classiques. Elles sont fondées sur la notion de parcours où l'algorithme part d'une solution initiale et s'en

[1] *L'éthologie* est l'étude du comportement des diverses espèces animales.

éloigne progressivement, pour réaliser une trajectoire ou un parcours progressif dans l'espace des solutions. Dans cette catégorie, se rangent :

- la méthode de descente,
- le recuit simulé[2],
- la méthode Tabou,
- la recherche par voisinage variable.

2. Les méta-heuristiques à "*population*", consistent à travailler conjointement avec un ensemble de solutions, que l'on fait évoluer graduellement. L'emploi de plusieurs solutions simultanément permet forcément d'améliorer l'exploration de l'espace des configurations. Dans cette seconde catégorie, on recense :
   - les *algorithmes évolutionnaires*,
   - les algorithmes par colonies de fourmis,
   - l'optimisation par essaim particulaire,
   - les algorithmes à estimation de distribution, ...

1.3 Introduction aux algorithmes évolutionnaires

La biologie est la source d'inspiration de nombreuses méta-heuristiques. Aussi, les *théories de l'évolution* ont inspiré les algorithmes évolutionnaires [Goldber 94], les phénomènes de suivi de *piste chez les fourmis* ont conduit à l'élaboration des algorithmes de colonies de fourmis [Bon et al. 99], l'étude de *l'organisation de groupes d'animaux* a donné naissance aux méthodes d'optimisation par essaim particulaire [Ebe et al. 01]. Il existe, par ailleurs, d'autres algorithmes, moins connus que ceux que nous venons d'évoquer, qui dérivent de la biologie. Il s'agit en particulier, des algorithmes inspirés du fonctionnement du *système immunitaire* [DeC Von 99], des algorithmes pour l'allocation dynamique de tâches [Cic Smi 01] se basant sur des modèles d'organisation du travail chez les fourmis, et enfin des algorithmes de classification inspirés des essaims d'insectes [Aup et al. 03].

1.3.1 Hérédité

On appelle hérédité la capacité qu'a un être vivant à transmettre certaines de ses caractéristiques à ses descendants. De tout temps, l'homme a su utiliser les principes

[2] Le *recuit simulé* est une méta-heuristique inspirée d'un processus utilisé en métallurgie

de l'hérédité sans vraiment en comprendre les mécanismes fondamentaux. Tous les éleveurs savent bien qu'il vaut mieux choisir les meilleurs animaux comme reproducteurs. Bien que la transmission des caractères d'un individu à ses descendants semble évidente, la disparition de certains caractères et la réapparition d'autres sont longtemps restées inexpliquées. La découverte des chromosomes, véritable support matériel des caractères héréditaires, puis celle de la structure moléculaire de l'acide désoxyribonucléique (ADN) ont permis de comprendre le fonctionnement exact des principes héréditaires. Les chromosomes sont constitués d'ADN, entité qui contient toute l'information nécessaire à la "fabrication" d'un individu [Madelin 02].

1.3.2 Évolution et sélection naturelle de Charles Darwin

Dans son livre, "The Origin of Species" en (1859), le naturaliste anglais Charles Darwin (1809-1882) met l'accent sur la lutte pour la survie et les mécanismes d'adaptation des êtres vivants, qui leur permettent de générer des espèces différentes. La multiplication des individus se faisant à une cadence supérieure à celle de leurs ressources, elle conduit à une impitoyable compétition vitale, entrainant la mort sélective des plus *faibles* et la survie des plus *aptes*. Mais cette sélection des individus les plus adaptés ne semble pas être le seul critère. Comment expliquer sinon l'évolution qu'a suivie par exemple un paon ?[3]. D'autres points posent aussi problème. En effet, une modification étant toujours un petit changement, généralement insuffisant pour donner un véritable avantage lors de la sélection naturelle. Le caractère ne devient généralement profitable qu'après une longue orthogénèse[4]. En fait, la nature semble éloigner les pires individus, mais laisse le soin au *hasard* en ce qui concerne tous les autres. Ce mécanisme permet deux choses. Premièrement, il y a conservation de la diversité génétique qui permet, entre autres d'engendrer de nouvelles variations et d'éviter l'expression de mutation fatale. Deuxièmement, il permet à la population d'avoir une meilleure capacité d'adaptation en cas de variation du milieu.

Le principe de la sélection naturelle est le suivant : puisque tous les individus sont différents et que certains, mieux adaptés, vont plus procréer, les caractéristiques avantageuses de ces derniers seront héritées par les générations suivantes, et avec le temps, deviendront dominantes dans la population. Cette théorie de la sélection

[3] Dont la queue en panache est un véritable appel pour les prédateurs.
[4] Processus évolutif au cours duquel un caractère est modifié par étapes dans le même sens.

naturelle s'applique aux individus d'une population et repose donc sur trois principes :

- *la variation* : les individus diffèrent les uns des autres ;
- *l'adaptation* : les individus les mieux adaptés à leur environnement vivent plus longtemps et procréent plus ;
- *l'hérédité* : les caractéristiques des individus sont héréditaires.

1.3.3 Du modèle biologique à l'informatique

L'idée d'appliquer les principes du darwinisme pour des implantations informatiques a été introduite par John Holland [Holland 75]. Puisque sur des millions d'années les êtres vivants, par le principe de sélection naturelle, se sont spécialisés et complexifiés afin d'être de plus en plus adaptés à leur environnement, on pouvait légitimement penser que l'application de ces principes pour l'optimisation de systèmes complexes s'avérerait fructueuse. Mais ce passage d'un principe biologique des plus complexes vers un modèle forcément réducteur ne s'est pas fait sans pertes. Les seuls principes retenus lors de ce passage et sur lesquels se basent les algorithmes évolutionnaires sont la *sélection*, le *croisement* et la *mutation*.

L'évolution naturelle peut donc être vue comme un processus d'*optimisation* où les individus essayent de maximiser le profit qu'ils peuvent obtenir des conditions naturelles environnantes afin d'assurer leur descendance. Dans l'évolution artificielle, les individus ne sont plus des êtres vivants, mais des solutions possibles d'un problème, les contraintes de l'environnement sont exprimées au moyen d'une fonction à satisfaire ou à maximiser, et les solutions sont modifiées avec des opérateurs qui simulent la mutation et la combinaison des individus. Les meilleures solutions ont plus de chance de se reproduire, dans un cadre d'amélioration continue.

1.3.4 Principe de fonctionnement

Le principe d'un algorithme évolutionnaire (AE) se décrit simplement (cf. figure 1.1). Considérant un problème d'optimisation combinatoire, une population est un ensemble de points de l'espace des solutions, chacun de ces points est appelé individu. Un individu est constitué d'un patrimoine génétique qui le caractérise et le différencie des autres individus; concrètement les gènes sont les blocs élémentaires caractérisant une solution. Habituellement, un individu est pris comme une liste d'entiers s'il s'agit de problèmes combinatoires, un vecteur de nombres réels

pour des problèmes numériques dans des espaces continus ou une chaîne de nombres binaires pour des problèmes booléens. Au besoin, ces représentations peuvent être combinées dans des structures complexes.

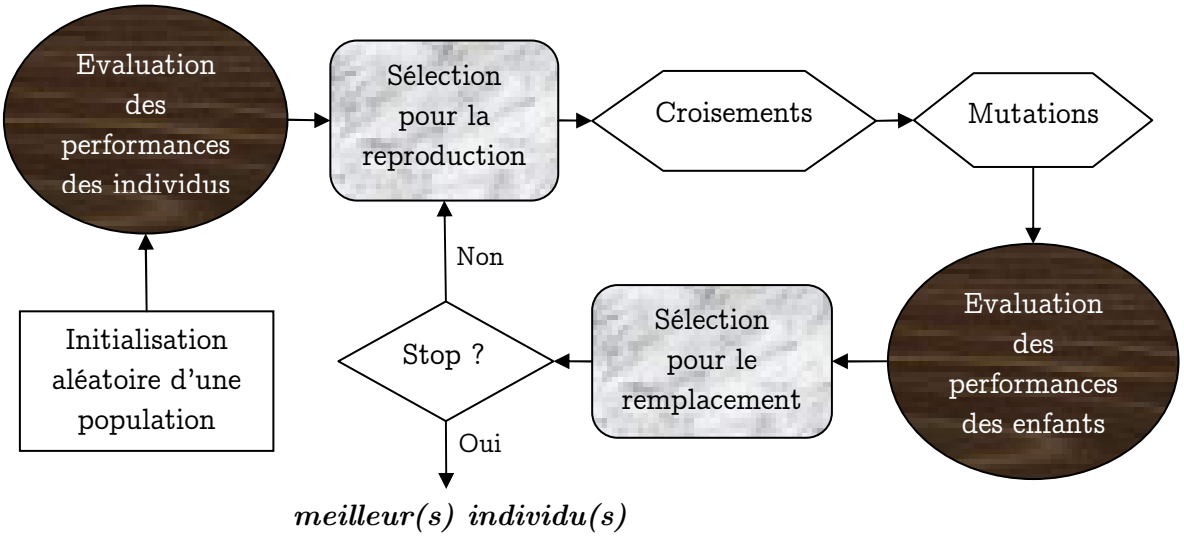


Fig 1.1. Principe d'un algorithme évolutionnaire standard (extrait du livre [Dré et al. 03])

La fonction *objectif* évalue l'adaptation d'un individu à son environnement. A ce propos, elle est appelée fonction d'adaptation ou *fitness*. Les AEs ont pour but de maximiser la fonction d'adaptation. Dans le cas d'une fonction objectif à minimiser, en supposant qu'elle est positive, l'inverse de cette fonction peut être utilisé comme fonction d'adaptation.

Le terme "algorithmes évolutionnaires" réunit en fait un ensemble de techniques : *les algorithmes génétiques* sont certainement l'approche la plus connue, notamment grâce à l'ouvrage de référence de David Goldberg [Goldber 89] : "Genetic Algorithms in Search, Optimization and Machine Learning" (voir dans [Goldber 94] la traduction française), *les stratégies d'évolution* (Evolution strategies [Rechen 73b]), *la programmation évolutionnaire* (Evolutionary Programming [Fog et al. 66]) et *la programmation génétique* (Genetic Programming [Koza 92]).

Les AEs ont été appliqués dans pratiquement tous les domaines où l'on retrouve des problèmes d'optimisation comme dans les travaux de [Mic Fog 98, Pol et al. 08]. Etant donné leur généralité, les AEs sont plutôt un cadre algorithmique pour la résolution des problèmes. On peut trouver des applications dans des domaines d'optimisation variés comme la chaîne logistique et le commerce [Jaw Bal 09], le transport [Chu et al. 09], la production [Rom Slo 09], la production d'énergie

[Zio et al. 09], l'investissement [Sol et al. 09], l'hydrologie [Che Cha 09], la biologie cellulaire [Moh et al. 08], la génétique [Kim et al. 08], la médecine [Smi Tim 08], la pharmacologie [Ter Gas 01], la recherche documentaire [Sil et al. 09] ou les jeux [Ber et al. 09].

Nous nous sommes intéressés dans la section suivante aux algorithmes génétiques étant donné leur utilisation dans le cadre de l'approche proposée dans cette thèse. Des explications concernant les autres techniques des algorithmes évolutionnaires sont données à la section 1.6.

1.4 Les algorithmes génétiques

Les algorithmes génétiques (AGs) font partie de la classe des algorithmes dits stochastiques. En effet, une grande partie de leur fonctionnement est basée sur le hasard. Cependant, ce hasard est dirigé grâce à la fonction d'évaluation qui permet d'introduire dans les opérateurs génétiques une quantité de déterminisme utile afin d'obtenir une solution. Ces algorithmes respectent le schéma d'un AE avec une particularité : lors des étapes d'évaluation, une solution du problème est entièrement construite à partir du génotype d'un unique individu (celui-ci est souvent une chaine de symboles binaires). Les AGs puisent une grande partie de leur terminologie de la biologie, et pour une bonne compréhension du bon nombre de ces analogies, nous donnons ici un certain nombre de définitions. Nous expliquons ensuite le fonctionnement d'un AG standard.

1.4.1 Vocabulaire

Définition 1.1 (Gène). Un gène est une unité d'information génétique. Dans les AGs, on appellera gène la suite de symboles qui codent la valeur d'une variable. Dans le cas général, un gène correspond à un seul symbole (0 ou 1 dans le cas binaire).

Définition 1.2 (Génotype). Correspond à l'ensemble des valeurs des gènes.

Définition 1.3 (Individu). Il représente le codage d'une solution potentielle (un élément de l'espace de recherche).

Définition 1.4 (Chromosome). Un chromosome est constitué de gènes, l'ensemble des chromosomes d'un individu regroupe l'intégralité de son patrimoine génétique. Généralement un AG utilise un chromosome par individu.

Définition 1.5 (Performance). Elle représente la mesure de la qualité (*fitness*) des individus basée sur l'objectif de l'optimisation et permettant de comparer les individus entre eux afin d'en déterminer plus et moins aptes.

Définition 1.6 (Evaluation d'un individu). Elle représente le calcul de la performance de l'individu.

Définition 1.7 (Population). Un ensemble fini (de taille N) d'individus.

Définition 1.8 (Evolution). Représente un processus d'optimisation itératif de recherche d'un (ou plusieurs) individu(s).

Définition 1.9 (Génération). Correspond à l'itération, mais ce terme signifie parfois la population en une certaine itération.

Définition 1.10 (Croisement). Il s'agit d'un opérateur de reproduction (*Cross over*) appliqué avec la probabilité p_c et qui correspond à un brassage d'information entre les individus de la population. Il consiste à échanger des parties composantes (gènes) entre deux ou plusieurs individus.

Définition 1.11 (Mutation). Opérateur de modification d'un ou plusieurs gènes appliqué avec la probabilité p_m dans le but d'introduire une nouvelle variabilité dans la population.

Définition 1.12 (Sélection). Processus du choix des individus pour la reproduction basé sur leur performance.

Définition 1.13 (Remplacement). Il représente le processus de formation d'une nouvelle population à partir des ensembles de parents et d'enfants, effectué le plus souvent sur la base de leur performance.

1.4.2 Fonctionnement

A- Principe

Le schéma global d'un AG est constitué de 6 éléments principaux :

1. Une population initiale de configurations ;
2. Une fonction de codage/décodage des chromosomes ;
3. Des opérateurs génétiques (Mutation, Croisement, ...) ;
4. Une fonction d'évaluation ;
5. Un algorithme de sélection (Tournoi, Roulette, ...) ;
6. Des paramètres.

Enfin, il suffit d'appliquer l'algorithme 1.1, pour lequel nous avons les notations suivantes :

- POP_t : population à la génération t ,
- POP_{sel} : population sélectionnée.

Algorithme 1.1 Algorithme Génétique Standard (SGA : Standard Genetic Algorithm)

Début
  $t = 0$;
  Initialisation de la population POP_t ;
  Evaluation des individus de la population POP_t ;
  Tant que (Condition de terminaison non satisfaite) **faire**
    $t \leftarrow t + 1$;
    Sélection et copies des parents de la population POP_{t-1} dans POP_{sel} ;
    Application des opérateurs génétiques sur POP_{sel} ;
    $POP_t = POP_{sel}$;
    Evaluation des individus de la population POP_t ;
  Fin Tantque
  Donner la meilleure solution ;
Fin

B- Codage

Il existe essentiellement deux types de codage : le codage binaire ou représentation sous forme de chaîne binaire et le codage réel qui est une représentation directe des valeurs réelles de la variable. Nous pouvons naturellement passer d'un codage à l'autre. Le codage initialement utilisé par John Holland dans l'AG standard est le codage binaire. Nous allons par conséquent d'abord présenter ce cas.

Définition 1.14. (Codage binaire). On appelle "séquence ou chaîne" de longueur $l\,(A)$ une séquence avec $A = \{a_1, a_2, \ldots, a_l\}$ avec $\forall\ i \in \{1, \ldots l\},\ a_i \in V = \{0, 1\}$.

Chaque chromosome est donc constitué d'une chaîne de n bits. Les opérateurs génétiques vont donc effectuer des réorganisations de ces chaînes binaires en vue d'améliorer l'évaluation des chromosomes.

Définition 1.15. (Codage réel). Dans le cas d'un codage non binaire, tel que le codage réel, la séquence A ne contient qu'un point, nous avons $A = \{a\}$ avec $a \in \mathrm{R}$.

Nous expliquons dans ce qui suit chaque étape de l'algorithme génétique standard présenté précédemment dans l'algorithme 1.1.

C- Initialisation de la population

Généralement, la population initiale est choisie aléatoirement, chacun des bits de chaque chromosome étant choisi au hasard. Cependant, rien n'interdit d'utiliser des initialisations gloutonnes, de fournir des solutions déjà connues que l'on désire améliorer. Ainsi on peut utiliser d'autres algorithmes de recherche qui fourniront à l'AG des solutions ayant subi une première phase d'optimisation. Le choix de l'initialisation se fera en fonction des connaissances que l'utilisateur a du problème. S'il n'a pas d'informations particulières, on préférera une initialisation aléatoire, la plus uniforme possible, afin de favoriser une exploration maximale de l'espace de recherche.

D- Évaluation

L'opérateur d'évaluation est un opérateur très dépendant du problème à traiter. Il est généralement défini par type de problème, voire par problème. Cet opérateur est loin d'être banal. Il sert à l'opérateur de sélection pour faire son choix des individus à conserver. Il agira donc sur la convergence de l'algorithme. Dans certains cas, on pourra distinguer la fonction d'évaluation de la fonction de *fitness* : la fonction d'évaluation calcule le coût d'un individu, alors que la fonction de *fitness* calcule le coût de cet individu par rapport à la population courante. Lorsqu'une sélection (voir le paragraphe f) de type roulette est adoptée, il est nécessaire de faire la distinction entre fonction de *fitness* et fonction d'évaluation. En effet, il est nécessaire de définir la fonction de *fitness* de telle sorte qu'elle ne favorise pas trop les meilleurs individus, mais aussi qu'elle permette de récupérer les individus incontestablement mauvais et inutiles pour la recherche. Par contre, lorsqu'une sélection par tournoi est utilisée, il n'est pas nécessaire de faire de mise à l'échelle[5], la fonction d'évaluation suffit donc largement. Nous avons adopté dans notre approche la sélection par tournoi, nous utiliserons donc indifféremment les termes *fitness* et évaluation.

[5] La mise à l'échelle est une technique de réajustement de l'espérance du nombre de copies d'un individu dans la population POP_{sel}.

E- Condition de terminaison

La condition de terminaison d'un AG diffère suivant le type de problème traité. Si on traite un problème de décision, c'est à dire que l'on connaît la valeur de l'optimum que l'on cherche à atteindre, c'est assez simple. En revanche pour des problèmes d'optimisation, on ne connaît pas l'optimum, et on ne sait jamais si celui-ci est atteint. Dans le cas général, on se restreindra au nombre de générations maximum à réaliser depuis le début ou depuis la dernière amélioration trouvée. On peut aussi, décider d'arrêter lorsqu'il y a convergence prématurée. On parle de convergence prématurée quand la diversité entre les individus est insignifiante, pour espérer sortir d'un bassin d'attraction.

F- Sélection

L'opérateur de sélection sert à choisir dans la population courante les individus qui auront le "droit" de survivre et de se reproduire. Plus clairement, cet opérateur va générer à partir de la population courante POP_{t-1}, une population POP_{sel} par copie des individus choisis dans POP_{t-1}. Les opérateurs génétiques seront ensuite appliqués sur la population POP_{sel}. Le nombre d'occurrences d'un individu x_i de la population POP_{t-1} dans la population POP_{sel} tend généralement à être proportionnel au rapport entre la *fitness* et la *fitness* moyenne. De fait, les individus les mieux adaptés ont plus de chance de figurer (parfois plusieurs fois) dans la population POP_{sel}, et les individus les moins adaptés ont peu de chance d'y figurer. Nous relatons ici les procédures de sélection les plus utilisées.

Sélection par roulette

Dans ce type de sélection, les chromosomes parents sont choisis en fonction de leur performance. Meilleur est le résultat calculé à partir du code du chromosome, plus grandes sont ses chances d'être sélectionné. La sélection par roulette peut être vue comme une sorte de roulette de casino sur laquelle sont mis tous les chromosomes de la population. La place accordée à chaque chromosome étant en relation avec sa valeur d'adaptation. Un exemple de roulette de sélection est représenté par la figure 1.2.

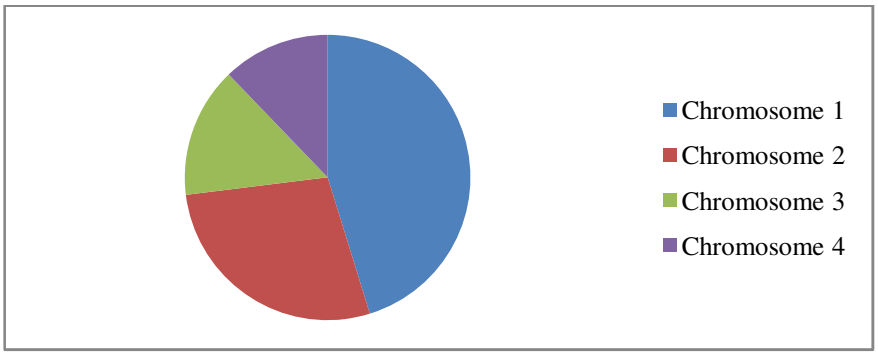


Fig 1.2. Schéma d'une roulette de sélection

Ensuite, la bille est lancée et s'arrête sur un chromosome. Ainsi, les meilleurs chromosomes peuvent avoir la chance d'être tirés plusieurs fois, et les plus mauvais ne jamais être sélectionnés. Cette sélection peut être simulée par l'algorithme suivant :

1. On calcule la somme *S1* de toutes les valeurs de la fonction d'évaluation d'une population.
2. On génère un nombre r entre 0 et *S1*.
3. On calcule ensuite une somme *S2* des évaluations en s'arrêtant dès que r est dépassé.
4. Le dernier chromosome pour lequel la fonction d'évaluation vient d'être ajoutée est sélectionné.

Sélection par rang

La sélection par roulette est confrontée à des problèmes quand la valeur d'adaptation des chromosomes varie considérablement. Si la meilleure fonction d'évaluation d'un chromosome représente un grand pourcentage de la roulette alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à un arrêt de l'évolution.

La sélection par rang effectue d'abord un tri sur la population par rapport à la *fitness*. Ensuite, à chaque chromosome est associé un rang en fonction de sa position. En conséquence, pour une population de N chromosomes, le plus mauvais aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N. La sélection par rang et la sélection par roulette sont presque identiques, sauf que, avec la sélection par rang les proportions sont en rapport avec le rang plutôt qu'avec la valeur de l'évaluation. Ainsi, les probabilités finales seront calculées avec la formule : Rang / $\sum$ (des Rangs). Le tableau 1.1 fournit un exemple de sélection par

rang. Nous voyons qu'avec cette méthode de sélection, les chromosomes ont tous une chance d'être sélectionnés. Néanmoins, elle aboutit à une convergence plus lente vers la bonne solution, car les meilleurs chromosomes ne sont pas très différents des plus mauvais.

Chromosomes	1	2	3	4	5	6	Total
Probabilités initiales	89 %	5 %	1 %	4 %	3 %	2 %	100 %
Rang	6	5	1	4	3	2	21
Probabilités finales	29 %	24 %	5 %	19 %	14 %	9 %	100 %

Tableau 1.1. Exemples de sélection par rang pour 6 chromosomes

Sélection par tournoi

On forme m paires de chromosomes dans une population de m chromosomes, Dans les paramètres de l'AG, on décide d'une probabilité de victoire du plus fort. Cette probabilité représente la chance que possède le meilleur chromosome de chaque paire d'être sélectionné. Cette probabilité doit être grande (entre 70% et 100%). A partir des m paires, on détermine ainsi m individus pour la reproduction.

Elitisme

Lors de la création d'une nouvelle population, il y a de fortes chances que les meilleurs chromosomes soient perdus après les opérations d'hybridation et de mutation. Pour échapper à cela, on utilise la méthode d'élitisme. Elle est réalisée par la recopie d'un ou de plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on produit le reste de la population selon l'algorithme de reproduction courant. Cette méthode améliore beaucoup les algorithmes génétiques, car elle permet de sauvegarder les meilleures solutions rencontrées.

G- Opérateurs de diversification

La diversification va permettre à l'algorithme d'explorer l'espace des solutions à la recherche du ou des optima globaux. Cette étape consiste à créer de nouvelles solutions (les enfants) à partir des solutions présentes au début de l'itération et sélectionnées à l'étape précédente (les parents). Les AGs se sont inspirés des méthodes de reproduction du vivant pour créer des opérateurs de diversification. Le croisement joue le rôle de la reproduction sexuée et la mutation celui de la reproduction asexuée.

Croisement (Crossover)

L'opérateur de croisement est généralement considéré comme opérateur d'exploitation. On entend par là qu'il va permettre de découvrir de meilleures solutions en combinant les avantages de solutions déjà découvertes. Dans la version classique d'un AG, on va choisir deux individus de la population POP_{sel} et leur appliquer l'opérateur de croisement ou bien les recopier dans la population POP_t tels quels, en fonction de la probabilité de croisement. Le croisement à un point va créer deux enfants à partir des deux parents de telle sorte que chacun des enfants ait une partie du chromosome de chaque parent. On va pour cela, choisir un entier i tel que $1 \leq i < n$, l'enfant 1 va recopier les gènes 1,...,i du parent 1 et les gènes i+1,...,n du parent 2, et réciproquement pour l'enfant 2 (cf. figure 1.3 a)). Une variation communément utilisée consiste à choisir deux points de croisement. Le croisement va s'effectuer comme sur la figure 1.3 b), la position du ou des points de croisement étant généralement choisie au hasard.

Il existe de très nombreuses variations de l'opérateur de croisement plus ou moins vouées à des problèmes particuliers. Dans [Esh et al. 89] et [Cerf 94], se trouve une comparaison des opérateurs de croisement les plus communs.

Fig 1.3. Illustration de l'opérateur de croisement

Mutation

Sans exploration, un AG converge vers un optimum local et ne peut pas s'en échapper. L'opérateur de mutation va permettre l'exploration de l'espace de recherche. En effet, les autres opérateurs ne permettent que des déplacements dans certaines zones de l'espace de recherche. En utilisant la mutation, on peut théoriquement atteindre n'importe quelle zone de l'espace de recherche. L'ergodicité de la mutation est d'ailleurs une condition utilisée pratiquement par tous les résultats théoriques de convergence des AGs, autrement dit, le fait que tout point de l'espace de recherche peut être atteint en un nombre fini de mutations. Dans le cas général,

cela consiste à effectuer une altération aléatoire d'un ou de plusieurs gènes du chromosome (cf. figure 1.4), en fonction du taux de mutation, généralement assez faible. De très nombreuses variations de l'opérateur de mutation ont été proposées dans la littérature, que ce soit en faisant varier le taux en fonction du temps comme le préconise Holland [Holland 75], ou encore en particularisant l'opérateur en fonction du problème.

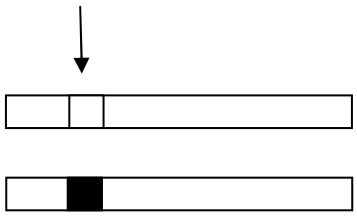


Fig 1.4. Illustration de l'opérateur de mutation

H- Paramétrage

Les AEs possèdent un nombre important de paramètres. Du réglage de ceux-ci, résultent la rapidité et la qualité de la convergence. Ces paramètres sont souvent difficiles à régler. Ils sont énumérés ci-après :

- la taille de la population, POP, est un paramètre important de l'optimisation. Plus elle sera grande, plus la probabilité de convergence de l'algorithme sera élevée. Cependant, plus il y aura d'individus, plus l'algorithme sera lent car la complexité de l'algorithme est directement proportionnelle à cette taille ;
- lors de l'étape de diversification, les individus subissent une mutation avec une probabilité P_m ;
- les individus subissent des croisements avec une probabilité P_c à chaque itération ;
- le ou les paramètres liés au critère d'arrêt de l'algorithme ;
- d'autres paramètres potentiels, comme la probabilité de la sélection par tournoi ou encore pour l'élitisme le nombre d'individus conservés entre les générations.

L'opérateur de mutation est souvent plus utilisé pour explorer l'espace des solutions et l'opérateur de croisement pour atteindre les optima locaux. Ainsi, lorsque la mutation est suffisamment aléatoire, les croisements peuvent être plus déterministes. Les opérateurs de croisement et de mutation sont délicats à régler. Trop simple et/ou pas assez fréquente la mutation va s'enfermer dans un optimum local ; trop compliquée et/ou trop fréquente elle va empêcher la convergence de

l'algorithme. De plus, en fonction de la convergence de l'algorithme, une diversification adaptée peut être utile. Dans le but de minimiser ces différents problèmes, des opérateurs adaptatifs peuvent être utilisés [Eib et al. 99, Yang 02].

Plusieurs auteurs proposent des approches spécialisant les opérateurs génétiques en fonction du problème. Dès que l'on sort du domaine de l'étude des AGs sur des fonctions dédiées, et qu'on s'attache à des problèmes particuliers, on va chercher à intégrer une partie de la connaissance particulière que l'on a des problèmes. La littérature foisonne de contributions aussi diverses que variées intégrant des opérateurs génétiques qui utilisent des heuristiques connues pour leurs efficacité.

1.4.3 Variations

De très nombreuses variations ont été proposées dans la littérature sur le modèle classique d'un AG.
Premièrement chaque type de problème nécessite généralement un codage particulier et une fonction d'évaluation propre, mais cela n'explique pas toute la diversité algorithmique que l'on peut trouver.
Certaines versions ont exploré le principe de niche écologique[6], et pour ce faire, diverses méthodes ont été proposées :

- En utilisant une fonction d'évaluation qui va noter plus favorablement les individus peu représentés dans la population [Deb Gol 89].
- En utilisant un opérateur de croisement restreint qui va interdire les croisements entre les individus trop différents [Bea et al. 93].
- Ces deux méthodes peuvent êtres avantageusement combinées. Ce principe de niche écologique est généralement utilisé dans le cadre de problèmes ayant plusieurs optima, et permet à l'AG de converger vers les différents optima simultanément.

Des comportements de sociétés ont parfois été modélisés dans un AG. Par exemple, les algorithmes mimétiques [Seb et al. 97] modélisent des comportements en fonction de deux individus particuliers (les médiateurs) qui sont le pire individu et le meilleur. Ensuite chaque individu peut choisir d'imiter ou de fuir chacun des

[6] Il décrit à la fois :
- la *"position"* dédiée à un *organisme*, une *population* ou plus couramment une *espèce* dans un *écosystème et*
- toutes les situations nécessaires à une population viable de cet organisme.

médiateurs. On aura des comportements sociaux très différents. Thomson et al. proposent un AG religieux [Tho et al. 00] où chaque individu a une religion, et ne pourra en principe que croiser les individus de la même religion. Mais des possibilités de convertir des individus, ou même de croiser des individus de religions différentes existent en fonction de différents critères (en fonction de la *fitness*, du nombre d'individus ayant la même religion...). D'autres comportements ont été modélisés comme la prévention de l'inceste [Esh Sch 91], ou encore des croisements polyandres et polygames, [Eib et al. 95], proposant des recombinaisons multi parentales.

En se rapprochant du modèle biologique, des codages utilisant deux chromosomes par individu (diploïde[7]), avec des règles de dominance et de récessivité ont été proposés dans [Ram Gre 93]. Ce modèle est voué aux environnements changeants, et permet d'obtenir un effet de mémoire. Des modèles utilisant plusieurs chromosomes par individu ont aussi été proposés, et à l'inverse certains ont proposé de supprimer littéralement de l'AG la génétique (la représentation chromosomique et le croisement) [Bal et al. 95].

1.4.4 Parallélisation des algorithmes génétiques

L'avènement des architectures dites massivement parallèles a permis l'exploitation du parallélisme intrinsèque des AGs. L'objectif de la parallélisation des AGs étant de pouvoir explorer efficacement l'espace des solutions afin d'aboutir rapidement à une solution optimale. Il existe pour le moins deux méthodes, utilisées habituellement pour la parallélisation des AGs :

- La première dite parallélisme par îlots : elle consiste à scinder la population de taille n en N sous-populations et à les distribuer sur l'ensemble des machines concernées. Un programme maître déclenche l'exécution de N occurrences du programme de l'AG sur ces machines en leur passant les paramètres nécessaires à leur bon fonctionnement. Les petites populations évoluent et après un certain nombre de générations, chacune des machines donne ses meilleurs individus à une machine de son choix. Cette dernière intègre alors ces nouveaux individus dans sa propre population en éliminant les moins bons individus de sa population.
- La seconde concerne la parallélisation des calculs : dans cette méthode, la population reste totale. On utilise alors des démons de calcul de *fitness* dont la seule fonction est de recevoir un individu et de renvoyer la valeur de sa *fitness*.

[7] Organisme dont les noyaux des cellules, possèdent deux jeux de chromosomes.

Le programme maître se charge de faire la sélection, les croisements, etc. En d'autres termes, il fait évoluer la population, puis répartit les calculs dont il a besoin sur l'ensemble des démons. Enfin, dès qu'il reçoit tous les résultats, l'algorithme commence une nouvelle génération. Par ailleurs, notons que ce mécanisme requiert un grand nombre de communications pour l'envoi des données et leurs évaluations. La méthode n'est donc avantageuse que si le temps passé pour le calcul est grand devant le temps de communication. Elle sera par conséquent utilisée pour des problèmes dont les évaluations de *fitness* prennent beaucoup de temps.

1.5 Les moteurs d'évolution des algorithmes génétiques

On regroupe sous ce nom les ensembles sélection/remplacement, qui ne peuvent être dissociés lors des analyses théoriques du darwinisme au sein des AEs. Un moteur d'évolution est donc la réunion d'une procédure de sélection et d'une procédure de remplacement. Toute combinaison des procédures présentées plus haut pour la sélection est admise. Toutefois, certaines combinaisons sont plus souvent utilisées, que ce soit pour des raisons historiques, théoriques ou expérimentales. Pour cette raison, les noms donnés sont souvent les noms des écoles historiques qui les ont popularisées.

1.5.1 Algorithme génétique générationnel (GGA)

Ce moteur utilise une sélection stochastique pour sélectionner exactement P parents (certains parents peuvent donc être sélectionnés plusieurs fois, d'autres pas du tout). Ces P parents donnent ensuite P enfants par application des opérateurs de variation (avec probabilité donnée). En définitive, ces P enfants remplacent carrément et naturellement les P parents pour la génération suivante. La variante élitiste consiste à garder le meilleur des parents s'il est plus performant que le meilleur des enfants.

1.5.2 Algorithme génétique stationnaire (Steady-state GA– SSGA)

Dans ce moteur, l'idée principale est qu'une grande partie de la population puisse survivre à la prochaine génération. L'algorithme génétique à état stationnaire

(SSGA : Steady State Genetic Algorithm en anglais) [Syswer 89], fonctionne alors de la manière suivante. A chaque génération sont choisis quelques chromosomes parmi ceux qui ont le meilleur score pour créer des chromosomes enfants. Ultérieurement, les chromosomes les plus mauvais sont retirés et remplacés par les nouveaux. En conséquence, le reste de la population survit à la nouvelle génération.

1.5.3 Stratégies d'évolution ((μ, λ)-ES et (μ, λ)-ES)

Sous ces appellations sont regroupés deux moteurs d'évolution, qui constituent une famille d'algorithmes évolutionnaires (cf. sous-section 1.6.3). Pour les deux moteurs, l'étape de sélection est effectuée par un tirage uniforme, on peut dire qu'il n'y a pas de sélection au sens darwinien. On part d'une population de taille μ, λ enfants sont alors générés par application des opérateurs de variation. L'étape de remplacement est ainsi totalement déterministe. Dans le schéma (μ, λ)-ES (avec $\lambda > \mu$), les μ meilleurs enfants deviennent les parents de la génération suivante, alors que dans le schéma $(\mu + \lambda)$-ES, ce sont les μ meilleurs des $\mu + \lambda$ parents plus enfants qui survivent.

1.5.4 Algorithme multi-objectif (NSGA : Non-Dominated Sorting Genetic Algorithm)

Dans le cas de problèmes multicritères, où l'on cherche à optimiser simultanément plusieurs critères, généralement contradictoires, différentes stratégies ont été développées, pour lesquelles l'opérateur de sélection est modifié afin de tendre vers une population répartie le long du front de Pareto[8]. Nous retraçons ici la méthode NSGA : cette méthode propose d'attribuer aux individus une performance biaisée, dépendant de la distance qui les sépare du front de Pareto. La population est classée par fronts avant la sélection, et cela à chaque génération. Les individus non-dominés constituent alors le front 1. Parmi les individus restants, les non-dominés forment le front 2, et ainsi de suite. Ensuite, une performance biaisée est attribuée à chaque individu, dépendant du front auquel il appartient. Une performance élevée est attribuée aux individus constituant le premier front. Ensuite, une stratégie de partage entre les individus du front est réalisée, suivant la distance qui les sépare dans l'espace phénotypique. Cette mesure favorise la diversité et permet l'étalement

[8] On appelle *front de Pareto* d'un problème, l'ensemble des points de l'espace de recherche tels qu'il n'existe aucun point qui est strictement meilleur qu'eux sur tous les critères simultanément.

de la population le long du front de Pareto. Les individus du second front subiront le même traitement, avec une performance initiale inférieure à la plus basse performance attribuée dans le front précédent. La procédure se poursuit ainsi jusqu'à ce qu'une performance biaisée soit associée à chaque individu de la population. La sélection est réalisée ensuite par un processus classique, en considérant comme critère la performance biaisée.
Jusque là, nous n'avons évoqué dans cette section que des techniques communes, applicables à tout problème et surtout à tout espace de recherche. Nous allons passer en revue les autres écoles historiques d'AEs.

1.6 Les autres algorithmes évolutionnaires

La grande famille des algorithmes évolutionnaires renferme en fait quatre éléments : les algorithmes génétiques, la programmation génétique, la programmation évolutionnaire et enfin les stratégies évolutionnaires.

1.6.1 Programmation génétique

La programmation génétique introduite par [Koza 92], dont le modèle s'inspire des algorithmes génétiques, a pour but de créer automatiquement un programme pour résoudre un problème. Pour cela on utilise non pas des chromosomes codant pour une solution, mais de petits programmes qui vont évoluer et s'assembler pour fournir un programme plus complet. Ces programmes sont en général des *S* expressions Lisp qui ont l'avantage d'être facilement représentées par des arbres syntaxiques, où les nœuds sont des opérateurs, et les feuilles des variables ou des valeurs. L'algorithme utilisé pour la programmation génétique suit le même schéma qu'un AG standard : après une initialisation aléatoire où chaque individu est un programme, on effectue la boucle suivante un certain nombre de fois :

- Evaluation des individus en mesurant l'adéquation entre le programme, la solution voulue, et la sélection des individus,
- Application des opérateurs génétiques.

Pour illustrer chacune des étapes utilisées en programmation génétique, on cite l'exemple suivant :
Le problème est de retrouver l'équation d'une courbe, par exemple la représentation d'un polynôme. L'initialisation va consister à générer un ensemble de petits

programmes, dans ce cas des fonctions simples, sous forme d'arbres syntaxiques de S expressions. Afin d'évaluer les programmes, on va comparer les valeurs des ordonnées des points avec les valeurs retournées par les programmes pour un certain nombre d'abscisses. La mutation va consister à remplacer un opérateur par un autre si on est sur un nœud, ou bien remplacer une valeur ou une variable par une autre si on est sur une feuille. Le croisement va permettre à deux programmes de s'échanger des sous arbres syntaxiques. L'idée principale de la génétique est que des S expressions de petite taille dont la *fitness* est élevée vont se recombiner en formant des S expressions plus grandes et de *fitness* plus élevées.
La programmation génétique permet en général de manipuler des structures complexes permettant d'optimiser des réseaux de neurones ou encore des automates cellulaires. Ce modèle va permettre, en utilisant des structures simples au départ, de créer une structure plus complexe dont le comportement devra répondre aux attentes de l'utilisateur.

1.6.2 La programmation évolutionnaire

La programmation évolutionnaire a été introduite par Fogel [Fog et al. 66]. Initialement elle a été créée pour faire évoluer les machines à états finis et par la suite étendue aux problèmes d'optimisation de paramètres. A la différence des autres branches de la famille des AEs, la programmation évolutionnaire n'utilise pas une représentation spécifique. Elle se focalise sur l'opérateur de mutation approprié à un problème spécifique. Pour résoudre un problème, on génère aléatoirement une population de taille μ, et chaque individu i va générer λ descendants suite à l'application de l'opérateur de mutation. Une opération de sélection naturelle permet par la suite de former une nouvelle génération de taille μ à partir des parents et des descendants.

1.6.3 Stratégies évolutionnaires

Les algorithmes utilisant les stratégies évolutionnaires (SEs) ont été développés à peu près à la même époque que les AG et indépendamment par Rechenberg [Rechen 73a] et améliorés plus tard par Schwefel [Schwef 81]. Contrairement aux AG qui utilisent des chaines binaires, les SEs ont été développées pour travailler sur les réels. La version initiale des SEs n'utilise qu'un seul individu qui peut subir des mutations, la sélection choisissant le meilleur du couple parent/enfant ; c'est le

(1+1)-SE. Ensuite des variantes ont utilisé une population d'individus. Ce sont les algorithmes $(\mu+\lambda)$-SE et (μ, λ)-SE. La population POP_t est un ensemble de μ individus qui, par recombinaison et mutation, génèrent λ nouveaux individus. La sélection se fait parmi les $(\mu+\lambda)$ individus pour la stratégie $(\mu+\lambda)$-SE et parmi les λ nouveaux individus pour l'algorithme (μ, λ)-SE pour obtenir la population POP_{t+1}. On pourra trouver une comparaison entre SE et AG dans [Hof Bac 91].

1.7 Approches Hybrides

Les AGs sont des mécanismes simples et efficaces pour explorer un espace donné. A ce propos, ils sont généralement utilisés pour faire avancer ou pour améliorer d'autres algorithmes. Ainsi des approches hybrides ont été proposées et se sont révélées très efficaces sur des problèmes donnés. Usuellement ces approches ne sont efficaces que sur le type de problème pour lequel elles ont été développées, mais sont aussi généralement d'une grande efficacité par rapport aux autres méthodes. La raison principale est souvent que l'opérateur de croisement doit être spécialisé pour ne pas avoir un effet destructeur sur des solutions potentielles découvertes par la recherche locale.

1.7.1 Hybridation avec la liste Tabou

L'AG a été utilisé conjointement avec la méthode Tabou dans une approche proposée dans [Gal Hao 99]. Après une initialisation où chaque individu subit une recherche Tabou sur un certain nombre d'itérations, on rentre dans la boucle de l'AG. À chaque génération deux individus sont choisis aléatoirement dans la population[9], et un croisement spécifique qui génère un enfant est réalisé. Ensuite, cet individu subit là aussi une recherche Tabou pendant un certain nombre d'itérations. Cet algorithme hybride a donné de très bons résultats sur le coloriage de graphes[10].

1.7.2 Hybridation avec le recuit simulé

L'hybridation de l'algorithme génétique avec le recuit simulé a fait l'objet de plusieurs publications. Dans [Bro et al. 89] Brown et al proposent un AG où à chaque

[9] Il s'agit plus ici d'un algorithme génétique de type *Steady State*

[10] *Colorer un graphe* signifie attribuer une couleur à chacun de ses sommets de manière à ce que deux sommets reliés par une arête soient de couleurs différentes.

génération, chaque individu subit une amélioration par recuit simulé. Deux autres versions différentes sont proposées dans [Hir et al. 90, Koa et al. 96]. La première (PGSA) est un algorithme de recuit simulé parallèle qui échange et croise certains individus. La deuxième (GSA) est basée sur un algorithme *Steady State*, dans lequel deux individus choisis au hasard vont être croisés pour ne donner qu'un seul enfant, celui-ci remplace le pire individu de la population. Cet individu est ensuite utilisé dans un processus de recuit simulé pendant un certain nombre d'itérations.

1.7.3 Hybridation avec la recherche locale (Algorithmes Mimétiques)

Les algorithmes mimétiques sont une hybridation entre algorithmes évolutionnaires et recherche locales. Ils ont été proposés par Moscato et Norman [Mos Nor 89]. L'idée de base étant de remplacer le mécanisme stochastique de l'AE par une recherche locale afin d'aider la convergence. Ainsi, on utilise généralement un AG avec une population initiale constituée exclusivement de minima locaux.

1.7.4 Hybridation avec opérateurs spécialisés

Plusieurs approches dites aussi hybrides consistent à spécialiser les opérateurs génétiques en fonction du problème traité. Une manière très utilisée comporte une recherche exhaustive du meilleur voisin lors d'une mutation, ou bien réaliser un croisement qui prend en compte la structure du problème en empêchant les croisements sur des gènes inadaptés. La littérature foisonne de travaux aussi divers que variés incorporant des opérateurs génétiques qui utilisent des heuristiques connues pour leur efficacité. Plusieurs travaux intègrent aussi dans les opérateurs génétiques une partie de la connaissance particulière que l'on a du problème traité. Ainsi, les AGs peuvent avoir une grande efficacité si cette intégration est bien effectuée.

1.8 Conclusion

Dans ce chapitre, nous avons présenté de manière générale les algorithmes évolutionnaires, et en particulier les algorithmes génétiques. Les algorithmes évolutionnaires ont été très largement utilisés et leur succès est dû en partie à la rapidité et la facilité d'implantation qu'ils permettent. Malgré la séduisante facilité

du processus évolutionnaire, produire un algorithme évolutionnaire efficace est une tâche difficile. En effet, les processus évolutionnaires sont très sensibles aux choix algorithmiques et paramétriques et notamment aux choix des représentations. En conséquence, le design d'un algorithme évolutionnaire efficace est difficile. L'expérience a montré que les plus beaux succès sont fondés essentiellement sur une compréhension fine des mécanismes évolutionnaires et surtout sur une très bonne connaissance du problème à traiter. Une forte spécialisation de l'algorithme évolutionnaire est donc souhaitée pour aboutir à des résultats probants, cependant cette spécialisation va à l'encontre de la facilité d'implantation. Un compromis est donc préférable lors de la conception d'un AG. C'est sur ces points importants que nous nous appuyons pour concevoir une approche évolutionnaire dédiée à la détection de communautés dans les réseaux.

2

Notions sur la détection de communautés dans les réseaux

2.1 Introduction

Beaucoup de systèmes complexes dans divers domaines comme la biologie, l'informatique, la linguistique, le commerce, etc., peuvent être représentés de manière abstraite par des réseaux. Une des caractéristiques commune que l'on retrouve dans de nombreux réseaux concerne l'existence de zones plus densément connectées que d'autres. Ces zones sont habituellement appelées communautés et correspondent intuitivement à des groupes de nœuds plus fortement connectés entre eux qu'avec les autres nœuds du réseau. La détection de ces zones dites communautaires est un outil important pour la compréhension des structures et des fonctionnements des grands réseaux. De tels réseaux peuvent être modélisés en termes de graphes, où un nœud représente un membre individuel du système, et une arête représente un lien entre les nœuds selon une relation bien déterminée du système. Ainsi, le concept de *graphe* est généralement utilisé comme modèle de représentation dès que les données sont intrinsèquement liées. Cette structure de données permet en effet de modéliser efficacement les relations entre différents acteurs d'un système complexe.

Ce chapitre tente de rapporter l'essentiel concernant la notion de communauté telle qu'elle a été abordée par différents auteurs et présentera le cadre formel de la détection de communautés. Mais avant cela, il convient d'introduire au préalable des cas de modélisation par des graphes, ainsi que certaines notions et définitions utiles de la théorie des graphes. En l'occurrence, ces concepts sont pour la plupart issus de l'ouvrage de référence de Claude Berge [Berge 70].

2.2 Préliminaires sur les graphes

Définition 2.1 (**Graphe**). Un graphe non-orienté $G = (V, E)$ est composé d'un ensemble V de sommets (ou nœuds) et d'un ensemble E de paires (non ordonnées) de sommets nommées arêtes (ou liens).
Nous adoptons les notations suivantes : n représente le nombre de sommets ($n = |V|$) et m le nombre d'arêtes ($m = |E|$), le graphe est dit d'ordre n et de taille m.
Les arêtes du graphe peuvent être pondérées grâce à une fonction de poids $w : E \rightarrow R^+$ permettant de modéliser plus finement les interactions entre sommets, nous obtenons ainsi un graphe pondéré $G = (V, E, w)$.

- Deux sommets liés dans un graphe sont dits voisins l'un de l'autre.
- On qualifie de *chemin* entre deux sommets, une séquence de liens consécutifs dont ils sont les extrémités ; la longueur de ce chemin sera le nombre de liens qu'il comporte ; la *distance* entre deux sommets sera le minimum des longueurs de chemins allant de l'un à l'autre.
- Si un graphe ne contient pas de liens associant un sommet à lui-même (boucle), on parle de *graphe simple.*

Définition 2.2 (Graphe connexe). Un graphe $G = (V, E)$ est connexe si, quels que soient les sommets u et v de V, il existe un chemin de u vers v.

Définition 2.3 (Graphe complet). Un graphe non orienté $G = (V, E)$ est dit complet si quelque soit la paire $(u, v) \in V$, il existe un arc $\in E$ reliant les deux sommets u et v.
Un graphe complet de n sommets contient $\frac{n(n-1)}{2}$ arcs.

Définition 2.4 (Sous-graphe). Un sous-graphe d'un graphe G est un graphe constitué de certains sommets de G et de toutes les arêtes qui les relient.

Définition 2.5 (Clique). Une clique de G est un sous-graphe complet de G. On parle de k-clique pour désigner un graphe complet de k sommets.

Définition 2.6 (Degré d'un sommet). Dans un graphe non orienté $G = (V, E)$, le degré d'un sommet $v \in V$ est le nombre d'arêtes auxquelles ce sommet appartient :

$$deg(v) = d_v = \text{cardinal}\ (\{(v, u) \in E \mid u \in V\}).$$

Définition 2.7 (Voisinage). Le voisinage d'un nœud correspond à l'ensemble de tous ses nœuds adjacents. Autrement dit, l'ensemble des voisins d'un sommet $v \in V$, notée $N(v)$, est défini comme $N(v) = \{u \in V \mid (v, u) \in E\}$.

Définition 2.8 (Distance). La distance entre deux sommets est la longueur de la plus courte chaîne entre ces sommets; elle est aussi appelée *distance géodésique*.

Définition 2.9 (Diamètre). Le diamètre d'un graphe est la plus grande distance entre deux sommets de ce graphe.

Définition 2.10 (Densité d'un graphe). La densité D d'un graphe $G = (V, E)$ est définie par le rapport du nombre d'arêtes du graphe sur le nombre d'arêtes d'un graphe complet ayant $|V|$[11] sommets : $D = \frac{|E|}{\frac{|V|(|V|-1)}{2}} = \frac{2|E|}{|V|(|V|-1)}$

Définition 2.11 (Graphe biparti). Un graphe biparti $G = (V, E)$ est un graphe possédant deux classes de sommets disjointes V_r et V_b avec $V = V_r U V_b$ et $V_r \cap V_b = \emptyset$ telles qu'il n'existe aucune arête reliant deux sommets de la même classe. Autrement dit, on a : $E \cap (V_r \times V_r) = \emptyset \; et \; E \cap (V_b \times V_b) = \emptyset$.

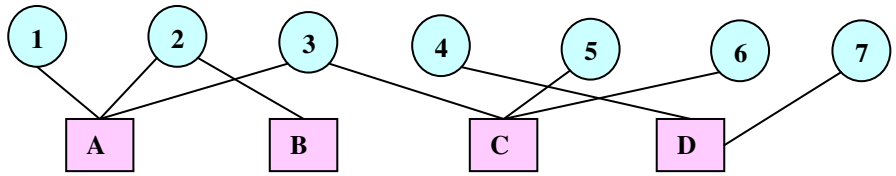


Fig 2.1. Exemple de graphe biparti

Ce type de graphe est utilisé pour modéliser des réseaux dans lesquels on identifie deux types distincts de nœuds (typiquement des agents et des événements).

Définition 2.12 (Projection d'un graphe biparti). En vue de considérer les relations entre sommets d'une même classe qui sont reliés par l'intermédiaire de leurs voisins de l'autre classe, on réalise une projection du graphe biparti sur la classe des sommets que l'on désire étudier.
Soit $G = (V, E)$, avec $V = V_{chiffres} \cup V_{lettres}$ (nous prendrons ici l'exemple d'une projection sur $V_{lettres}$). Nous créons ainsi un nouveau graphe $G_1 = (V_{lettres}, E_{lettres})$ sur

[11] Le cardinal d'un ensemble V noté $|V|$, représente le nombre d'éléments de V.

l'ensemble des sommets $V_{lettres}$. Il existe une arête entre deux sommets i et j dans G, si ces deux sommets sont liés à au moins un voisin commun de $V_{lettres}$: autrement dit : l'arête $\{i,j\} \in E_{lettres} \Leftrightarrow \exists\, u \in V_{lettres} \mid \{i,u\} \in E \text{ et } \{j,u\} \in E$.

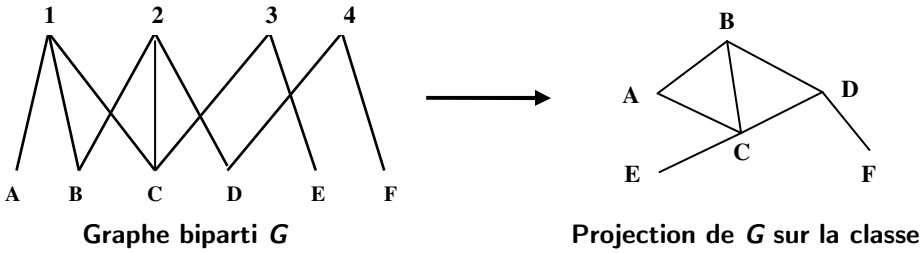


Fig 2.2. Exemple d'un graphe biparti projeté sur une classe

Définition 2.13 (Graphe-arête ou Line graph). Soit $G = (V, E)$ un graphe simple, on appelle graphe-arête (ou graphe représentatif des arêtes) de G, le graphe G^* obtenu de la façon suivante :

- les sommets de G^* représentent les arêtes de G,
- deux sommets de G^* sont reliés si et seulement si les arêtes correspondantes de G sont adjacentes (partagent une extrémité commune).

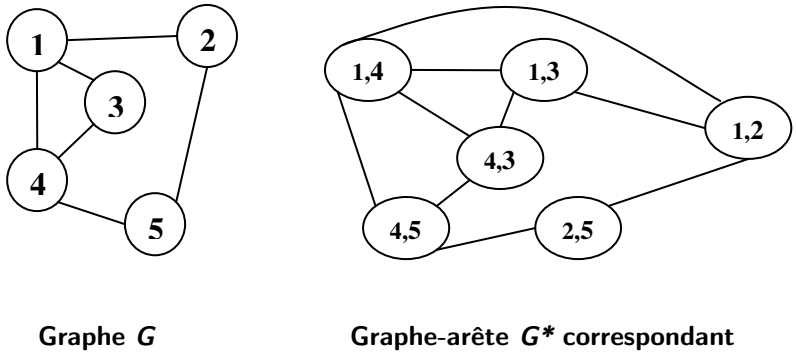


Fig 2.3. Exemple d'un graphe arête

Définition 2.14 (Matrice d'adjacence). Une matrice d'adjacence A d'un graphe G d'ordre n est une représentation matricielle exactement équivalente au graphe. Cette matrice $(n \times n)$ est binaire, $a_{a,b} = 1$ s'il existe un lien entre les nœuds x_a et x_b, $a_{a,b} = 0$ dans le cas contraire. On notera qu'elle est nécessairement symétrique selon notre définition d'un graphe, et pour un graphe simple ses éléments diagonaux sont nuls.

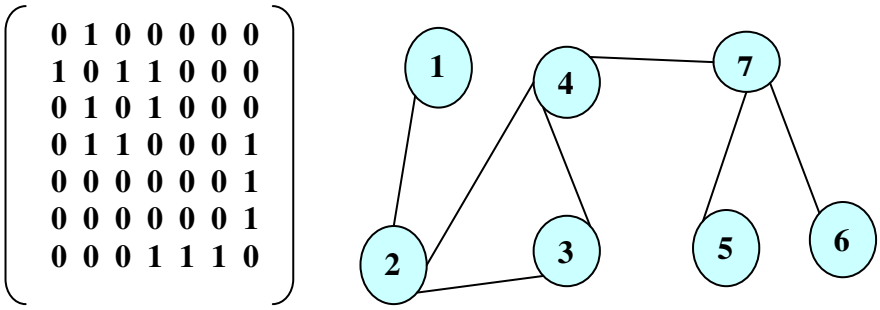


Fig 2.4. Matrice d'adjacence d'un graphe simple, non orienté

Définition 2.15 (Matrice des degrés). Étant donné un graphe $G = (V, E)$ contenant n sommets, la matrice D des degrés de G est une matrice carrée n x n qui contient des informations sur le degré de chaque sommet du graphe. Elle est définie par :

$$d_{i,j} = \begin{cases} deg(sommet\ i) & si \quad i = j \\ 0 & sinon \end{cases}.$$

Définition 2.16 (Matrice Laplacienne). Soient un graphe $G = (V, E)$, A sa matrice d'adjacence et D sa matrice des degrés. La matrice $A_{Lap} = D - A$ est appelée matrice Laplacienne (ou de Laplace) de G [Mohar 97].

Définition 2.17 (Partition des sommets d'un graphe). On considère un graphe $G = (V, E)$ et un ensemble de k sous-ensembles de V, noté $P_k = \{V_1, ..., V_k\}$. On dit que P_k est une partition de G si :

- aucun sous-ensemble de V qui est élément de P_k n'est vide :
  $\forall i \in \{1, ..., k\}, V_i \neq \varnothing$;
- les sous-ensembles de V qui sont éléments de P_k sont disjoints deux à deux :
  $\forall (i, j) \in \{1, ..., k\}^2, \ i \neq j; \ V_i \cap V_j = \varnothing$;
- l'union de tous les éléments de P_k est G :
  $\bigcup_{i=1}^{k} V_i = V$.

Les éléments V_i de P_k sont appelés les parties de la partition.
Le nombre k est appelé le cardinal de la partition, ou encore le nombre de parties de la partition.

Définition 2.18 (Ensembles chevauchants). Deux ensembles X et Y se chevauchent si et seulement si $X \cap Y \neq \varnothing$ et $X \setminus Y \neq \varnothing$ et $Y \setminus X \neq \varnothing$.

Définition 2.19 (coefficient de clustering d'un nœud). Soient un graphe $G = (V, E)$, un nœud $i \in V$, k_i le nombre de voisins de i, ni le nombre d'arêtes entre ces voisins. Le coefficient de clustering du nœud i est défini par :

$$CC_i = \begin{cases} \frac{n_i}{k_i} & k_i > 1 \\ 0 \quad si\ k_i = 0\ ou\ 1 \end{cases}.$$

Ce coefficient traduit en fait la probabilité que deux voisins du nœud i soient reliés.

Définition 2.20 (coefficient de clustering d'un graphe ou coefficient de clustering global). Deux définitions existent pour le coefficient de clustering GC d'un graphe G d'ordre n :

– La première utilise la formule suivante :

$$GC(G) = \frac{3 \times N_\Delta}{N_V}$$

où N_Δ est le nombre de triangles dans le graphe, et N_V le nombre de triplets connexes (trois sommets et au moins deux arêtes). Le facteur 3 est dû au fait que chaque triangle correspond à trois triplets.

– La seconde représente la moyenne du coefficient de clustering de tous les sommets qui ont un degré supérieur ou égal à 2, elle est donnée par la formule suivante :

$$GC(G) = \sum_{i=1}^{n} \frac{CC_i}{n}$$

2.3 Modélisation par des graphes

2.3.1 Divers domaines concernés

Les graphes ou réseaux permettent de modéliser de nombreux phénomènes qui proviennent d'horizons très variés. Nous allons d'abord présenter quelques exemples dans lesquels des graphes sont utilisés comme outil de modélisation de phénomènes complexes, pour lesquels la détection de communautés est une tâche importante. Ceci illustrera la diversité des domaines d'applications possibles. Pour chaque cas, nous identifierons les acteurs du phénomène modélisés par les sommets du graphe, et les interactions entre eux modélisées par des liens ou arêtes entre les sommets.

A- Les réseaux sociaux

Ils représentent un champ d'application important pour la détection de communautés. Dans de tels réseaux, les acteurs sont des individus ou entités sociales (entreprises, associations, pays, etc.), les liens entre eux peuvent être de natures différentes. Ainsi, nous pouvons distinguer plusieurs types de réseaux : *les réseaux d'appels téléphoniques* où deux individus ou numéros de téléphones sont joints s'il y

a eu un appel entre eux [Mau Res 00]. Les *réseaux de connaissance* où deux individus sont joints s'ils se connaissent. *Les réseaux de collaboration* où deux individus sont joints s'ils ont travaillé ensemble, aussi même de nombreux travaux ont étudié les collaborations scientifiques [Newman 01]. *Les réseaux d'échanges* où deux entités sont jointes si elles ont échangé un fichier ou un courrier électronique par exemple [Ebe et al. 02], etc.

B- Les réseaux d'infrastructure

Ils représentent des connections matérielles entre objets distribués dans un espace géographique. Nous pouvons citer *les réseaux de transport* qui modélisent les routes entre les villes ou les liaisons aériennes entre aéroports. *L*es *réseaux de distribution électrique* (câbles entre les lieux de production et de consommation) ou encore *le réseau physique de l'Internet* (câbles entre ordinateurs) [Fal et al. 99].

C- Les réseaux biologiques

Il en existe plusieurs types parmi lesquels nous pouvons citer : *les réseaux métaboliques* où les sommets sont des gènes ou bien des protéines qui sont liés par leurs interactions chimiques [Jeo et al. 00], *les réseaux de neurones* où chaque neurone est connecté à plusieurs autres neurones.

D- Les réseaux d'information

Ces réseaux symbolisent des liens conceptuels de référencement entre des supports d'information. Parmi eux, nous citons : *les réseaux de citation* d'articles et les *graphes du Web* où les sommets sont des pages Web liées par des liens hypertextes [Kle et al. 99].

E- Les réseaux linguistiques

Les réseaux linguistiques relient les mots d'un langage donné et rassemblent entre autres les réseaux des synonymies, en d'autres termes deux mots sont joints s'ils sont synonymes, *les réseaux de co-occurrences* [Fer et al. 01] où deux mots sont joints s'ils apparaissent dans une même phrase d'un écrit, ou encore *les réseaux de dictionnaires* dans lesquels deux mots sont joints si l'un sert à la définition de l'autre [Blo Sen 02].

2.3.2 Les grands graphes de terrain

On rencontre diverses qualifications pour les graphes (ou réseaux) dans le domaine de la détection de communautés. L'appellation "grands graphes de terrain" vient du fait que ces graphes sont de grandes tailles et obtenus de manière empirique en s'appuyant sur des données réelles. Elle fait référence aux mêmes objets dénommés "*Complex network*" ou "*real-world graphs*". Dans ces graphes est la présence d'une forte densité locale et d'une faible densité globale du graphe. Cette propriété fondamentale traduit la capacité des sommets à se regrouper en clusters ou groupes. Elle est à l'origine de la problématique de détection de communautés dans les réseaux. La plupart des graphes de terrain ont en communs des propriétés non-triviales [Wat Str 98] :

- les distributions des degrés de graphes de terrain sont la plupart du temps hétérogènes. En effet, on rencontre beaucoup de sommets avec un faible degré. Quelques sommets avec un très fort degré, qui jouent nécessairement des rôles particuliers par rapport aux autres sommets et tous les degrés intermédiaires sont aussi observables ;
- le degré moyen des sommets est relativement faible par rapport à la taille du graphe. Mais la *densité globale est faible* contrairement à la *densité locale* qui est *élevée* ;
- les graphes de terrain ont quasiment toujours une composante qui contient la très grande majorité des nœuds. Elle est appelée composante géante ;
- La distance moyenne dans la composante géante est petite par rapport au nombre de sommets ;
- Le coefficient de clustering est élevé.

2.3.3 Les graphes aléatoires

En mathématiques, un graphe aléatoire est un graphe qui est généré par un processus aléatoire. Le premier modèle de graphes aléatoires a été introduit par Paul Erdös et Alfréd Rényi dans [Erd Rén 59].
Sommairement, un graphe aléatoire de taille n est un graphe de n sommets dont on a choisi aléatoirement les arêtes, en fixant la probabilité d'avoir une arête entre les paires de sommets (probabilité identique pour chaque paire de sommets).
Les graphes aléatoires sont utilisés pour évaluer la complexité en moyenne d'algorithmes utilisant les graphes ou encore pour modéliser de vrais réseaux. Ces

graphes ont des distributions de degrés homogènes et un faible coefficient de clustering. Or les réseaux du monde réel comme le web, l'internet, la collaboration des auteurs, etc. se détournent considérablement de ce modèle aléatoire. De ce fait, ils ne sont pas un bon modèle pour les graphes de terrain. Un domaine de recherche essentiel [Bar Alb 99, Gui Lat 06] se consacre d'ailleurs à construire des modèles de graphes permettant d'imiter les propriétés des graphes de terrain.

2.4 Détection de communautés : définitions et intérêts

Nous rapportons dans les sections suivantes, les diverses définitions relatives à la notion de communautés. Nous donnons ensuite la définition formelle et l'intérêt de la détection de communautés dans les réseaux.

2.4.1 Qu'est ce qu'une communauté ?

A- Notion intuitive

Le terme communauté désigne, comme son radical l'indique, un ensemble d'individus ayant une ou plusieurs caractéristiques en *commun*. Cette notion de communauté a fait l'objet de nombreuses études de la part des sociologues depuis plusieurs décennies, notamment dans le cadre de l'analyse des réseaux sociaux [Scott 00]. Les réseaux sociaux sont des structures modélisant les relations sociales (par exemple l'amitié, la collaboration, la parenté, etc.) qui existent entre un ensemble d'individus appelés aussi acteurs. Ceci nous donne nécessairement une idée intuitive de ce qu'est une communauté.

Depuis quelques années, l'utilisation du mot communauté s'est généralisée à d'autres types de réseaux et n'est désormais plus réservée aux réseaux sociaux. Nous retrouvons cette extension d'usage dans plusieurs disciplines telles que l'informatique [Dou et al. 07], la physique [Hasting 06] ou encore la biologie [Bor Sch 03]. En informatique par exemple, la notion de communauté est devenue dès la fin des années 90 très populaire avec le développement du Web et de la recherche d'information basée sur les liens hypertextes. Des chercheurs comme Kleinberg [Kle et al. 99] ou Flake [Fla et al. 00] ont introduit la notion de *communauté web* qui désigne un ensemble de pages web traitant d'une même thématique ou d'un même sujet (ou "topic" en anglais). Ainsi, des structures communautaires sont définies et

observées dans de nombreux réseaux, et jouent un rôle important dans leur organisation ou leur structuration. De ce fait, il est fondamental de les définir nettement et de les détecter automatiquement.

Bien qu'aucune définition formelle de ce qu'est une communauté ne soit actuellement reconnue, on désigne intuitivement par ce terme un groupe de nœuds du réseau plus fortement connectés entre eux qu'avec les autres nœuds du réseau (cf. figure 2.5).

Fig 2.5. Exemple de structure communautaire dans un réseau

B- Diverses définitions adoptées

La première question qu'on se pose inévitablement est : comment formuler mathématiquement l'appartenance d'un nœud à une communauté plutôt qu'à une autre ? Si son sens sociologique est assez clair pour un être humain, il devient tout à fait inexploitable lorsqu'il s'agit d'un calcul exact.
Dans la théorie des graphes, il existe déjà quelques définitions qui répondent à ces contraintes. Nous citons :

- *Une k-clique* est un sous-graphe complet maximal de k sommet, où chaque nœud est lié avec tous les autres. Il est certain qu'une *k-clique* possède les propriétés requises pour une communauté, mais elle impose aussi des restrictions importantes, surtout pour les réseaux peu denses.

- Une structure moins stricte est le *k-core*, un sous-graphe de dimension supérieure à k, ou chaque nœud doit avoir des liens vers au moins k autres nœuds. Même si les contraintes d'un k-core sont moins dures, il désavantage les nœuds de faible degré.

  Les notions de *k-clique* et de *k-core* sont basées uniquement sur l'analyse d'un sous-graphe donné.
- Une troisième définition, qui prend en compte la totalité du graphe, est celle d'un *LS set*. Un *LS set* est un groupe de nœuds qui présente la propriété que chaque sous-groupe de ce groupe possède plus de liaisons vers l'intérieur du groupe que vers le reste du graphe.

A part ces définitions, assez limitatives, qui peuvent être utilisées aussi dans la détection des communautés, il existe trois autres tentatives d'exprimer mathématiquement le concept. Deux de ces définitions, celle de communauté au sens *fort* et celle de communauté au sens *faible*, sont données par Radicchi et al. [Rad et al. 04].

La définition *forte* de la communauté tient compte de chaque nœud du sous-ensemble. Ainsi, un sous-graphe V est considéré comme une communauté si

$$k_i^{in}(V) > k_i^{out}(V), \quad \forall\, i \in V$$

où $k_i^{in}(V)$ représente le nombre de liens du nœud i avec les nœuds de l'ensemble V et $k_i^{out}(V)$ est le nombre de liens du nœud i vers l'extérieur de la communauté formée par les nœuds de l'ensemble V.

Cette définition exige une contrainte sur chaque nœud, ce qui n'est pas le cas pour une communauté au sens *faible* dont la contrainte est donnée par :

$$\sum_{i \in V} k_i^{in}(V) > \sum_{i \in V} k_i^{out}(V).$$

Pour ainsi dire, dans une communauté *forte* chaque nœud a plus de liens vers sa communauté que vers le reste du graphe et dans une communauté *faible* le nombre de liens internes est supérieur au nombre de liens externes.

Plus récemment, Hu et al. ont proposé une nouvelle définition d'une communauté, encore moins contraignante [Hu et al. 08]. Dans leur définition, un nœud i fait partie de la communauté V_a s'il a au moins autant de liens vers cette communauté que vers n'importe quelle autre communauté, autrement dit, il vérifie la relation suivante :

$$k_i^{in}(V_a) \geq k_i^{in}(V_b), \quad \forall\, b \neq a, \ \forall\, i \in V.$$

Cependant, il existe peu d'algorithmes qui utilisent directement ces définitions. Dans la plupart des cas, on ne regarde même pas si le partitionnement final les vérifie, mais on préfère comparer les propriétés de ce partitionnement avec un modèle

de référence qui provient d'une topologie semblable mais sans une structure modulaire le "null model" (cf. section suivante).

Le fait qu'il n'existe pas de définition usuellement reconnue pour la notion de communauté, rend plus difficile la construction d'un algorithme accepté par tous, mais, en même temps, cela permet de diversifier et de particulariser les approches proposées pour détecter ce type de structures.

C- Définitions fondées sur une fonction de qualité du partitionnement

Une définition plus moderne de la notion de communauté est basée sur l'utilisation d'une fonction de qualité. Cette famille de définitions a d'ailleurs reçu l'attention d'un nombre important de chercheurs et cela depuis la publication en 2004 d'un article par Newman et Girvan [New Gir 04]. Une fonction de qualité est une fonction qui associe à un sous-graphe S une valeur quantifiant le fait que S correspond à une communauté. Suite à l'apparition de cet article de référence, plusieurs fonctions de qualité ont été proposées par d'autres chercheurs. Nous présentons ici trois de ces fonctions. La première est largement utilisée, il s'agit de la modularité de Newman [New Gir 04], la deuxième est celle de Mancoridis [Man et al. 98]. Nous l'avons adaptée aux graphes bipartis (cf. chapitre 4). Enfin, la dernière est la Performance définie dans [Bra Erl 05]. Le lecteur intéressé peut consulter l'ouvrage de synthèse [Bra Erl 05] où d'autres fonctions de qualité sont rapportées.

Modularité de Newman

La fonction de qualité introduite par Newman et Girvan [New Gir 04] connue sous le nom de la modularité réseau Q, est une métrique de qualité pour l'évaluation du partitionnement d'un réseau en communautés. Elle a été largement utilisée dans les travaux sur les réseaux. Elle est fondée sur l'idée intuitive que les réseaux aléatoires ne renferment pas de structure communautaire. Par conséquent un sous-graphe forme une communauté si la distribution des liens entre ses nœuds n'est pas due au hasard. Elle se base alors sur la différence entre le nombre de liens à l'intérieur d'une communauté et le nombre de liens attendus à l'intérieur de cette communauté si les liens apparaissent aléatoirement dans le graphe tout en respectant la distribution des degrés des nœuds (″null model″). Une communauté est d'autant mieux appréciée que sa proportion d'arêtes internes sera supérieure à sa proportion attendue d'arêtes. Ceci

est synthétisé dans l'équation 2.1 donnant la modularité, et que l'on cherchera toujours à maximiser :

$$Q = \sum_{C_i} e(C_i) - a(C_i)^2 \qquad (2.1)$$

où $e(C_i)$ représente la proportion d'arêtes ayant les deux extrémités dans C_i et $a(C_i)$ est la probabilité pour qu'une arête ait une extrémité dans C_i.

Autrement dit, considérons un graphe G non pondéré et non orienté, avec n sommets et m arêtes, dont les sommets sont dans les communautés $C_1, ..., C_k$. Soient $A[n, n]$ la matrice d'adjacence du graphe, et $P[n, n]$ la matrice d'adjacence du "null model" correspondant, où $P[i, j] = d_i \times d_j / 2m$ est la probabilité dans le modèle de référence telle qu'il existe une arête entre les sommets i et j de G de degrés respectifs d_i et d_j. L'expression détaillée de la modularité de Newman est donnée par l'équation 2.2.

$$Q = \frac{1}{2m} \sum_{ij} (A[i,j] - P[i,j])\, \partial(C_i, C_j) \qquad (2.2)$$

où $\partial(C_i, C_j) = \begin{cases} 1 & si \quad C_i = C_j \\ 0 & sinon \end{cases}$

La modularité Q peut avoir des valeurs entre -1 et 1 : $-1 \leq Q \leq 1$. Une partition contenant une communauté unique regroupant tous les sommets, possède une modularité de valeur nulle. Cette définition a été introduite pour des graphes non-pondérés, mais nous pouvons l'adapter aux graphes pondérés en remplaçant le nombre d'arêtes internes par le poids total des arêtes internes, et en remplaçant le nombre total d'arêtes par le poids total du graphe. Enfin, remarquons que le calcul de la modularité d'une partition peut être effectué avec une complexité temporelle $O(m)$.

Exemple :

Considérons le graphe G de la figure 2.6, qui présente deux communautés perceptibles et supposons que l'on souhaite comparer la qualité des deux partitions suivantes :
$P1 = \{\{A, B, C, D\}, \{E, F, G, H\}\}$ et $P2 = \{\{A, B\}, \{C, D, E, F, G, H\}\}$.

Fig 2.6. Graphe à 8 sommets présentant deux communautés

Le calcul des modularités de *P1* et *P2* indique que $Q(P1)= 0.42$ et $Q(P2)= 0$. Ce résultat signifie que la partition *P1* est meilleure que *P2* et que cette dernière ne correspond pas à une structure communautaire.

Modularité de Mancoridis

La modularité MQ (Modularity Quality) a été introduite par Mancoridis [Man et al. 98] pour mesurer la qualité d'une partition de nœuds du réseau. Cette métrique repose sur la différence entre les ratios de connectivité interne et externe des communautés. Elle est définie comme suit par l'équation 2.3 :

$$MQ = \frac{1}{k}\sum_{i=1}^{k}\left[S(C_i,\ C_i) - \frac{1}{(k-1)}\sum_{i,j=1\ i\neq j}^{k} S(C_i,\ C_j)\right] \qquad (2.3)$$

avec $S(C_i,\ C_j) = \frac{E(C_i, C_j)}{|C_i|\times|C_j|}$.

$E(C_i,\ C_j)$ représente le nombre de liens entre les nœuds des classes C_i et C_j, et le produit $|C_i| \times |C_j|$ représente le maximum de liens qu'on peut avoir entre les noeuds des classes C_i et C_j.

L'équation 2.3 peut s'écrire sous la forme $MQ = MQ^{+} - MQ^{-}$ avec $-1 \leq MQ \leq 1$.
MQ^{+} et MQ^{-} représentent respectivement la cohésion interne et la cohésion externe des classes.

La performance Q^P

Une définition de cette fonction de qualité est donnée dans [Bra Erl 05]. Elle compte le nombre de paires de nœuds qui sont en accord avec la partition en communautés. Deux nœuds reliés par une arête sont en accord avec la partition s'ils appartiennent à la même communauté. Deux nœuds qui ne sont pas liés par une arête sont en accord avec la partition s'ils appartiennent à deux communautés différentes. La performance $Q^P(P)$ d'une partition P est la proportion de paires de nœuds en accord avec la partition :

$$Q^P(P) = \frac{\left|\{\{u,v\} \in E,\ C(u) = C(v)\}\right| + \left|\{\{u,v\} \notin E,\ C(u) \neq C(v)\}\right|}{\frac{1}{2}n(n-1)}$$

C(u) représente la communauté de la partition P qui contient le nœud u. Nous avons donc $0 \leq Q^P(P) \leq 1$. Il n'est pas possible de généraliser directement et simplement cette fonction de qualité à des graphes pondérés. Le nombre d'arêtes internes aux communautés peut être remplacé par leur poids total. En revanche, il n'est pas possible d'attribuer un poids aux paires de nœuds non liés. Plusieurs généralisations possibles sont proposées dans [Bra Erl 05]. Le calcul de la performance d'une partition peut être aussi effectué avec une complexité temporelle $O(m)$.

D- Communautés chevauchantes

La définition de la communauté exige que les communautés C_i soient disjointes, mais en réalité rien n'empêche l'existence de communautés recouvrantes. Dans de nombreux réseaux réels, les nœuds du réseau peuvent appartenir à plus d'une communauté. Dans ce cas, on parle de chevauchement de communautés et on utilise aussi le terme de couverture ou *décomposition*, plutôt que de partition, dont la définition standard interdit des adhésions multiples de nœuds.

Les algorithmes traditionnels de détection de communautés affectent chaque nœud à une seule communauté. Ce faisant, ils négligent des informations potentiellement pertinentes. Les nœuds appartenant à d'autres communautés sont susceptibles de jouer un rôle important d'intermédiaires entre les différents compartiments du réseau. Dans les réseaux biologiques d'interactions protéine-protéine par exemple, nombreuses protéines peuvent avoir plusieurs fonctions. La détection de communautés chevauchantes dans de tels réseaux permet de prédire la ou les fonctions des protéines. Un exemple de communautés chevauchantes est donné dans la figure 2.7. Le réseau qui y est présenté, possède trois communautés chevauchantes, avec deux nœuds chevauchants (nœuds en rouge). Le nœud chevauchant Zds1 appartient à trois communautés (jaune, verte et violette), le nœud Zds2 quant à lui, appartient aux communautés verte et violette.

La détection de telles communautés est un problème bien plus difficile que la détection de communautés disjointes. En effet, la recherche de communautés chevauchantes introduit un paramètre supplémentaire, la composition de nœuds dans les différentes communautés, ce qui augmente le nombre de chevauchements

possibles. Par conséquent, la recherche de communautés chevauchantes exige généralement beaucoup plus de calculs que la détection de communautés disjointes.

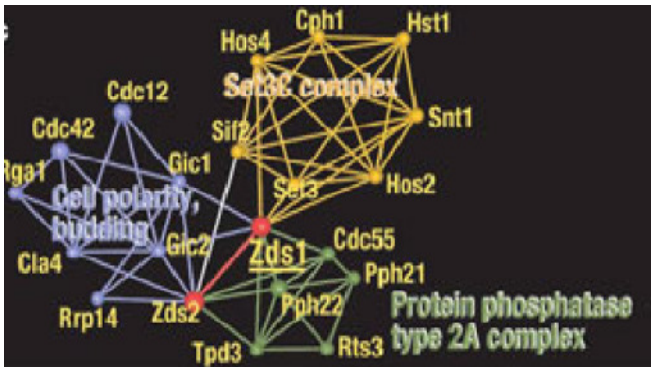


Fig 2.7. Exemple de communautés chevauchantes dans un réseau, extrait de [Pal et al. 05]

2.4.2 Détection de communautés

Ces dernières années, et en particulier depuis l'article de référence de Girvan et Newman [Gir New 02], l'identification de structures communautaires a suscité l'intérêt d'un grand nombre de chercheurs dans diverses disciplines, notamment avec l'entrée en jeu des physiciens et des mathématiciens. Depuis, une panoplie de techniques a été proposée et de nouvelles méthodes continuent régulièrement d'apparaître. Ceci étant, nous pouvons dire que l'identification de structures communautaires est un domaine vaste et interdisciplinaire auquel contribuent plusieurs communautés scientifiques.

Afin de disposer d'un cadre rigoureux, nous allons donner une définition du problème basée sur l'optimisation de fonctions de qualité.

A- Définition formelle du problème

Soit une fonction de qualité Q, qui à toute partition P de V associe un score de qualité du point de vue de la notion de communauté considérée. Le but de la détection de communautés dans un graphe $G = (V, E)$ est de trouver une partition $P = \{C_1, \ldots, C_k\}$ de l'ensemble V des sommets telle que $Q(P)$ est maximale. Les communautés C_i sont donc disjointes (i.e. $\forall\, i \neq j, C_i \cap C_j = \varnothing$) et recouvrent V (i.e. $\cup_i C_i = V$).

Notons que nous ne connaissons a priori ni le *nombre,* ni la *taille* des communautés qui composent la partition recherchée.
Comme le montre cette définition, nous n'avons pas a priori de définition de la notion de communauté : tout se base sur la fonction de qualité Q qui joue donc un rôle prépondérant. La fonction de qualité donne un score à toute partition. Ce score mesure à quel point les communautés de cette partition vérifient des critères caractéristiques des communautés. En particulier, ces fonctions de qualité tiennent généralement compte des densités de liens à l'intérieur et entre les communautés. L'optimisation d'une fonction de qualité sur toutes les partitions possibles est habituellement un problème NP-difficile.

B- Intérêts

L'existence de structures communautaires dans les réseaux constitue une caractéristique commune indépendante du domaine d'origine du réseau. La détection de communautés peut donc être vue comme un procédé commun qui peut s'appliquer à tous les réseaux rencontrés. La structure communautaire existe dans de nombreux réseaux réels et, la plupart du temps, elle a une signification concrète en termes d'organisation.

En réalité, les communautés peuvent avoir des interprétations différentes suivant le type de réseau considéré. Ainsi dans les réseaux sociaux, les communautés peuvent représenter des groupes d'individus ayant des activités communes ou des intérêts communs. De manière complètement différente, dans les réseaux d'interaction Protéine-Protéine en biologie, les communautés correspondent généralement à des ensembles de protéines qui collaborent à une même fonction cellulaire. L'identification des structures communautaires permet de mieux comprendre la structure de ces réseaux, de regrouper et d'identifier les nœuds qui jouent potentiellement des rôles semblables. Enfin les algorithmes de détection de communautés peuvent aussi être utilisés comme des éléments de base dans la conception d'autres tâches plus difficiles, parmi lesquelles on peut citer la parallélisation et la visualisation.

2.5 Conclusion

Nous avons consacré le premier volet de ce chapitre à l'introduction de certains concepts de base indispensables (issus de la théorie des graphes) afin de décrire avec aisance les notions utilisées dans les différents travaux rencontrés et l'approche que nous proposons. Nous nous sommes particulièrement penchés dans le deuxième volet de ce chapitre sur les différentes définitions relatives à la notion de communauté. Nous avons par ailleurs, décrit formellement le problème de la détection de communautés. Dans le chapitre suivant, nous proposons un état de l'art sur la détection de communautés dans les réseaux.

3

Détection de communautés : un état des lieux

3.1 Introduction

Depuis la publication de l'article fondateur de Girvan et Newman [Gir New 02], la détection de communautés ne cesse d'attirer l'attention de nombreux chercheurs du domaine [New Gir 04, Barber 07, Far et al. 07, Fortuna 10, Gregory 10, Kel et al. 11, Wei et al. 11, Sou et al. 13]. La détection de communautés consiste en un procédé commun s'appliquant à tout type de réseau, permettant le partitionnement de nœuds en communautés. L'ensemble de ces communautés doit répondre à certains critères de qualité, en particulier la modularité réseau. De nouvelles approches sont de plus en plus proposées afin de minimiser les coûts de détection en termes de temps et d'espace, tout en maximisant la qualité des partitions trouvées. Selon les cas, différents compromis (sur le temps, l'espace, la définition adoptée, etc.) sont permis ; ce qui se traduit par la multiplication de méthodes de détection proposées.
Dans ce chapitre, nous décrivons les principales approches proposées dans la littérature. Nous nous sommes limités dans le cadre de cette thèse uniquement aux approches qui ont reçu le plus d'attention de la part des chercheurs du domaine. De ce fait, la liste des travaux cités, reste non exhaustive. Pour mettre en exergue les caractéristiques des méthodes, nous avons regroupé ces dernières selon certains critères comme le type de communautés découvertes (disjointes ou chevauchantes), le type de réseaux pour lesquels elles sont conçues (monopartis ou bipartis) et le type d'approche utilisée.

3.2 Détection de communautés disjointes[12]

3.2.1 Communautés disjointes dans des réseaux monopartis

A- Les méthodes classiques

Approches basées sur le partitionnement de graphe

Le but du partitionnement de graphe est de diviser un graphe G en plusieurs sous-graphes $C_1, \ldots C_k$ (correspondants à des communautés) tels que le nombre total de liens (appelé aussi *taille de la coupe*) entre les différents sous-graphes soit minimal. En pratique, la plupart des approches de partitionnement de graphes procèdent par une division du graphe en deux sous-graphes, puis par un partitionnement récursif des deux sous-graphes ainsi obtenus. L'arrêt du partitionnement a lieu lorsque le nombre de communautés souhaité est atteint. Mais cette approche ne convient pas, à proprement parler, à la détection de communautés, car elle requiert des connaissances préalables sur le nombre de communautés recherchées et leurs tailles. Les deux méthodes générales ayant connu le plus de succès sont la méthode de bissection spectrale et la méthode de Kernighan et Lin.

1- La méthode de bissection spectrale [Barnes 82]

Comme son nom l'indique, cette méthode est basée sur le calcul des vecteurs propres. Pus précisément, elle est basée sur le premier vecteur propre non trivial de la matrice de Laplace. Le graphe est ainsi séparé en deux parties en fonction du signe de leur composante selon ce vecteur propre. Cette méthode fonctionne avec une complexité temporelle $O(n^3)$.

2- La méthode de Kernighan et Lin [Ker Lin 70]

Cet algorithme procède par bissection afin de trouver la coupe du graphe minimisant le nombre d'arêtes existantes entre les deux groupes. Une coupe de bonne taille est choisie aléatoirement comme point de départ en adoptant une heuristique gloutonne. La bissection est perfectionnée progressivement en effectuant des permutations de sommets entre les communautés. A chaque étape, on échange les

[12] Il est à noter que n représente le nombre de sommets (nœuds) d'un graphe, tout au long de ce chapitre.

deux sommets fournissant la meilleure diminution du nombre d'arêtes externes. Ces échanges sont réalisés de sorte à ne jamais modifier deux fois un même sommet. Ainsi, nous passons exactement par $n/2$ étapes de calcul, et la meilleure partition rencontrée au cours du déroulement de l'algorithme est retenue. La complexité temporelle de l'algorithme est, dans le pire des cas, de l'ordre $O(n^3)$.
Pour de plus amples détails, un état de l'art complet sur les techniques de partitionnement de graphes se trouve dans [Coo Hol 06] et [Schaeff 07].

Approches basées sur le clustering par partitionnement

Le problème de détection de communautés est considéré sous l'angle de clustering de données pour lequel on choisit une distance (ou une similarité) appropriée [Ait et al. 09a, 09b, 12]. Le but du clustering par partitionnement est de regrouper un ensemble d'objets en clusters (classes ou groupes) tels que chaque objet appartienne à au moins un cluster. Les divers algorithmes appartenant à cette famille d'approches diffèrent par la fonction de qualité utilisée pour évaluer la pertinence d'un partitionnement, ainsi que par la distance choisie pour effectuer les regroupements (clustering).

B- Les méthodes de clustering hiérarchique

Au lieu de constituer une partition "plate" comme cela se produit dans le clustering par partitionnement, le clustering hiérarchique ne construit pas une partition en k clusters, mais construit plutôt une hiérarchie de partitions se présentant sous la forme d'un arbre appelé également dendrogramme et contenant $n-1$ partitions (cf. figure 3.1). L'intérêt de cet arbre est de pouvoir donner une idée du nombre de clusters existants effectivement dans l'ensemble des objets. Chaque coupure de l'arbre fournit une partition ayant d'autant moins de clusters que l'on coupe plus haut.
En pratique, une fois l'arbre construit, l'utilisateur n'est pas forcément intéressé par la hiérarchie entière mais par une seule partition obtenue en coupant l'arbre généré par une droite horizontale. Se pose donc le problème de trouver la meilleure méthode pour couper l'arbre au bon endroit, et par conséquent, de décider du nombre adéquat de clusters trouvés. L'approche la plus utilisée pour résoudre ce problème est basée sur un principe simple. Elle consiste en fait à couper l'arbre au niveau où la variation

de la distance minimale entre les classes est maximale, c'est-à dire là où les clusters sont les plus éloignés (pour la figure 3.1, la méthode propose trois clusters). Mais en général, on donne le nombre k de clusters désirés. Ce qui impose dans le cas de détection de communautés, de connaitre a priori le nombre de communautés recherchées.

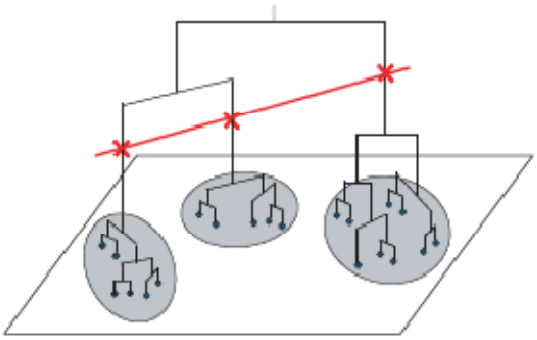


Fig 3.1. Arbre hiérarchique dit Dendrogramme

Les algorithmes de clustering hiérarchiques sont de deux types : les algorithmes ascendants et les algorithmes descendants.

Approches hiérarchiques ascendantes (agglomerative clustering)

Les algorithmes de clustering hiérarchique ascendant supposent au départ que chaque objet (ou nœud) forme un cluster (ici une communauté). Les clusters les plus proches sont ensuite fusionnés de manière récursive jusqu'à obtenir un seul cluster contenant tous les objets. Le calcul de proximité entre deux clusters nécessite l'utilisation d'une mesure de proximité entre les objets (distance euclidienne, coefficient de Jaccard, etc.), ainsi qu'un critère ou indice d'agrégation. Il existe dans la littérature plusieurs indices d'agrégation. Le plus simple (*single linkage*) considère que la distance entre deux communautés est la distance minimale entre deux sommets de celles-ci. A l'opposé, on peut considérer la distance maximale (*complete linkage*). De manière intermédiaire, on peut considérer que la distance entre deux communautés est la moyenne des distances entre chaque paire de leurs sommets (*average linkage*). Il est encore possible dans certains cas de représenter chaque communauté par un sommet moyen (*centroïd*) et de considérer leurs distances respectives. La dernière méthode dite de *Ward*, consiste à minimiser la somme des carrés des distances de chaque sommet au sommet moyen représentant sa communauté. Dans cette catégorie, nous citons les trois approches suivantes.

1- Approche basée sur les propriétés spectrales de la matrice Laplacienne du graphe

Un algorithme de classification hiérarchique ascendante a été proposé par Donetti et Munoz [Don Mun 05] pour l'identification de structures communautaires. Ces auteurs ont testé leur algorithme en utilisant deux mesures de proximité (distance euclidienne et similarité du cosinus) ainsi que deux indices d'agrégation (lien minimum et lien maximum). Ils rapportent que la similarité du cosinus permet une meilleure détection de la structure de communautés que la distance euclidienne.
Concernant l'indice d'agrégation, leurs résultats n'ont pas permis de conclure en faveur de l'un des deux indices comparés ; dans certains cas le lien minimal était meilleur alors que dans d'autres cas c'est le lien maximal qui l'était. Pour le choix de la meilleure partition, Donetti et Munoz utilisent la modularité de Newman.
L'originalité de cette approche est due à la méthode utilisée pour le calcul de proximité entre les N objets. Ce calcul n'est pas fait dans l'espace initial de dimension N (où chaque dimension correspond à un objet) mais plutôt dans un espace de dimension D (où $D< N$) formé par les D premiers vecteurs propres non triviaux (i.e. ceux associés aux D plus petites valeurs propres différentes de zéro) d'une matrice particulière appelée *matrice de Laplace* (cf. section 2.2). Il s'agit d'une matrice dont les propriétés ont été beaucoup étudiées par les chercheurs dans le cadre du partitionnement de graphes [Agg et al. 10]. Donetti et Munoz montrent que la projection des objets sur le nouvel espace permet de faire un "pré-regroupement" des nœuds similaires, ce qui permet d'obtenir de meilleurs indices de proximité.

2- Approche basée sur l'optimisation de la modularité de Newman

Newman et Girvan ont introduit une notion de modularité, à l'aide d'une fonction Q mesurant la qualité d'une partition du graphe en communautés [New Gir 04]. Afin de maximiser cette quantité, l'algorithme *glouton* proposé fusionne à chaque étape les communautés permettant d'avoir la plus grande augmentation de la modularité.
Afin d'augmenter les performances de l'algorithme, seules les communautés ayant une arête entre elles peuvent être agrégées à chaque étape. Chaque agrégation se fait en $O(n)$ et la mise à jour des valeurs des variations de Q (pour chaque nouvelle agrégation possible) peut être effectuée facilement en $O(m)$. La complexité globale est donc en $O((m+n)n) = O(mn)$. Cette méthode est très rapide et permet de traiter de

très grands graphes. La qualité des partitions obtenues est cependant moins bonne qu'avec des méthodes plus coûteuses.
La complexité de cette méthode a été améliorée par Clauset [Cla et al. 04] en utilisant une structure de données adaptée. Un autre perfectionnement de cette méthode a été proposé par Wakita et Tsurumi [Wak Tsu 07] afin de traiter des graphes de taille encore plus importante.

3- Approche utilisant les marches aléatoires et clustering hiérarchique

Les marches aléatoires dans les graphes sont des processus aléatoires dans lesquels un marcheur est positionné sur un sommet du graphe et peut à chaque étape se déplacer vers un des sommets voisins. Le comportement des marches aléatoires est étroitement lié à la structure du graphe. Pons et Latapy [Pon Lat 05] [Pons 07] ont proposé un algorithme de clustering hiérarchique ascendant dans lequel le calcul des distances entre les nœuds (ou objets) repose sur le principe de la marche aléatoire. Plus précisément, la distance entre deux nœuds i et j est exprimée par la différence entre le comportement de deux marcheurs aléatoires commençant respectivement aux nœuds i et j et effectuant une marche de longueur fixe. Pour le calcul de proximité entre deux communautés, Pons et Latapy généralisent la distance proposée pour les nœuds à des ensembles de nœuds (i.e. communautés). Une fonction de qualité basée sur la similarité des nœuds est utilisée à la fin pour le choix de la meilleure partition (i.e. meilleure structure de communautés).

Approches hiérarchiques descendantes (separative clustering)

Le clustering hiérarchique descendant procède de manière inverse au clustering hiérarchique ascendant. On commence au début par une partition constituée d'un seul cluster contenant les N objets à regrouper. Ce cluster est ensuite divisé en deux clusters, puis de manière récursive tout cluster est éclaté en deux jusqu'à l'obtention de N clusters singletons. Le choix, à chaque fois, du cluster à diviser représente l'opération la plus importante et la plus délicate des méthodes descendantes. Dans cette catégorie, nous rapportons ci-dessous les trois approches dédiées à la détection de communautés.

1- Approche basée sur la centralité d'intermédiarité

Girvan et Newman dans l'article [Gir New 02] proposent (cf. l'algorithme 3.1) de retirer à chaque étape une arête du graphe, ce qui peut parfois avoir pour conséquence de diviser une composante connexe (considérée comme un cluster) en deux composantes connexes. L'arête à retirer correspond à celle qui possède la plus forte *centralité d'intermédiarité*. La centralité d'intermédiarité d'une arête est égale au nombre total de chemins géodésiques qui utilisent cette arête divisé par le nombre total de chemins géodésiques dans le graphe (cf. étape 2 de l'algorithme 3.1). En ce qui concerne le choix de la meilleure partition, Newman et Girvan utilisent la modularité Q de Newman. L'algorithme proposé calcule la centralité de toutes les arêtes en $O(mn)$. Ce calcul est effectué à chaque étape sur le graphe obtenu après retrait des arêtes. La complexité temporelle de l'algorithme est donc en $O(m^2n)$.

Algorithme 3.1- Algorithme de Girvan et Newman

Entrée : un graphe non orienté $G = (V, E)$ d'ordre N
Sortie : P une partition en communautés
Début

1. Initialisation : $T = \{v_1, ..., v_N\}$, $G' = G$;

2. Calculer la centralité d'intermédiarité pour chaque arête e_i du graphe G'

$$C^{int}(e_i) = \sum_{j=1}^{N} \sum_{k=1}^{N} \frac{g_{jk}(e_i)}{g_{jk}} ;$$

3. Retirer du graphe G' l'arête e_m ayant la plus grande centralité d'intermédiarité

$$E' = E' \setminus \left\{ e_m \in E' \mid e_m = argmax_{e_j \in E'} C^{int}(e_j) \right\};$$

4. Identifier l'ensemble $C = \{C_1, ..., C_l\}$ de toutes les composantes connexes du graphe G';

5. **Si** $C \notin T$ **alors** rajouter C à T ;

6. **Si** $|E'| \neq 0$ **alors** aller à l'étape 2 ;

7. Retourner la partition P ayant la plus grande modularité

$$P = argmax_{T_i \in T} \{modularité(G, T_i)\} ;$$

Fin

2- Approches basées sur le clustering d'arêtes

Dans ces approches, la détection des arêtes intercommunautaires est basée sur le fait que de telles arêtes sont dans des zones peu clustérisées. Radicchi et al proposent dans [Rad et al. 04] un coefficient (d'ordre g) de clustering d'arêtes. Il est défini comme étant le nombre de cycles de longueur g passant par l'arête divisé par le

nombre total de tels cycles possibles (étant donné les degrés des extrémités de l'arête). Dans ce contexte, cet algorithme retire donc à chaque étape l'arête de plus faible clustering (d'un ordre donné, 3 ou 4 en pratique). Chaque suppression d'arête ne demande alors qu'une mise à jour locale des coefficients de clustering, ce qui lui permet d'être bien plus rapide que l'algorithme précédent. La complexité temporelle totale est en $O(m^2)$. Une autre approche, proposée par Auber et al [Aub et al. 03], utilise un clustering d'arêtes d'ordre 3 pour réaliser un outil de visualisation de graphes.

3- Approche basée sur la centralité d'information

L'algorithme de Fortunato et al., présenté dans [For et al. 04] est en fait une variante de l'algorithme de Girvan et Newman [Gir New 02]. Il est basé sur une autre notion de centralité, à savoir la *centralité d'information*. Les auteurs définissent l'efficacité de communication ϵ_{ij} entre deux sommets i et j du graphe comme étant l'inverse de leur distance (en termes de plus courts chemins). L'efficacité E du réseau est alors définie comme la moyenne des ϵ_{ij} pour tous les couples de sommets. En définitive, on peut dire que la centralité d'information d'une arête est définie comme étant la diminution relative de l'efficacité du réseau lorsque l'on enlève cette arête du graphe. Cette approche fournit de meilleurs résultats que l'approche de Girvan et Newman mais avec une complexité temporelle en $O(m^3 n)$.

C- Approches par optimisation

Plusieurs approches se basent sur l'optimisation et principalement l'optimisation de la modularité réseau pour détecter des communautés disjointes dans les réseaux. Nous citons trois types de méthodes.

Optimisation de la modularité par recuit simulé

Une approche directe d'optimisation de la modularité par recuit simulé a été utilisée dans le cadre de travaux sur la cartographie de réseaux métaboliques. Cette méthode, proposée par Guimerà et Amaraln [Gui Ama 05], prévoit des modifications aléatoires d'une partition, les probabilités de transition sont alors déterminées en fonction de l'amélioration de la modularité. Les transformations opérées sur les partitions sont en fait des fusions et divisions de communautés ainsi que des déplacements de sommets d'une communauté vers une autre.

Approche d'optimisation de la modularité de Duch et Arenas

La méthode de Duch et Arenas [Duc Are 05] commence par scinder le graphe en deux communautés et réalise récursivement des divisions sur les deux communautés ainsi trouvées. En se basant sur le fait que la modularité est additive, la méthode part d'une division choisie arbitrairement et des sommets vont ensuite changer de communauté de façon à augmenter la valeur de la modularité. Le sommet à déplacer est choisi aléatoirement (selon des probabilités adéquates) parmi les sommets ayant les moins bonnes contributions à la modularité globale de la coupe. La coupe obtenant la meilleure modularité tout au long du processus est finalement retenue.

Approches d'optimisation par algorithmes génétiques

Les algorithmes génétiques [Holland 75] ont également été utilisés pour optimiser la modularité réseau. Dans un algorithme génétique standard, on considère un ensemble de solutions candidates à un problème, qui sont codées numériquement comme chromosomes, et une fonction *objectif* pour optimiser l'espace de solutions. La fonction *objectif* joue le rôle de *fitness* biologique pour les chromosomes. L'utilisation de la modularité comme fonction *objectif* à optimiser est une idée qui a été explorée par plusieurs chercheurs. Le but est de trouver, parmi toutes les partitions possibles, celle qui possède la meilleure modularité. Une autre fonction de qualité a été également utilisée. Elle consiste en la détermination d'une mesure globale de la qualité d'une division au sein des communautés appelée *score de la communauté*. Nous décrivons ci-après quatre algorithmes génétiques qui optimisent une fonction de qualité.

1- Algorithme génétique de Tasgin

Dans [Tas et al. 07], Tasgin et al proposent un AG, dans lequel les partitions sont des chromosomes et la modularité est la fonction de *fitness*. Un choix approprié des paramètres de l'algorithme, comme le nombre de chromosomes et les taux de mutation et de crossover a été adopté. Un tableau d'entiers est utilisé pour la représentation des données du problème de la détection de communautés. Le tableau stocke les identifiants communautaires des nœuds, c'est-à-dire à l'entrée i du tableau on trouve l'identifiant de la communauté du nœud i. Le tableau à n éléments est pris comme chromosome pour l'AG. L'algorithme commence avec la création d'une population initiale, en mettant dans la même communauté certains nœuds qui sont

connectés dans le graphe. L'algorithme utilise un *crossover*, une *mutation* et un nouvel opérateur "*clean-up*" visant à réduire le nombre de nœuds mal placés dans des communautés. En effet, une communauté doit contenir plus de liens internes entre les nœuds à l'intérieur de la communauté que de liens externes avec d'autres communautés. Pour cette raison, les voisins d'un nœud doivent être la plupart du temps à l'intérieur d'une même communauté. Ainsi, les auteurs ont défini une métrique nommée "*variance de la communauté d'un nœud i* " qui représente le nombre de communautés différentes entre les voisins et le nœud lui-même. Cette valeur doit être faible pour une bonne structure de la communauté. L'opérateur *clean-up* analyse la variance de communauté de certains nœuds choisis au hasard. Si la variance de communauté du nœud choisi est supérieure à une valeur seuil constante, alors le nœud et tous ses voisins sont mis dans une même communauté. La nouvelle communauté sera la communauté la plus fréquente chez les voisins, autrement dit, la communauté qui contient le plus grand nombre de nœuds dans le voisinage du nœud sélectionné. Si la valeur seuil n'est pas dépassée, aucune opération n'est effectuée sur les identifiants des communautés.

2- Algorithme génétique de Liu et al.

Liu et al. proposent dans [Liu et al. 07] un algorithme dans lequel la modularité maximale de la partition est obtenue par l'intermédiaire de bipartitions successives du graphe. Ainsi, chaque bipartition est calculée en appliquant un algorithme génétique pour chaque sous-graphe (à partir du graphe d'origine lui-même). Chaque sous-graphe est considéré comme isolé du reste du graphe et une bipartition n'est acceptée que si elle augmente la modularité totale du graphe.

3- Algorithme génétique de Pizzuti

Pizzuti propose dans [Pizzuti 08] un algorithme génétique, nommé GA-Net, pour découvrir des communautés dans les réseaux sociaux. Toutes les communautés denses présentes dans la structure du réseau sont obtenues à la fin de l'algorithme en procédant par une exploration sélective de l'espace de recherche, sans avoir besoin de connaître à l'avance le nombre exact de communautés. L'approche introduit deux nouveaux concepts, à savoir, le *score de la communauté* et la notion *d'individu sûr.* Le premier mesure la qualité du partitionnement du réseau en communautés ; l'algorithme cherche alors à l'optimiser. Le second est utilisé pour éviter à l'algorithme génétique d'effectuer des calculs inutiles. Des opérateurs de variation

spécialisés qui génèrent des individus sûrs sont aussi employés. Cette approche a été testée seulement sur de petits réseaux dont les tailles ne dépassent pas 120 nœuds. Son utilisation sur des réseaux complexes n'a pas été rapportée.

4- Algorithme génétique de Lipczak et Milios

Dans [Lip Mil 09], le problème de la détection de communautés dans des réseaux sociaux à grains fins a été abordé. Les auteurs proposent un algorithme de classification ascendante génétique (ACGA). Cet algorithme se base sur le fait que la taille des réseaux sociaux rend difficile à appliquer les approches traditionnelles basées sur le clustering évolutionnaire. Dans les approches standards qui utilisent les AGs pour des problèmes de clustering, chaque individu représente une solution complète. Ainsi, l'algorithme ACGA utilise la représentation d'un individu comme solution au problème de clustering. Une population de clusters évolue de l'état initial, dans lequel chaque cluster représente un utilisateur, vers une solution de clustering en effectuant des fusions entre clusters. Chaque étape du processus d'évolution est réalisée localement, engageant seulement une petite partie du réseau social limité à deux clusters et à leur voisinage direct. L'opérateur de croisement permet à deux clusters d'échanger du matériel génétique améliorant ainsi localement la valeur de la fonction de *fitness*.

D- Autres approches

Approches utilisant des marches aléatoires

Les marches aléatoires dans les graphes sont des procédés stochastiques dans lesquels un marcheur est affecté à un sommet du graphe, et peut à chaque étape se déplacer vers un des sommets voisins. La structure du graphe influe profondément sur le comportement des marches aléatoires. Ainsi plusieurs approches de détection de communautés se basent sur ces comportements. Nous avons déjà mentionné à la sous-section 3.2.1 B, une approche de clustering hiérarchique utilisant les marches aléatoires, nous citons dans ce cadre deux autres approches.

1- Markov Cluster Algorithm

L'approche présentée dans [Dongen 00] se base sur le calcul des probabilités de transition entre tous les sommets du graphe. Une matrice, de transition des marches aléatoires est alors utilisée. Deux opérations matricielles sont utilisées successivement.

La première calcule les probabilités de transition par des marches aléatoires de longueur fixée r et correspond à une élévation de la matrice à la puissance r. Quant à la seconde, elle amplifie les différences en augmentant les transitions les plus plausibles et en diminuant les transitions les moins probables. Les transitions entre sommets d'une même communauté sont alors encouragées. Les itérations successives des deux opérations mènent à une situation limite dans laquelle seules les transitions entre sommets d'une même communauté sont faisables. La complexité temporelle totale de cette méthode est en $O(n^3)$.

2- Approches utilisant le temps moyen pour atteindre un sommet

Afin de mesurer la proximité structurelle de deux sommets, deux approches proposées dans [Zho Lip 04] et [Yen et al. 05] utilisent le nombre moyen d'étapes pour qu'une marche aléatoire parvienne à un sommet donné en partant d'un autre sommet. Cette longueur moyenne peut être calculée grâce à une inversion de matrice en $O(n^3)$. Ces deux approches, utilisent cette quantité pour définir deux distances mesurant la similarité des sommets. Cette distance est utilisée dans la première approche dans une méthode de clustering hiérarchique pour réaliser un algorithme de détection de communautés nommé Netwalk. La deuxième approche l'utilise dans un algorithme de clustering basé sur la *k-means*[13].

Approche par propagation d'étiquettes (LP : Label propagation)

Raghavan et Albert proposent [Rag et al. 07] un algorithme LP pour détecter des communautés. L'idée de base de l'algorithme LP est simple. Au départ, chaque nœud du graphe est initialisé avec une étiquette unique, et à chaque étape, chaque nœud adopte l'étiquette actuelle de la plupart de ses voisins. Dans le cas d'égalité (cas de deux ou plusieurs étiquettes maximales), il choisit une au hasard. Dans ce processus itératif, le groupe de nœuds densément connectés forment un consensus sur une étiquette unique pour former une communauté rapidement.

3.2.2 Communautés disjointes dans des réseaux bipartis

Les réseaux bipartis, par opposition aux réseaux monopartis, appartiennent à une catégorie spéciale de réseaux, où les nœuds peuvent être divisés en deux sous-ensembles disjoints, tels qu'il ne peut y avoir de nœuds joints dans le même sous-

[13] Méthode de clustering par partitionnement connue dans le domaine de Data mining.

ensemble. Les réseaux auteur-publication, les réseaux acteur-film, les réseaux produit-consommation, etc., sont des exemples concrets dans la catégorie des réseaux bipartis. La détection de communautés dans de tels réseaux donne un aperçu sur l'interaction entre les entités correspondantes.

Les algorithmes de détection communautaires conçus pour les réseaux monopartis ne peuvent généralement pas être utilisés pour les réseaux bipartis. Certains auteurs [Wei et al. 11, Liu Mor 09, Gui et al. 07, Barber 07] ont étudié la problématique de détection de communautés dans les réseaux bipartis. Cependant, la notion de communauté dans de tels réseaux n'a pas été abordée de la même manière. En effet, dans [Liu Mor 09, Gui et al. 07], une communauté d'un réseau biparti est composée de nœuds de même type. Par exemple, dans un réseau acteur-événement, les acteurs forment les communautés acteur et les événements forment les communautés événement. Par conséquent, l'algorithme consiste à détecter uniquement des communautés de nœuds d'un même type à la fois. La détection de communautés dans un réseau biparti se fait alors généralement en le projetant sur chaque mode et en trouvant des communautés dans chacune des projections. En revanche, les approches de [Wei et al. 11, Barber 07] considèrent une communauté de réseaux bipartis comme un groupe de nœuds mixtes densément connectés, qui est le point de vue traditionnel dans un réseau monoparti.

A- Approche de Barber

Compte tenu de la dépendance explicite de la modularité du modèle nul (le "null model" défini en section 2.4.1 dans la modularité de Newman), il est clair que le choix spécifique du modèle nul a un impact considérable sur la modularité. Dans Barber [Barber 07], est défini un modèle nul approprié pour les réseaux bipartis, qui est utilisé pour définir une modularité bipartie.

Soit p le nombre de sommets rouge et q le nombre de sommets bleus, ce qui implique $n = p + q$. Sans perte de généralité, supposons que les sommets sont répertoriés de sorte que les sommets rouges soient numérotés de 1, 2,. . . , p et les sommets bleus étiquetés par $p + 1$, $p + 2$,. . . , $p + q$. La matrice d'adjacence A dispose alors d'un bloc hors diagonale sous forme du bloc- matrice A suivant :

$$A = \begin{bmatrix} \boldsymbol{O}_{p\times p} & \tilde{\boldsymbol{A}}_{p\times q} \\ (\tilde{\boldsymbol{A}}^{T})_{q\times p} & \boldsymbol{O}_{q\times q} \end{bmatrix}$$

où $O_{i\times j}$ est la matrice entièrement nulle avec i lignes et j colonnes. D'où la matrice P du modèle nul donnée par :

$$P = \begin{bmatrix} \boldsymbol{O}_{p\times p} & \widetilde{\boldsymbol{P}}_{p\times q} \\ \left(\widetilde{\boldsymbol{P}}^{T}\right)_{q\times p} & \boldsymbol{O}_{q\times q} \end{bmatrix}$$

Cette forme de la matrice P attribue la probabilité nulle d'avoir des arêtes entre les sommets de même couleur, ce qui exclut de telles arêtes dans le modèle nul. Une matrice B est définie par $B = A - P$:

$$B = \begin{bmatrix} \boldsymbol{O}_{p\times p} & \widetilde{\boldsymbol{B}}_{p\times q} \\ \left(\widetilde{\boldsymbol{B}}^{T}\right)_{q\times p} & \boldsymbol{O}_{q\times q} \end{bmatrix}$$

Le modèle nul doit donc être défini. Dans le modèle nul habituel, la probabilité de présence d'une arête entre deux sommets est proportionnelle au produit des degrés des sommets. Dans le cas d'un réseau biparti, cela devient $\widetilde{P}_{ij} = Ck_i d_j$, et cela pour une certaine constante C à déterminer, avec k_i les degrés des sommets rouges et d_j les degrés des sommets bleus : $k_i = \sum_{j=1}^{q} \tilde{A}_{ij}$ et $d_j = \sum_{i=1}^{p} \tilde{A}_{ij}$

En limitant les degrés attendus dans le modèle nul pour les faire correspondre aux degrés réels, on obtient : $\sum_{j=1}^{q} \tilde{P}_{ij} = k_i \quad \sum_{i=1}^{p} \tilde{P}_{ij} = d_j$.

Étant donné que $\sum_{i=1}^{p} k_i = m$ et $\sum_{j=1}^{q} d_j = m$, on a alors : $\tilde{P}_{ij} = \frac{k_i d_j}{m}$ ce qui définit complètement le modèle nul.

La modularité Q du réseau biparti est donc présentée comme la modularité de la matrice B. Certaines propriétés des valeurs propres de B sont identifiées et utilisées pour décrire un algorithme d'identification des modules dans les réseaux bipartis. Il s'agit de l'algorithme BRIM (Bipartite Recursively Induced Modules) qui est un algorithme itératif maximisant la modularité bipartite Q, il est basé sur l'idée que les modules dans les deux parties du réseau sont dépendants, chaque partie étant mutuellement utilisée pour induire les sommets des modules de l'autre partie. En effet, ils supposent que les sommets bleus sont tous affectés à des modules par le biais d'un certain mécanisme. La maximisation de la modularité consiste alors uniquement en l'attribution des sommets rouges aux modules. Il s'agit d'une tâche relativement simple. A partir de n'importe quelle partition des sommets rouges par exemple, il est facile d'identifier la partition optimale des sommets bleus. A partir de là, on procède à l'optimisation de la partition des sommets rouges, et ainsi de suite (d'un mode à

l'autre). Pour chaque itération, la modularité Q est garantie ne jamais diminuer, car il est au moins possible de garder la modularité obtenue et maintenir le précédent partitionnement des sommets. Par conséquent, l'algorithme BRIM trouvera toujours une partition avec une valeur maximale pour Q. En général, la partition identifiée correspond à un maximum local de Q et pas un maximum global.

B- Approche de Liu et Murata

Dans [Liu Mor 09], les auteurs proposent un algorithme rapide appelé LP & BRIM pour la détection de communautés dans les grands réseaux bipartis. Celui-ci est basé sur une stratégie utilisant conjointement les deux algorithmes LP et BRIM. L'algorithme LP de propagation d'étiquettes de Raghavan et Albert [Rag et al. 07] (cf. section 3.5.1 D), qui est un algorithme très rapide de détection de communautés, est utilisé pour effectuer la recherche d'un partitionnement initial. L'algorithme BRIM [Barber 07] (vu dans la section précédente), est utilisé pour générer récursivement une meilleure structure communautaire en induisant des divisions entre les deux types de nœuds dans les réseaux bipartis. Le pseudo-code de l'algorithme LP & BRIM est donné dans l'algorithme 3.2.

Algorithme 3.2 - LP & BRIM

Entrée : réseau biparti G
Sortie : division en communautés

Fonction

départ de chaque nœud rouge avec un label unique ;
Tant que (chaque nœud a une étiquette qui est le label maximum de ses voisins)
faire
propager les étiquettes de nœuds rouges aux nœuds bleus ;
propager les étiquettes des nœuds bleus aux nœuds rouges ;
Fin tant que
$division_{LP}$ = la division correspondant à la plus grande Bipartite_modularity dans la boucle tant que ci-dessus ;
$division_{rouge}$ = la division des nœuds rouges dans $division_{LP}$;
$division_{bleu}$ = la division des nœuds bleus dans $division_{LP}$;
induire-Drapeau-Bleu = vrai ;
Tant que (induire-Drapeau-Bleu)
faire
preBipartitemodularity = Bipartitemodularity ($division_{rouge}$, $division_{bleu}$) ;
if (induire-Drapeau-Bleu == vrai)
induire $division_{bleu}$ de $division_{rouge}$;
sinon
Induire $division_{rouge}$ de $division_{bleu}$;
newBipartitemodularity = Bipartitemodularity ($division_{rouge}$, $division_{bleu}$) ;
if (newBipartitemodularity == preBipartitemodularity)
break ;
sinon
preBipartitemodularity = newBipartitemodularity ;
induire-Drapeau-Bleu = ! induire-Drapeau-Bleu ;
Fin tant que
division = ($division_{rouge}$, $division_{bleu}$) ;

Fin fonction

C- Approche de Zhan et al.

Le problème d'identification des communautés mixtes dans des réseaux bipartis a été étudié dans [Wei et al. 11], où une version modifiée de l'algorithme génétique adaptatif (MMOGA) basée seulement sur l'opérateur de mutation (MOGA : Mutation-Only Genetic Algorithm) a été présenté. Cet algorithme est exempt de paramètres à la différence des algorithmes génétiques classiques. Le seul paramètre nécessaire à fournir dans MMOGA est la taille de la population. Ce paramètre peut

avoir une influence importante sur l'application des algorithmes génétiques. Avec de petites populations, l'algorithme de recherche cible une plus petite région de manière à converger rapidement, mais termine par les plus mauvaises solutions. Alors qu'avec de plus grandes populations, il a une capacité de recherche plus puissante pour avoir de meilleures solutions, mais exigent énormément de temps. L'efficacité de MMOGA est basée sur les trois améliorations apportées. En premier lieu, l'introduction d'une mesure différente de l'informativité d'un locus[14] au lieu de l'écart type, permet de déterminer précisément la mutation à utiliser. Cette mesure est un compromis entre la distribution d'un locus sur la population actuelle et la distribution uniforme du locus. Par ailleurs, une technique de réaffectation a été développée pour différencier l'état atteint d'informativité d'un locus de l'état aléatoire lors de la phase initiale. En outre, les auteurs ont présenté une règle de mutation modifiée qui incorpore des opérations connexes pouvant garantir la convergence de MMOGA vers l'optimum global et peut accélérer le processus de convergence.

3.3 Détection de communautés chevauchantes

Dans les réseaux réels, de nombreuses communautés peuvent se chevaucher comme c'est le cas par exemple dans les réseaux sociaux où les individus sont naturellement caractérisés par des appartenances à plusieurs communautés. Ainsi, une personne a habituellement des connexions à plusieurs groupes sociaux : la famille, les amis et collègues, un chercheur peut être actif dans plusieurs domaines. Particulièrement, dans les réseaux sociaux en ligne, le nombre de communautés auxquelles un individu peut appartenir est pratiquement illimité, car une personne peut simultanément s'associer avec plusieurs groupes comme elle le désire. Dans [Kel et al. 11], les auteurs ont montré que le chevauchement est en effet une caractéristique importante de nombreux réseaux sociaux du monde réel. Cela se produit aussi dans d'autres réseaux complexes tels que les réseaux biologiques, où un nœud peut avoir plusieurs fonctions. La détection des communautés chevauchantes peut donner un aperçu des structures et des fonctions des réseaux.

Pour cette raison, il y a un intérêt croissant ces dernières années pour les algorithmes de détection de communautés chevauchantes, qui permettent d'identifier un ensemble de communautés qui ne sont pas nécessairement disjointes. Il pourrait y avoir des nœuds qui peuvent appartenir à plusieurs communautés.

[14] *Un locus* est un emplacement physique précis et invariable sur un chromosome.

Dans la section suivante, nous présentons les algorithmes de détection de communautés chevauchantes qui ont reçu le plus d'attention de la part de la communauté scientifique. Les algorithmes sont rassemblés selon le type de réseaux pour lesquels ils sont conçus : réseaux monopartis ou bipartis, et à la façon avec laquelle les communautés sont découvertes.

3.3.1 Communautés chevauchantes dans des réseaux monopartis

Plusieurs méthodes ont été proposées pour détecter des communautés chevauchantes dans les réseaux monopartis. Nous les décrivons ci-après en mettant l'accent sur leurs principales caractéristiques.

A- Clique percolation

La méthode de clique percolation (CPM : Clique Percolation Method) est basée sur l'hypothèse selon laquelle une communauté se compose d'un ensemble de sous-graphes complets chevauchants, et détecte les communautés par la recherche de cliques adjacentes. La CPM construit donc des communautés de *k*-cliques, qui correspondent aux graphes complets de *k* sommets. Elle commence donc par identifier toutes les cliques de taille *k* dans un réseau. Une fois qu'elles ont été identifiées, un nouveau graphe est construit de telle sorte que chaque sommet représente une de ces *k*-cliques. Deux nœuds sont alors connectés si les *k* cliques les représentant partagent *k-1* membres. Des composants connectés dans le nouveau graphe déterminent quelles cliques composent les communautés. Puisque un sommet peut être dans plusieurs *k*-cliques en même temps, le chevauchement entre les communautés est possible. CPM est adaptée aux réseaux denses. Empiriquement, les petites valeurs de *k* (précisément entre 3 et 6) se sont révélées intéressantes [Pal et al. 05, Lan et al. 09, Gregory 10]. L'algorithme CFinder est une implémentation de CPM, sa complexité temporelle est par ailleurs polynomiale dans de nombreuses applications [Pal et al. 05]. Cependant, il échoue pour de nombreux grands réseaux sociaux.

Dans [Far et al. 07], un algorithme CPMw a été présenté, dans lequel a été introduit un seuil d'intensité pour des sous-graphes dans des réseaux pondérés. Seuls les *k*-cliques avec une intensité supérieure à un seuil fixe sont inclus dans une

communauté. Au lieu de traiter toutes les valeurs de k, l'algorithme SCP [Kum et al. 08] trouve des communautés de clique de taille donnée. Dans la première phase, SCP détecte les k-cliques en vérifiant toutes les (k-2)-cliques dans les voisins communs des deux points d'extrémité où les liens sont insérés au réseau séquentiellement dans l'ordre décroissant de poids. Dans la deuxième phase, la k-communauté est détectée par la recherche des composants connectés dans la (k-1)-clique projection de la représentation bipartie, dans laquelle un type des nœuds, représente une k-clique et l'autre représente une (k-1)-clique. Comme chaque k-clique est traitée exactement deux fois, le temps d'exécution augmente linéairement en fonction du nombre de cliques. SCP permet des seuils de poids multiples en une seule fois et est plus rapide que le CMP.

En dépit de leur simplicité conceptuelle, on peut dire que les algorithmes CMP ressemblent plus à l'appariement de formes plutôt qu'à trouver des communautés, puisqu'ils visent à trouver une structure spécifique localisée dans un réseau.

B- Graphe de liens et partitionnement des liens

L'idée de partitionner les liens au lieu des nœuds pour découvrir la structure des communautés a également été envisagée. Un nœud dans le graphe original est dit chevauchant si les liens qui y sont connectés sont mis dans plus d'une communauté.

Dans [Ahn et al. 10], les liens sont partitionnés par un algorithme de clustering hiérarchique des arêtes. Étant donné une paire de liens e_{ik} et e_{jk} incidents à un nœud k, une similarité peut être calculée par l'indice de Jaccard défini comme

$$S(e_{ik}, e_{jk}) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$$

où $N(i)$ représente le voisinage du nœud i, y compris i (dans cette définition de voisinage). Le clustering hiérarchique avec Single-linkage est ensuite utilisé pour construire un dendrogramme des liens. La coupe de ce dendrogramme à un certain seuil conduit aux communautés de liens. La complexité en temps est en $O(nk_{max}^{2})$, où k_{max} est le degré maximum de nœuds dans le réseau.

Les auteurs Evans et al, dans [Eva Lam 09], projettent le réseau en un graphe de liens pondéré, dont les nœuds sont les liens du graphe originel. Ensuite, des algorithmes de détection de communautés disjointes peuvent être appliqués. La

partition des nœuds d'un graphe de liens conduit à une partition d'arêtes du graphe original. Cette dernière définit des communautés chevauchantes de nœuds.

C- Extension locale et optimisation

Des algorithmes utilisant une extension locale et l'optimisation sont basés sur la croissance d'une communauté naturelle [Lan et al. 09] ou d'une communauté partielle. La plupart d'entre eux comptent sur une fonction de gain local qui caractérise la qualité à forte densité des groupes de nœuds connectés.

Une des premières méthodes pour trouver les communautés chevauchantes a été conçue par Baumes et al. [Bau et al. 05]. Les auteurs ont proposé un processus en deux étapes. Tout d'abord, l'algorithme RankRemoval est utilisé pour classer les nœuds suivant un critère. Ensuite, le processus élimine itérativement les nœuds hautement cotés jusqu'à ce que de petits noyaux de clusters disjoints soient formés. Ces noyaux servent de semences pour les communautés dans la deuxième étape du processus. L'algorithme Itératif Scan (IS), qui étend les noyaux en ajoutant ou en supprimant des nœuds jusqu'à ce qu'une fonction de densité locale ne puisse plus être améliorée. La fonction de densité proposée peut être formellement donnée comme suit :

$$f(c) = \frac{w_{in}^{C}}{w_{in}^{C} + w_{out}^{C}}$$

où w_{in}^{C} et w_{out}^{C} sont les poids totaux interne et externe de la communauté C. Dans le pire des cas, le temps d'exécution est en $O(n^2)$. La qualité des communautés découvertes dépend de la qualité des semences. Comme l'algorithme permet de supprimer des sommets lors de l'expansion, IS peut produire des composants déconnectés dans certains cas. Pour cette raison, une version modifiée appelée CIS [Kelley 09] a été proposée. Dans CIS, la connexité est vérifiée après chaque itération. Dans le cas où la communauté est divisée en plus d'une partie, seule celle qui a la plus grande densité est maintenue. CIS développe également une nouvelle fonction de *fitness f* qui intègre la probabilité arête e_p :

$$f(c) = \frac{w_{in}^{C}}{w_{in}^{C} + w_{out}^{C}} + \lambda e_p.$$

Le paramètre λ contrôle la façon dont l'algorithme se comporte dans les zones creuses du réseau. L'ajout d'un nœud doit trouver un équilibre entre la variation de la densité des degrés internes et la variation de la densité des arêtes.

Dans [Lan et al. 09], l'algorithme LFM élargit une communauté à partir d'un nœud de départ aléatoire pour former une communauté naturelle jusqu'à ce que la fonction de *fitness f* soit localement maximale :

$$f(c) = \frac{k_{in}^{C}}{(k_{in}^{C} + k_{out}^{C})^{\alpha}}$$

où k_{in}^{C} et k_{out}^{C} sont les degrés totaux interne et externe de la communauté C, et α est le paramètre de résolution contrôlant la taille des communautés. Après avoir trouvé une communauté, LFM choisit aléatoirement un autre nœud non encore affecté à une communauté pour agrandir une nouvelle communauté. LFM dépend largement du paramètre de résolution α. La complexité de calcul pour une valeur fixe de α est d'environ $O(n_c s^2)$, où n_c est le nombre de communautés et s est la taille moyenne des communautés. La complexité dans le pire des cas est en $O(n^2)$.

D- Propagation d'étiquettes (Label propagation)

L'algorithme de propagation d'étiquettes [Rag et al. 07] (cf. section 3.2.1 D), dans lequel les nœuds ayant les mêmes étiquettes forment une communauté, a été étendu à la détection communauté chevauchantes en permettant à un nœud d'avoir plusieurs étiquettes.

Dans COPRA [Gregory 10], l'auteur propose un algorithme pour trouver la structure des communautés chevauchantes dans de très grands réseaux. La communauté est ici prise comme constituée de sommets d'un même type à la fois, l'algorithme trouve ainsi des communautés pour chaque type de nœuds. Comme l'algorithme original, les sommets ont des étiquettes qui se propagent entre les sommets voisins afin que les membres d'une communauté parviennent à un consensus sur leur appartenance communautaire. Leur contribution se résume en l'extension de l'étape de propagation d'étiquettes. En effet, ils incluent des informations sur plus d'une communauté à chaque nœud: chaque sommet peut désormais appartenir jusqu'à v communautés, où v est le paramètre de l'algorithme qui est utilisé pour contrôler le nombre maximal de communautés auxquelles un nœud peut s'associer. La complexité en temps est en $O(vm\ log\ (vm/n))$ par itération.

E- Autres méthodes

Dans [Gregory 07], l'algorithme CONGA élargit l'algorithme de clustering descendant de Girvan et Newman [Gir New 02] en permettant à un nœud de se scinder en plusieurs exemplaires. Les deux scissions de l'intermédiarité (définie par le nombre de chemins les plus courts) sur l'arête imaginaire, et la scission de l'intermédiarité conventionnelle sont prises en considération. CONGA hérite de la grande complexité de calcul de GN (cf. algorithme 3.1).

Dans une version plus raffinée, CONGO [Gregory 08] utilise la scission de l'intermédiarité locale afin d'optimiser le temps d'exécution. Gregory [Gregory 09] a également proposé d'appliquer des algorithmes de détection de communautés disjointes sur le réseau, produit en fractionnant le nœud en plusieurs copies, en utilisant la scission de l'intermédiarité.

Zhang et al. [Zha et al. 09] ont proposé un processus itératif qui renforce la topologie du réseau. Il s'agit de la "*proximité*" qui est interprétée comme étant la probabilité pour qu'une paire de nœuds appartiennent à la même communauté. La proximité entre deux sommets est définie comme étant la somme du nombre de liens directs, du nombre de voisins communs et du nombre de liens dans le voisinage commun. Étant donné la topologie, la proximité est calculée. La proximité au-dessus d'un certain seuil est alors utilisée pour redistribuer les liens et mettre à jour la topologie. Si la proximité est grande, un lien est ajouté au réseau, sinon, le lien est retiré. La proximité peut être utilisée pour effectuer le regroupement micro sur chaque sommet afin de permettre le chevauchement.

3.3.2 Communautés chevauchantes dans des réseaux bipartis

La détection de communautés chevauchantes dans les réseaux bipartis n'a pas reçu beaucoup d'attention, et par conséquent, peu de travaux ont été réalisés dans cette catégorie.

Dans [Nan et al. 08], les auteurs proposent un algorithme qui adopte une stratégie d'optimisation locale. Autrement dit, au lieu de diviser un réseau dans ses zones les plus faiblement connectées, l'algorithme identifie plutôt des groupes de bicliques, c'est-à-dire, des communautés basées sur les zones les plus densément connectées. Chaque groupe de bicliques ayant un chevauchement maximal sera pris comme centroïde d'un clustering. Autour de chaque centroïde, l'algorithme construit des communautés d'une manière progressive en augmentant les valeurs de certaines

métriques jusqu'à ce que, chaque sommet du réseau apparaisse dans au moins une communauté.

Gregory dans [Gregory 10] a proposé l'algorithme COPRA (Community Overlap PRopagation Algorithm) (cf. section 3.3.1 D) pour détecter des communautés chevauchantes dans des réseaux monopartis, qui est une généralisation de l'algorithme basé sur la technique de propagation d'étiquette de Raghavan et al. [Rag et al. 07]. Gregory a aussi adapté COPRA pour la détection de communautés chevauchantes dans des réseaux bipartis. En effet, il a réalisé un changement au niveau de l'algorithme COPRA comme suit.
Soient Vr et Vb les deux sous ensembles de sommets du graphe G, de telle sorte que $V = Vr \cup Vb$, $Vr \cap Vb = \varnothing$ et $|Vr| \geqslant |Vb|$. L'auteur a modifié une étape de son algorithme de telle sorte que, à chaque itération, les étiquettes des sommets soient propagées d'abord du grand ensemble Vr vers l'ensemble Vb, puis les nouvelles valeurs des étiquettes de l'ensemble Vb sont propagées vers l'ensemble Vr. Signalons que la propagation des communautés d'un mode à l'autre est également une caractéristique de l'algorithme de détection de communautés dans des réseaux bipartis de Barber [Barber 07] (cf. section 3.2.2 A).

3.4 Conclusion

Dans ce chapitre, nous avons proposé un bref aperçu de l'art des travaux traitant de la détection de communautés disjointes et de la détection de communautés chevauchantes dans les deux types de réseaux : uniparti et biparti. Dans les réseaux bipartis, habituellement la recherche de communautés se fait d'abord en projetant le réseau sur chaque mode, ensuite on essaie de localiser les communautés dans chacune des projections. Cela est nécessairement moins efficace que la détection des communautés dans le réseau d'origine biparti, car une partie de l'information est perdue lors de la projection. Ceci est clair du fait que nous ne pouvons pas reconstruire le réseau à partir des projections. Un autre désavantage est que généralement les communautés détectées dans chacune des projections ne sont pas cohérentes entre elles. Une analyse directe du réseau biparti est une option plus naturelle, qui capte les nuances importantes de la structure du réseau qui sont invisibles aux analyses fondées sur des projections. Par ailleurs, nous déduisons de l'étude faite dans ce chapitre qu'il n'y a que très peu de travaux qui considèrent la détection de communautés chevauchantes dans des réseaux bipartis. Ce qui constitue

un axe de recherche pratiquement vierge, qui doit répondre sans doute à des attentes considérables dans ce domaine. Ainsi, notre choix de trouver une méthode de détection de communautés chevauchantes mixtes dans des réseaux bipartis est tout à fait justifié.

4

Double optimisation de la modularité pour la détection de communautés chevauchantes

4.1 Introduction

La grande force des algorithmes évolutionnaires consiste en leur capacité à trouver la zone de l'espace des solutions contenant l'optimum de la fonction à optimiser. Ils deviennent, cependant, infructueux lorsqu'il s'agit de chercher la valeur exacte de l'optimum dans cette zone. Néanmoins, d'autres algorithmes permettent de réaliser au mieux cette tâche. Ces derniers forment la famille des algorithmes dits "d'optimisation locale". Il est donc naturel de penser à associer un algorithme d'optimisation locale à un algorithme évolutionnaire de manière à approcher la valeur exacte de l'optimum. On peut donc appliquer un algorithme d'optimisation locale sur le meilleur élément produit par un algorithme évolutionnaire. En effet, l'association "*algorithme évolutionnaire – méthode locale*" est pratiquement une exigence. Les deux méthodes sont complémentaires et agissent successivement l'une après l'autre à des phases différentes du processus d'optimisation. L'algorithme évolutionnaire permet de faire alors "disparaître" la combinatoire du problème, laissant ainsi le champ libre à la méthode locale. Cette dernière est appliquée dans des zones connexes susceptibles de contenir l'optimum global. L'approche que nous proposons pour la découverte de communautés chevauchantes mixtes dans des réseaux bipartis

est basée sur une double optimisation de la modularité [Sou et al. 13]. L'association d'une optimisation globale et d'une optimisation locale est une manière qui permet d'approcher au mieux la valeur exacte de l'optimum.

Le présent chapitre est consacré à la description de l'approche que nous proposons dans cette thèse, il est organisé comme suit. Nous présentons tout d'abord le schéma général de l'approche proposée. Nous détaillons par la suite les quatre parties constituant notre approche, en commençant par l'algorithme évolutionnaire dédié à la détection de communautés disjointes. Ensuite, nous explicitons clairement la procédure de passage d'un graphe biparti au graphe-arête associé. Enfin, nous décrivons les différents points de notre algorithme d'optimisation locale pour obtenir des communautés chevauchantes significatives. Nous montrerons au passage comment adapter la fonction de qualité pour quantifier le découpage de l'ensemble des nœuds de réseaux bipartis en communautés recouvrantes.

4.2 Présentation générale de l'approche

Les méthodes existantes de partitionnement de graphes permettent de répartir les sommets d'un graphe en communautés disjointes. Nous proposons une nouvelle approche de détection de communautés chevauchantes dans un graphe biparti non orienté et non pondéré, sur la base d'un partitionnement de ses arêtes. Nous abordons ainsi le problème de détection de communautés chevauchantes dans un réseau biparti comme nous l'aurions fait avec un problème de détection de communautés disjointes dans un réseau monoparti. Notre méthode s'applique aux graphes bipartis où la densité globale est faible, mais présente toutefois des zones de forte densité constituées d'un nombre de nœuds relativement réduit. Dans un tel graphe, le partitionnement des arêtes n'augmente pas la complexité du problème puisque le nombre d'arêtes est proportionnel au nombre de sommets du graphe, i.e., $O(\textit{nombre d'arêtes}) = O(\textit{nombre de sommets})$. De ce fait, considérer les arêtes au lieu des sommets, n'affecte guère la complexité algorithmique de détection de communautés (cf. sous-section 4.4.5).

Un graphe biparti peut être vu comme un graphe où il est possible de colorier les sommets avec deux couleurs différentes sans jamais colorier deux sommets adjacents avec la même couleur. On définit ainsi deux classes de sommets : l'une nommée rouge et l'autre nommée bleue.

Dans ce contexte, nous considérons un graphe biparti non orienté et non pondéré $G(V_r \cup V_b, E)$ où V_r et V_b sont respectivement les ensembles de sommets rouges et bleus, et E est l'ensemble des arêtes. Pour simplifier, on note $r = |V_r|$, $b = |V_b|$, $m = |E|$ et d_i le degré du sommet i. On a également $n = |V| = r + b$ qui est le nombre total de sommets dans le graphe.

Définition 4.1. (Communauté mixte). Une communauté mixte dans un réseau (graphe) biparti est une communauté pouvant contenir des nœuds (sommets) appartenant aux deux classes de nœuds du réseau biparti.

L'idée est de partitionner l'ensemble E des arêtes du graphe biparti en groupes disjoints (ou clusters). En associant à chaque cluster, tous les sommets liés par les arêtes du cluster, on obtient ainsi une décomposition des sommets du graphe en groupes chevauchants (ou communautés chevauchantes).

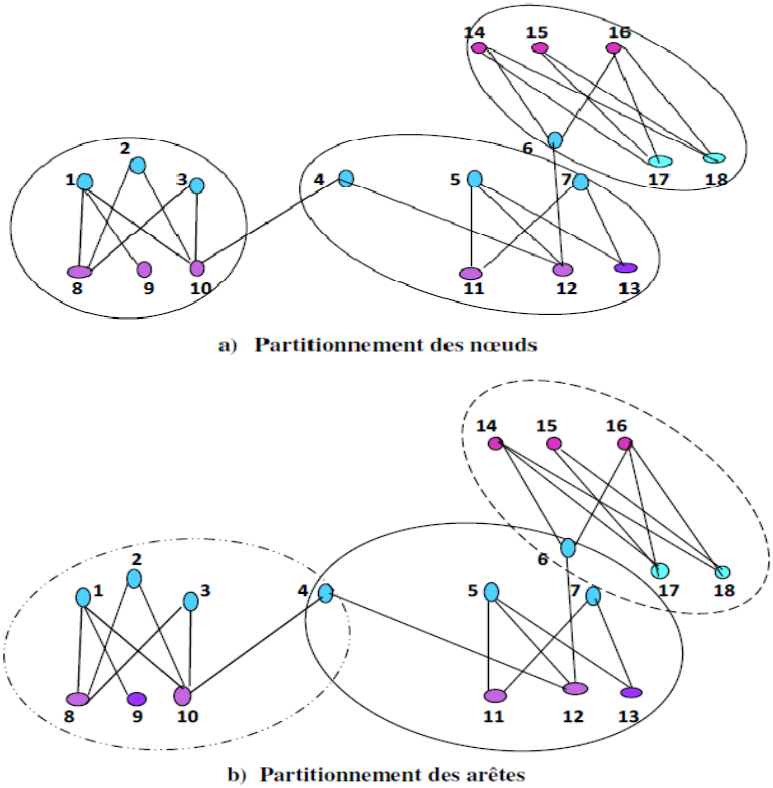


Fig 4.1. Deux catégories de partitionnement de réseaux

Cette décomposition des sommets va être améliorée en optimisant la modularité chevauchante. Dans la figure 4.1, pour le cas *a)Partitionnement des sommets*, les sommets 4 et 6 appartiennent à une seule communauté, alors que le sommet 4 fait partie de deux communautés qu'on peut mettre en évidence par le partitionnement

de l'ensemble des arêtes du graphe, comme illustré par *b) Partitionnement des arêtes*. Le sommet 6, en revanche, ne peut pas être considéré comme un sommet qui chevauche entre 2 communautés, puisqu'il est faiblement connecté aux autres communautés. Nous devons, par conséquent, filtrer les sommets en commun pour déduire ceux qui chevauchent réellement.

Comme illustré par la figure 4.1, le sommet 4 appartient effectivement à 2 communautés selon le partitionnement des arêtes : {1, 2, 3, 4, 8, 9 ,10} et {4, 5, 7, 11, 12, 13}. Cependant, le sommet 6 qui est mis en évidence par ce partitionnement, n'est pas un sommet qui chevauche entre les 2 communautés {4, 5, 6, 7, 11, 12, 13} et {6, 14, 15, 16, 17, 18} ; intuitivement, il appartient plus au deuxième groupe qu'au premier (cf. section 4.5.3).

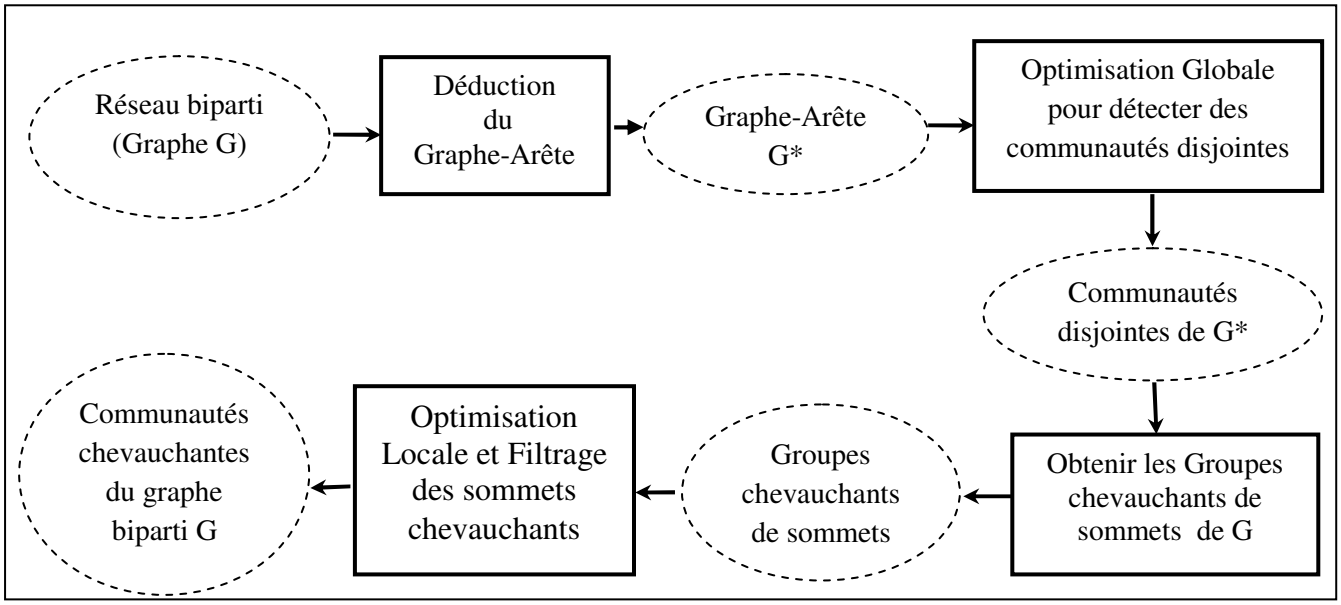


Fig 4.2. Schéma général de l'approche proposée dans [Sou et al. 13]

En vertu de ce qui a été déjà annoncé précédemment, notre approche consiste en quatre grandes étapes schématiquement illustrées par la figure 4.2. Ces différentes étapes sont clairement explicitées par les points suivants :

1. Construction du graphe-arête $G*$;
2. Détection de communautés disjointes dans le graphe $G*$;
3. Déduction des groupes de sommets du graphe biparti initial G. Plus précisément, nous établissons des groupes chevauchants de sommets de G, en

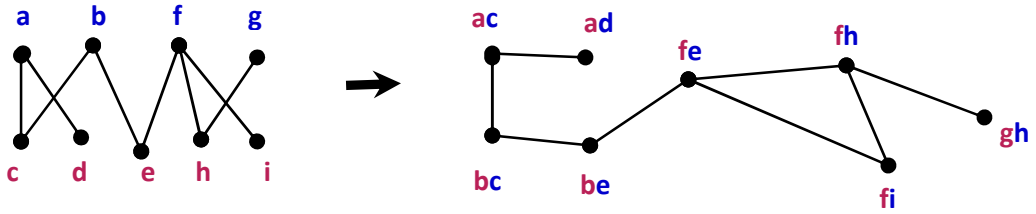
utilisant les groupes disjoints de sommets de G^*, obtenus précédemment à l'étape 2, selon ce qui est indiqué sur la figure 4.2.

4. Afin de déterminer des communautés mixtes chevauchantes dans le graphe biparti G, nous optimisons localement la modularité chevauchante des groupes de sommets obtenus précédemment à l'étape 3. Toutefois, pour obtenir des communautés significatives, nous terminons le processus par une étape de filtrage des nœuds apparaissant simultanément dans plusieurs groupes.

4.3. Construction du graphe-arête (Line Graph)

Le partitionnement de l'ensemble des arêtes E est défini par l'ensemble P= $\{C_1, \dots C_{Nc}\}$ de Nc clusters tel que $\forall i \neq j,\ C_i \cap C_j = \emptyset$ et $\bigcup_i C_i = E$, pour i =1,...Nc.

Le but du partitionnement de l'ensemble E, est d'obtenir la meilleure décomposition de l'ensemble des sommets $V_r \cup V_b$ en communautés chevauchantes. Le partitionnement de E nécessite la construction du graphe G^* comme illustré par la figure 4.3. Les sommets de G^* représentent les numéros des arêtes de G, tandis que ses arêtes représentent l'adjacence des arêtes de G.



a) Graphe biparti originel **b)** Graphe-arête

Fig 4.3. Passage du graphe d'origine au graphe-arête associé

Pour expliciter la procédure de passage du graphe d'origine G au graphe-arête G^*, on définit un cadre formel adéquat.
Soit un graphe G (V, E) non orienté et non pondéré complètement défini par sa matrice d'adjacence A comme suit :

$$A[i,j] = \left\{ \begin{array}{cc} 1 & \text{si } e(i,j) \in E \\ 0 & \text{sinon} \end{array} \right\}$$

où e (i, j) représente l'arête liant les sommets i et j de G.

Dans le cas où G est un graphe biparti, on a r sommets rouges et b sommets bleus tel que $|V| = r + b$. On suppose que ces sommets soient répertoriés de sorte que les

sommets rouges soient numérotés de 1, 2, ..., r et les sommets bleus étiquetés par $r + 1$, $r + 2$, . . ., $r + b$. La matrice d'adjacence du graphe biparti dispose alors de sous-matrices hors diagonale sous la forme suivante :

$$A = \begin{bmatrix} O_{r\times r} & A_{r\times b} \\ (A^T)_{b\times r} & O_{b\times b} \end{bmatrix}$$

où $O_{i\times j}$ est la matrice entièrement nulle (matrice constituée entièrement de zéros), avec i lignes et j colonnes.

On génère le graphe-arête monoparti $G^*(V^*, E^*)$ correspondant. Rappelons que le nombre de sommets de G^* est $|V^*| = |E| = m$.
On construit le graphe G^*, selon les étapes suivantes :

- On modifie d'abord la matrice d'adjacence A en la parcourant ligne par ligne, en numérotant au passage de 1 à m les arêtes du graphe biparti dans cette même matrice, comme indiqué dans l'équation 4.1.

$$A[i,j] = \begin{cases} k & \text{si } A(i,\ j) \neq 0 \\ 0 & \text{sinon} \end{cases} \quad \text{pour } k = 1..m \tag{4.1}$$

- Par définition du graphe-arête G^*, la matrice d'adjacence associée $A^*[m, m]$ donne les arêtes qui sont incidentes entre elles, c'est-à-dire :

$$A^*[i,j] = \begin{cases} 1 & \text{si l'arete } i \text{ est incidente à l'arête } j \\ 0 & \text{sinon} \end{cases} \tag{4.2}$$

Ainsi, les arêtes dont les numéros se retrouvent dans une même ligne de la matrice A obtenue avec l'équation 4.1, ont une extrémité commune, donc, elles sont incidentes deux à deux (de même pour les numéros d'arêtes de la même colonne). En conséquence, le parcours de la matrice A permet de déduire les arêtes qui sont incidentes deux à deux, et déterminer par la même, la matrice d'adjacence A^*. L'exemple suivant illustre le passage de la matrice A à la matrice A^*.

Exemple :
Soit un graphe biparti G défini par la matrice d'adjacence A (6, 6) suivante :

$$A = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

On considère ainsi juste la sous-matrice $A = \begin{matrix} 4 & 5 & 6 \end{matrix}$
$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Le graphe arête G* possède 5 arêtes, numérotées de 1 à 5, ligne par ligne, dans la matrice A :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 0 \\ 0 & 0 & 5 \end{bmatrix}.$$

La matrice d'adjacence $A^*[5, 5]$ du graphe arête G^* est donc

$$A^* = \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix}
\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Ainsi, nous obtenons le graphe-arête G^* monoparti, représentant l'incidence des arêtes de G. L'étape suivante consiste à trouver une bonne partition des sommets de G^* qui est également, une bonne partition des arêtes de G. Autrement dit, nous développons un algorithme évolutionnaire optimisant la modularité de Newman [New Gir 04].

4.4 Méthode évolutionnaire de partitionnement

Les algorithmes évolutionnaires (AEs) sont des heuristiques basées sur la théorie de l'évolution de Darwin. Ils définissent une approche dynamique et adaptative permettant d'explorer une population de solutions. Une solution possible, c'est-à-dire, un membre de la population, est appelé individu. Chaque itération d'un AE implique une sélection compétitive qui exclut progressivement les mauvaises solutions à travers l'évaluation d'une valeur de *fitness* qui indique la qualité d'un individu en tant que solution du problème. Par ailleurs, un ensemble d'opérateurs génétiques comme M*utation*, *Crossover*, etc., sont appliqués à chaque génération pour créer de nouveaux individus qui viendront remplacer toute ou partie de la population.

Afin d'optimiser globalement la modularité réseau de Newman Q du graphe G^* (cf. section 2.4.1 C), nous utilisons un algorithme génétique. Notre algorithme se base sur deux idées principales qui sont la phase d'initialisation des individus et l'évolution de la population par des opérateurs spécialisés [Sou et al. 13]. En effet

chaque individu est initialisé adroitement (cf. sous-section 4.4.3 initialisation) pour éliminer les itérations inutiles. La population, quant à elle, doit permettre l'exploitation des connaissances acquises au cours des processus d'initialisation et d'évolution. Pour cela la stratégie de mise à jour des individus de la population ne doit pas être globale mais doit s'établir plutôt individu par individu en utilisant une certaine connaissance.

En s'appuyant sur ces concepts l'algorithme génétique à état stationnaire (SSGA = Steady State Genetic Algorithm) est le plus adapté. Par ailleurs, l'efficacité des AEs peut être augmentée substantiellement si l'on intègre dans leur utilisation un mécanisme inductif qui correspond à une certaine représentation de leur environnement, de sorte à orienter leur exploration. Cette constatation est à la base de notre AE. Nous l'explicitons dans la section suivante avec les nouveaux opérateurs conçus à cet effet.

4.4.1 Algorithme évolutionnaire

Soit G (V, E) un graphe à partitionner, représenté dans une structure de données adaptée à l'algorithme génétique. Les sommets du graphe sont numérotés de 1 à $|V|$ et placés initialement aléatoirement dans des clusters. On prend alors au plus un nombre de clusters égal à $(|V|)/2$. Ce nombre est appelé à décroitre au cours de l'exécution de l'algorithme. Une population de μ solutions au problème est utilisée. La fonction *objectif* modularité Q de Newman définie précédemment est prise comme fonction de *fitness* pour évaluer chaque individu. Nous modifions l'algorithme génétique standard pour satisfaire les besoins de notre approche. En bref, nous utilisons un *Crossover*, une *Mutation* et deux nouveaux opérateurs *Decrease* et *Increase*. Nous ajoutons quelques étapes supplémentaires lors de la création de la population initiale afin d'améliorer la convergence de l'algorithme (cf. sous-section 4.4.3 paragraphe initialisation).

Après la création de la population initiale de μ individus, nous réalisons des opérations génétiques pour un certain nombre d'itérations *Gen_ max.* Au cours de chaque itération, on sélectionne deux parents pour réaliser le *Crossover* et obtenir un fils unique. Une mutation sera appliquée à ce fils avec une certaine probabilité. Ce nouvel individu fils, remplacera le plus mauvais individu de la population courante. Les opérateurs *Decrease* et *Increase* sont appliqués à chacun des parents avant leurs réinjection dans la population. Afin de préserver la meilleure solution rencontrée lors

du processus évolutif, nous utilisons la stratégie élitiste. A ce propos, à chaque itération, le meilleur membre est sauvegardé pour être réinjecté dans la population de l'itération suivante. De cette manière nous garantissons la survie du meilleur chromosome pour les générations suivantes. Un pseudo-code est donné dans l'algorithme 4.1 :

Algorithme 4.1- Adaptation du Steady State Genetic Algorithm

Entree : Pm : probabilité d'appliquer la mutation,
Pc : probabilité d'appliquer le croisement,
Gen_Max ;
Sortie : BestIndividual // *Meilleur individu issu du processus d'évolution*
Debut
- POP←Generate_initial_Population ;
- Evaluer (POP) ;
Pour Gen := 1 to Gen_Max **faire**
- $P1$ ←SélectFrom(POP) ; // par tournoi
- $P2$ ←SélectFrom(POP) ;
- **Si** *fitness*($P1$) > fitness($P2$)
Alors début Emetteur ←$P1$; Récepteur ←$P2$ **fin** ;
Sinon début Emetteur ←$P2$; Récepteur ←$P1$ **fin** ;
- Enfant ←Crossover (Emetteur, Récepteur, Pc) ;
- Enfant ←Mutate (Enfant, Pm) ;
- Evaluer (*Enfant*) ;
- InsertInto(POP, Enfant) ;//*Remplacement du plus mauvais individu de POP par Enfant*
- Emetteur ←Decrease (Emetteur) ; Emetteur ←Increase (Emetteur) ;
- Evaluer (Emetteur) ; InsertInto (Emetteur, POP) ;
- Récepteur ←Decrease (Récepteur) ; Récepteur ←Increase (Récepteur) ;
- Evaluer (Récepteur) ; InsertInto (Récepteur, POP) ;
Fin pour
- Return BestIndividualIn (POP);
Fin

4.4.2 Représentation d'un individu

Un chromosome représente une solution au problème de partitionnement de l'ensemble des sommets $|V|$ du graphe G. Il est constitué de $|V|$ gènes. Le i^{eme} gène contient l'identifiant du cluster (communauté) auquel appartient le sommet i du graphe G.

Le chromosome est représenté par un tableau T de type entier de taille $|V|$. Rappelons que $|V|$ représente le nombre de sommets du graphe G à partitionner. Le tableau T permet de stocker l'identifiant du cluster (IDclust) de chaque sommet i, i.e., $T(i)$ est l'identifiant du cluster du sommet i. La figure 4.4 illustre la représentation du chromosome adoptée dans notre algorithme.

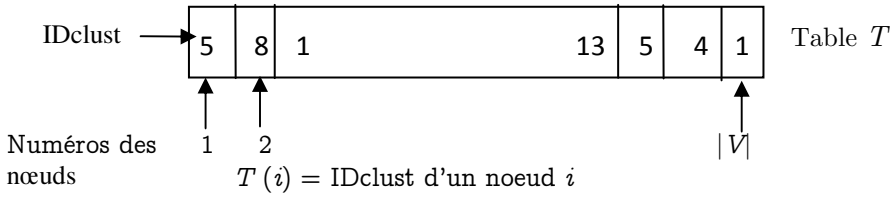


Fig 4.4. Représentation d'un chromosome

4.4.3 Initialisation d'un Individu

Lors de l'initialisation d'une solution, chaque sommet se voit attribuer aléatoirement un identifiant de cluster. Cependant nous avons besoin d'un mécanisme pour donner un sens au placement initial des sommets dans des clusters. Si deux sommets doivent être dans le même cluster, ils doivent posséder une connectivité avec les autres sommets. Dans le cas le plus simple, ils peuvent être voisins. De ce fait, après avoir attribué aléatoirement des IDclust aux sommets du graphe, nous sélectionnons certains sommets et nous attribuons respectivement leurs identifiants aux sommets qui leurs sont adjacents. Ce procédé est une étape clé dans le processus d'initialisation; il permet d'améliorer substantiellement la convergence de l'algorithme, et élimine les itérations inutiles.

4.4.4 Les opérateurs

Les opérateurs génétiques utilisés classiquement dans les AGs (Crossover, mutation) sont repris dans notre algorithme génétique. On y a intégré, notamment

une certaine connaissance du domaine de détection de communautés. On a également conçu deux nouveaux opérateurs spécialisés qui ne nécessitent pas de réglage particulier : *Increase* et *Decrease* où des heuristiques y sont incorporées pour améliorer le score de qualité du partitionnement.

Crossover

Comme la structure des clusters est une propriété relationnelle, des identifiants de clusters différents peuvent représenter le même cluster dans deux chromosomes différents. Par exemple IDclust = 1 de l'individu X et IDclust = 9 de l'individu Y ont leurs clusters respectifs qui représentent des ensembles de nœuds identiques, tandis que l'ensemble des nœuds associé au cluster d'IDclust = 1 de l'individu X n'a rien à voir avec l'ensemble des nœuds associé au cluster d'IDclust = 1 de l'individu Y. Notre algorithme ne peut donc pas utiliser le schéma traditionnel du *Crossover*. Ainsi, tout comme dans l'approche [Tas et al. 07], nous ne faisons pas un échange de gènes dans notre *Crossover*, mais plutôt un transfert des identifiants de clusters de quelques sommets d'un chromosome (Emetteur) vers les sommets correspondants d'un autre chromosome (Récepteur), comme on peut le constater sur la figure 4.5. Cette opération est donc réalisée entre deux chromosomes parents, le chromosome Emetteur et le duplicata du deuxième chromosome parent sélectionné, avec une probabilité P_c. Le duplicata sera nommé chromosome Récepteur. Nous sélectionnons d'abord un IDclust du chromosome Emetteur. Nous recherchons alors itérativement les sommets du cluster identifié par IDclust dans le chromosome Sender, ensuite nous transférons l'identifiant IDclust sélectionné précédemment aux sommets correspondants du chromosome Récepteur. Cette opération de transfert entre l'Emetteur et Récepteur est réalisée pour un nombre de clusters défini aléatoirement. Le chromosome Récepteur ayant subi des modifications génétiques remplacera le plus mauvais chromosome de la population.

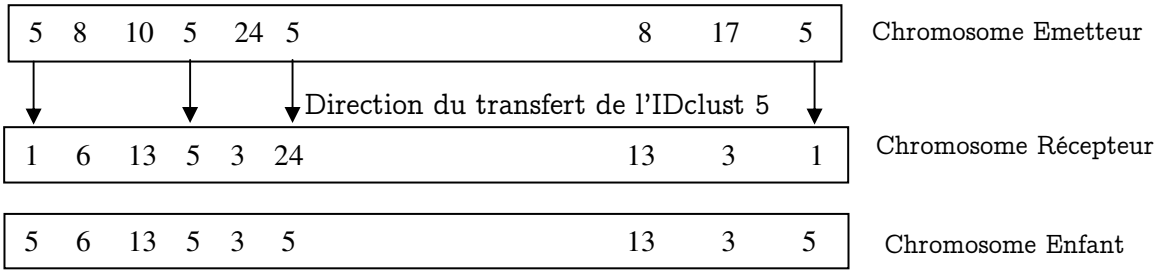


Fig 4.5. Illustration du crossover entre Emetteur et Récepteur

Mutation

Après avoir effectué le *Crossover*, nous procédons à la mutation du fils obtenu, avec une certaine probabilité P_m. La *Mutation* sur un individu fils consiste à tirer un nœud aléatoirement de l'individu, et lui affecter un nouvel IDclust pris aléatoirement parmi ceux existant dans l'individu.

Decrease

Cet opérateur vise à diminuer le nombre de clusters tout en améliorant la *fitness* d'un individu. Dans certains cas la fonction objectif d'un individu est convenable, mais la structure codée par cet individu peut comporter des clusters pour lesquels localement cette fonction n'est pas satisfaisante (négative). C'est le cas lorsque le nombre de liens internes W_{in} dans un cluster est inférieur à son nombre de liens vers l'extérieur W_{out}. Pour améliorer la fitness globale de l'individu nous devons supprimer les clusters pour lesquels W_{in}-$W_{out} < 0$. Ainsi, un cluster est supprimé en mettant chacun de ses nœuds dans le cluster de l'un de ses voisins qui améliore mieux la *fitness* globale.

Increase

Cet opérateur est utilisé pour augmenter le nombre de clusters. Il arrive d'avoir un cluster qui peut contenir des sommets qui ne sont pas connectés entre eux. Un tel cluster sera divisé en deux clusters. Ainsi, les sommets interconnectés seront laissés dans ce cluster, les autres seront transférés vers le deuxième cluster nouvellement créé.

Après le *Crossover*, on applique aux deux parents les opérateurs *Increase* et *Decrease*. Ces parents seront réévaluées et réinjectés dans la population en remplacement des parents initiaux.

4.4.5 Complexité de l'algorithme évolutionnaire

A la fin des Gen_max itérations l'algorithme se termine et donne la meilleure partition trouvée des nœuds du graphe G à travers le processus d'évolution. L'étape la plus coûteuse du processus évolutionnaire est l'évaluation des individus. Le coût de cette évaluation est proportionnelle au nombre d'arêtes $|E|$ du graphe G.

L'algorithme évolutionnaire adopté est l'algorithme génétique à l'état stationnaire (SSGA), son adaptation est présentée dans l'algorithme 4.1.
L'évaluation d'un individu (calcul de la modularité de la partition correspondante) est réalisée avec une complexité temporelle en $O(|E|)$ (voir section 2.4.1). Rappelons que $(|E|)$ est le nombre d'arêtes du graphe.
L'évaluation de la population entière est effectuée une seule fois au début de l'algorithme avec une complexité $O(S\times|E|)$, avec S qui représente la taille de la population. A chaque itération, seul trois individus sont évalués, à savoir:

- L'enfant obtenu après l'opération *Crossover*.
- Les deux parents sélectionnés après leur avoir appliqués, les opérateurs *Increase* et *Decrease*.

Par conséquent, la complexité temporelle pour les Gen_max itérations est de l'ordre de $O(3\times Gen\_max\times|E|)$.
Enfin, en vertu des évaluations antérieures concernant la complexité engendrée par les différentes opérations de l'algorithme adopté, la complexité du processus évolutionnaire est de l'ordre de $O((S+3\times Gen\_max)\times|E|)$.

Ainsi, étant donné un graphe biparti $G=(V, E)$ et son graphe-arête $G^*=(E, V^*)$. Le nombre d'arêtes $|V^*|$ est proportionnel au nombre de nœuds $|E|$ de G^*, car il dépend du degré moyen de G^*, désigné par le terme D_{Avg}. En effet, $D_{Avg}= 2\times|V^*|/|E|$, autrement dit, $|V^*| = D_{Avg}\times|E|/2$. Nous en déduisons que, $O(|V^*|) = O(|E|)$. Par conséquent, l'exécution de l'algorithme évolutionnaire sur le graphe-arête G^* requiert une complexité algorithmique de l'ordre de $O((S+3\times Gen\_max)\times|E|)$.

En somme, l'utilisation de la modularité de Newman en tant que *fitness* pour trouver un bon partitionnement des arêtes du graphe est une première étape d'optimisation. Elle permet d'obtenir une base solide sur laquelle s'appuiera la seconde phase de notre approche. Le partitionnement résultant définit alors des communautés chevauchantes de sommets. Un score de modularité chevauchante de Mancoridis (cf. sous-section 2.4.1 C) est associé à cette structure communautaire. Mais, afin d'affiner cette structure et obtenir des communautés significatives, il est nécessaire d'améliorer ce score tout en intégrant une certaine sémantique. A ce titre, nous procédons à une optimisation locale de la modularité. En fait, le but ultime de notre approche concerne l'optimisation de la modularité chevauchante de Mancoridis,

la modularité Newman ne sera pas prise en compte lors de la seconde phase de notre approche. La sous-section 4.5 suivante prend en considération cet objectif et décrit en détail la phase d'optimisation locale.

4.5 Algorithme d'optimisation locale de la modularité chevauchante

Les groupes de sommets du graphe G^* obtenus à travers le processus d'évolution définissent des groupes de sommets chevauchants du graphe G. D'après l'analyse de certaines partitions trouvées nous avons remarqué qu'il est possible d'avoir des cas particuliers, à savoir, des communautés singletons ou des communautés à deux sommets seulement, et des communautés constituées exclusivement de sommets chevauchants. L'étape suivante consiste alors à optimiser au mieux la modularité réseau. A ce propos, on doit utiliser la modularité MQ (Modularity Quality), introduite par Mancoridis [Man et al. 98] afin de mesurer la qualité d'une partition de sommets du graphe. Cette métrique repose sur la différence entre les ratios de connectivité interne et externe des communautés. Elle est définie comme suit par l'équation 4.3 :

$$MQ = \frac{1}{k}\sum_{i=1}^{k}\left[S(C_i, C_i) - \frac{1}{(k-1)}\sum_{j=1\ j\neq i}^{k} S(C_i, C_j)\right] \tag{4.3}$$

Avec $S(C_i, C_j) = \frac{E(C_i, C_j)}{|C_i| \times |C_j|}$ tel que $E(C_i, C_j)$ représente le nombre d'arêtes entre les sommets des communautés C_i et C_j.

L'équation 4.3 peut s'écrire sous la forme $MQ = MQ^+ - MQ^-$ avec -1 < MQ < 1. Les expressions $MQ+$ et MQ^- représentent respectivement la cohésion interne et la cohésion externe des communautés.

4.5.1 Mesure MQ_{Over} pour une décomposition en groupes chevauchants

La mesure de qualité MQ a été généralisée par Bourqui [Bou Aub 09] pour une décomposition de l'ensemble des sommets d'un graphe en groupes chevauchants. La généralisation de cette mesure est notée MQ_{Over}.

Considérons un graphe $G = (V, E)$ où V est l'ensemble des sommets, et E l'ensemble des arêtes. Soit alors, une décomposition $Cm = \{Cm_1, Cm_2, \dots, Cm_k\}$ de l'ensemble des sommets V, telle qu'il existe un couple (i, j) pour lequel on a $Cm_i \cap Cm_j \neq \emptyset$. Ainsi, la mesure de qualité MQ_{Over} a été définie comme la différence entre la cohésion interne et la cohésion externe des groupes Cmi comme suit :

$$MQ_{Over} = MQ^{+} - MQ^{-}_{Over} \text{ avec } MQ^{+} = \frac{1}{k}\sum_{i=1}^{k} S(Cm_i, Cm_i)$$

et $\quad MQ^{-}_{Over} = \frac{1}{k(k-1)} \sum_{i=1} \sum_{j=1, j\neq i} S_{Over}\left(Cm_i, Cm_j\right)$

tel que $\quad S_{Over}\left(Cm_i, Cm_j\right) = \frac{E\left(Cm_i, Cm_j \backslash Cm_i\right)}{|Cm_i| \times |Cm_j \backslash Cm_i|}$

Avec

- $\left|Cm_j \backslash Cm_i\right|$: le nombre de sommets du groupe Cm_j dépourvu des sommets qu'il a en commun avec Cm_i,
- $E\left(Cm_i, Cm_j \backslash Cm_i\right)$: le nombre d'arêtes entre Cm_i et $Cm_j \backslash Cm_i$. Notons que $Cm_j \backslash Cm_i$ représente le groupe Cm_j sans les sommets qu'il a en commun avec Cm_i.

La formulation de la mesure de qualité MQ_{Over} est décrite en détails par l'équation 4.4.

$$MQ_{Over} = \frac{1}{k}\sum_{i=1}^{k} \frac{E(Cm_i, Cm_i)}{|Cm_i| \times |Cm_i|} - \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j=1, j\neq i}^{k} \frac{E\left(Cm_i, Cm_j \backslash Cm_i\right)}{|Cm_i| \times |Cm_j \backslash Cm_i|} \quad (4.4)$$

4.5.2 Adaptation de la mesure MQ_{Over} à un graphe biparti

Soit $G = (V_r \cup V_b, E)$ un graphe biparti non orienté et non pondéré où V_r et V_b représentent respectivement les ensembles de sommets rouges et bleus comme défini précédemment dans la sous-section 4.2, et E représente l'ensemble des arêtes.

Soit $Cm = \{Cm_1, Cm_2, \dots, Cm_k\}$ une décomposition de l'ensemble $V_r \cup V_b$. On décompose chaque Cm_i, en deux sous-ensembles de sommets rouges et bleus, représentés respectivement par Cmr_i et Cmb_i sachant que $Cm_i = Cmr_i \cup Cmb_i$.

La mesure de qualité d'une décomposition des sommets d'un graphe biparti en groupes chevauchants [Sou et al. 13] est définie par MQB_{Over}:

$$MQB_{Over} = MQB^{+} - MQB^{-}_{Over}$$

telle que $MQB^{+} = \frac{1}{k}\sum_{i=1}^{k} S(Cm_i, Cm_i)$

et $MQB^{-}_{Over} = \frac{1}{k(k-1)}\sum_{i=1}^{k}\sum_{j=1,j\neq i}^{k} S_{Over}\left(Cm_i, Cm_j\right)$

Pour un groupe de sommets Cm_i, le nombre maximum d'arêtes qu'on peut avoir est $|Cmr_i|\times|Cmb_i|$. Ceci représente le nombre d'arêtes dans un sous-graphe biparti complet. Par conséquent, on a la cohésion interne des groupes formulée par l'équation 4.5 :

$$MQB^{+} = \frac{1}{k}\sum_{i=1}^{k} \frac{E(Cm_i, Cm_i)}{|Cmr_i|\times|Cmb_i|} \tag{4.5}$$

En ce qui concerne la cohésion externe MQB^{-}_{Over}, on doit considérer le cas du sous-graphe complet contenant tous les sommets rouges et bleus des deux groupes de sommets Cm_i et Cm_j à la fois. On doit prendre en compte le fait que nous avons des liens seulement entre les nœuds rouges et les nœuds bleus. La figure 4.6 illustre les notations utilisées dans la formule 4.6 calculant MQB^{-}_{Over}.

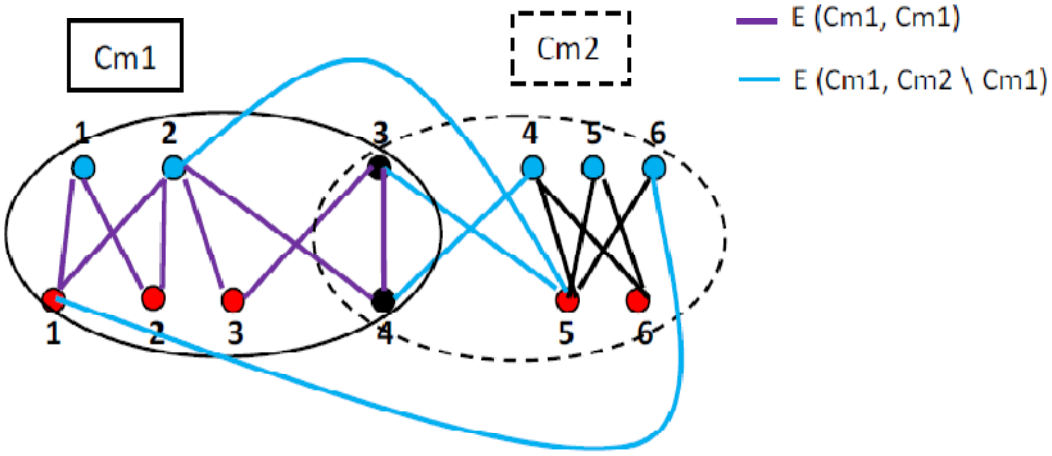


Fig 4.6. Notations pour le calcul de la modularité de Mancoridis

Ainsi, on a :

$$MQB^{-}_{Over} = \frac{1}{k(k-1)}\sum_{i=1}^{k}\sum_{j=1,j\neq i}^{k} S_{Over}\left(Cm_i, Cm_j\right) = \frac{1}{k(k-1)}\sum_{i=1}^{k}\sum_{j=1,j\neq i}^{k} \frac{E(Cmi,\ Cmj\backslash Cmi)}{|Cm_i|\times|Cm_j\backslash Cm_i|}$$

Qui donne finalement l'expression de l'équation 4.6:

$$MQB^{-}_{Over} = \frac{1}{k(k-1)}\sum_{i=1}^{k}\sum_{j=1,j\neq i}^{k} \frac{E(Cmi,\ \ Cmj\backslash Cmi)}{|Cmr_i|\times|Cmb_j\backslash Cmb_i| + |Cmb_i|\times|Cmr_j\backslash Cmr_i|} \tag{4.6}$$

4.5.3 Optimisation locale de la modularité

Avant d'aborder la façon d'optimiser localement la modularité de Mancoridis, il convient d'introduire quelques notations utiles. Soit alors $G = (V, E)$ un graphe biparti et $Cm = \{Cm_1, ..., Cm_k\}$ une décomposition de V en groupes de sommets chevauchants.

- $E(i, Cm_j)$ représente le nombre de liens du sommet i avec le groupe Cm_j ;
- $Ov_i = \{Cm_j \mid i \in Cm_j\}$ définit l'ensemble des groupes $Cm_j \in Cm$ auxquels le sommet i appartient;
- $Ov^i_j = \{Cm_k \in Ov_j \mid j \in N(i)\}$ est l'ensemble de groupes auxquels le voisin j de i appartient. Rappelons que $N(i)$ représente l'ensemble des voisins de i;
- $O_{Over} = \{Cm_i \in Cm$ *et* $\forall$ le sommet $n \in Cm_i, \exists\, j \neq i \mid n \in Cm_j\}$, il définit l'ensemble des groupes Cm, composés exclusivement de sommets chevauchants.

Afin d'optimiser localement la modularité $MQB_{Over,}$ nous utilisons le fait que cette modularité est additive [Sou et al. 13]. Cette optimisation est réalisée à trois niveaux comme suit:

1. Nous optimiserons localement la modularité MQB_{Over} en supprimant les groupes Cmi de l'ensemble O_{Over}, qui diminuent la valeur de la modularité. En effet les groupes pour lesquels nous avons $S(Cm_i, Cm_i) < \sum_{j \neq i} S_{Over}(Cm_i, Cm_j)$ diminuent la modularité. Après la suppression d'un groupe, certains sommets peuvent changer de statut (un sommet chevauchant devient un sommet non chevauchant). On met alors à jour les sommets des groupes qui sont en relation avec le groupe supprimé.
2. Les sommets appartenant aux autres groupes vont être filtrés pour décider s'ils peuvent être considérés comme des sommets chevauchants. Une certaine sémantique doit être incorporée dans l'algorithme pour décider de cette appartenance. Soit i un sommet appartenant à plusieurs groupes et soit $E(i, Cm_j)$ le nombre de liens du sommet i avec le groupe Cm_j. Si un sommet chevauchant appartient à un certain groupe, et s'il ne participe pas activement à la cohésion interne mais contribue beaucoup à la cohésion externe de ce groupe, il sera supprimé du groupe. Ceci a pour effet d'augmenter la modularité de la structure communautaire du réseau. Nous proposons en conséquence une équation que doivent vérifier les sommets chevauchants pour appartenir à une certaine communauté. Pour qu'un sommet i reste membre de

la communauté Cm_j, la proportion de ses liens avec cette communauté doit être supérieure ou égale à la moyenne des proportions des liens de tous les groupes partageant ce nœud.

Toutefois si le sommet i réalise un score d'au moins 0.5, il est considéré comme membre du groupe Cm_j (cf. exemple de la figure 4.7). Le sommet i doit donc vérifier l'une des conditions suivantes :

$$\text{Si } i \in Cmr_j \text{ alors } \frac{E(i,\ Cm_j)}{|Cmb_j|} \geq Min\left(\frac{(\sum_j \frac{E(i,\ Cm_j)}{|Cmb_j|})}{|Ov_i|},\ o.5\right)$$

$$\text{Si } i \in Cmb_j \text{ alors } \frac{E(i,\ Cm_j)}{|Cmr_j|} \geq Min\left(\frac{(\sum_j \frac{E(i,\ Cm_j)}{|Cmr_j|})}{|Ov_i|},\ o.5\right)$$

3. A la fin de cette étape, nous pouvons obtenir des communautés ayant seulement un ou deux sommets. De telles communautés ne sont pas significatives, elles seront alors supprimées en mettant chacun de ses sommets dans la communauté d'un de ses voisins qui augmenterait la modularité.

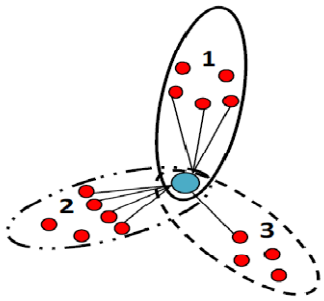


Fig 4.7. Un nœud appartenant à 3 groupes mais n'appartient qu'à 2 communautés chevauchantes

Dans le schéma de la figure 4.7, on vérifie les conditions suivantes:

- pour la communauté1, nous avons:$\frac{3}{5} = 0.6 \geq \frac{\frac{3}{5}+\frac{4}{6}+\frac{1}{4}}{3} = 0.505$, alors le sommet appartient à la communauté 1 ;
- pour la communauté 2, nous avons aussi $\frac{4}{6} = 0.67 \geq \frac{\frac{3}{5}+\frac{4}{6}+\frac{1}{4}}{3} = 0.505$, alors le sommet appartient à la communauté 2 ;
- Mais, pour la communauté 3, la condition n'est pas vérifiée. $\frac{1}{4} = 0.25 < \frac{\frac{3}{5}+\frac{4}{6}+\frac{1}{4}}{3} = 0.505$, et $\frac{1}{4} < 0.5$, par conséquent le sommet n'appartient pas à la communauté 3.

Les trois étapes précédentes de l'optimisation locale de la modularité ont été récapitulées dans l'algorithme 4.2.

Algorithme 4.2-Algorithme d'optimisation locale de la modularité

Début

Pour tout $Cm_i \in O_{Over}$ **faire**

Calculer $mq_i = S(Cm_i, Cm_i) - \sum_{j=1, j\neq i}^{k} S_{Over}(Cm_i, Cm_j)$;

Fin pour ;

Tant que $\text{argmin}_{\{\forall Cmi \in Oover\}} (\{mq_i < 0\}) \neq \phi$ **faire**

$O_{Over} = O_{Over} - \{Cm_i\}$; *//Retirer le groupe Cm_i donné par argmin*

StatusUpdate (Cm); *//Mise à jour de O_{over}et des autres groupes*

Pour tout $Cm_i \in O_{Over}$ **faire**

Calculer mq_i ;

Fin pour ;

Fin Tantque ;

Pour tout i chevauchant **faire** *//Filtrage des nœuds chevauchants*

Pour tout $Cm_j \in Ov_i$ **faire**

Si $[i \in Cmr_j$ et $\frac{E(i,\ Cm_j)}{|Cmb_j|} < Min\left(\frac{(\sum_j \frac{E(i,\ Cm_j)}{|Cmb_j|})}{|Ov_i|},\ o.5\right)]$ ou

$[i \in Cmb_j$ et $\frac{E(i,\ Cm_j)}{|Cmr_j|} < Min\left(\frac{(\sum_j \frac{E(i,\ Cm_j)}{|Cmr_j|})}{|Ov_i|},\ o.5\right)]$

alors $Ov_i = Ov_i - Cm_i$ *// i est supprimé de la communauté Cm_j*

Fin pour ;

Fin pour ;

Pour tout $Cm_i \in Cm$ / $|Cm_i| = 1$ ou 2 **faire**

//Suppression des communautés à 1ou à 2 nœuds

Pour tout n non chevauchant $\in Cm_i$ **faire**

$k = \text{argmax}_{\{\forall j \in N(n)\}}(\{MQB_{Over} | Ov_n = Ov_j{}^n\})$;

//Mettre n dans la communauté d'un voisin pour lequel on a une meilleure

//modularité, rappelons que N(n) représente l'ensemble des voisins de n

$Ov_n = Ov_k{}^n$;

Fin pour ;

Pour tout n chevauchant $\in Cm_{i,}$ **faire**

$Ov_n = Ov_n - \{Cm_i\}$; *//Suppression de n de la communauté Cm_i*

Fin pour ;

$Cm = Cm - \{Cm_i\}$; *// Suppression de communauté Cm_i de la décomposition Cm*

Fin pour ;

Fin

Une fois les sommets chevauchants filtrés, on obtient des communautés chevauchantes de sommets du graphe biparti initial. La complexité de cet algorithme est au pire des cas en $O(|V|)$. Notre approche constituée de deux phases principales, possède au final une complexité temporelle en $O((S+3\times Gen\_max) \times |E|+|V|)$.

4.6 Conclusion

Dans ce chapitre nous avons présenté une approche de détection de communautés chevauchantes mixtes significatives dans des réseaux bipartis. L'approche est basée sur une double optimisation de la modularité réseau. La première étant une optimisation globale de la modularité réseau, réalisée par un algorithme génétique doté d'opérateurs classiques adaptés au domaine, et de deux nouveaux opérateurs spécialisés. La deuxième est une optimisation locale visant à améliorer encore le score de la modularité réseau. Par ailleurs, nous y avons intégré une certaine sémantique en vue d'obtenir des communautés chevauchantes significatives. Cette sémantique est introduite sous forme d'une équation qui exprime l'appartenance des nœuds chevauchants à une certaine communauté. Nous avons également adapté une mesure de qualité pour quantifier la structure des communautés chevauchantes dans un réseau biparti. Cette mesure est à l'origine destinée à l'évaluation d'un découpage des sommets d'un réseau monoparti en communautés chevauchantes. Le chapitre suivant sera consacré totalement à la mise en œuvre de cette approche afin de mettre en évidence sa pertinence et sa faisabilité.

5

Expérimentation

5.1 Cadre expérimental

Il convient à présent d'effectuer des tests afin d'évaluer l'efficacité et l'efficience de notre approche. Pour cela, des expérimentations ont été menées sur deux fronts. Dans le premier, les tests consistent à montrer la performance de l'algorithme génétique à trouver des communautés disjointes sur deux ensembles de données, à savoir, plusieurs réseaux réels et un groupe de réseaux synthétiques. Dans le second, les tests visent à montrer l'efficacité de l'approche proposée à découvrir des communautés chevauchantes mixtes dans des réseaux bipartis. À cet effet, un groupe de réseaux synthétiques ainsi qu'un réseau réel dont la structure est connue pour être difficile à comprendre ont été également utilisés.

Notre logiciel de détection de communautés chevauchantes dans des réseaux bipartis est composé de 4 modules (ils correspondent aux éléments du schéma de la figure 4.2), il a été implémenté sous l'environnement Eclipse Java, sur un Compaq avec Intel Core 2 Duo CPU 2GHz et 2 Go de RAM. Les différents diagrammes présentés dans ce rapport sont réalisés avec le logiciel de visualisation Gephi 0.8 [15].

[15]Logiciel en open source, téléchargeable à l'adresse : https://gephi.org/users/download/

5.2 Ajustement des paramètres de l'algorithme génétique

Les algorithmes génétiques ont prouvé leur efficacité en optimisation dans différents domaines. Cependant, cette efficacité est fortement liée à un choix judicieux des paramètres de l'algorithme. Afin de sélectionner la meilleure combinaison de ses paramètres à considérer dans la suite des expérimentations, plusieurs tests d'évaluation de ses paramètres ont été réalisés. Ces tests sont effectués sur le réseau Zackary club.

Réseau Zachary club [Zackary 77] correspond à un réseau social des liens d'amitié entre 34 membres d'un club de karaté, dans une université aux États-Unis, au courant des années 1970.

L'expérimentation menée est basée sur trois séries de tests qui permettent la variation de la modularité en fonction de la taille de la population, du nombre de générations, du taux de croisement et, enfin, du taux de mutation. Pour avoir une idée plus réaliste, l'algorithme génétique a été exécuté une dizaine de fois pour chaque série de tests. La moyenne des modularités obtenues est utilisée par la suite, comme résultat du test. Les résultats de ces tests sont exposés dans les trois sections suivantes.

5.2.1 Variation de la taille de la population et du nombre de génération

Dans ce test, le taux de croisement ainsi que le taux de mutation ont été fixés respectivement à 0,8 et 0,2. Ensuite, en fixant le nombre de générations à 100, 300 et 500, on affecte dans chaque cas, les valeurs 10, 20, 30, 70 et 100 à la taille de la population. Comme annoncé ci-dessus, nous considérons, dans chaque cas, la moyenne de la modularité après dix exécutions de l'algorithme. Les résultats de cette expérimentation concernant la modularité Q du réseau Zackary club sont résumés dans le tableau 5.1.

On constate que dans le cas où la taille de la population est supérieure ou égale à 30 ($\geqslant$30), et pour un nombre de générations supérieur à 300, la modularité atteint ses valeurs optimales (entre 0.382 et 0.404). Avec cette tendance, l'algorithme semble

converger vers une solution raisonnable à partir de 300 générations et une population de 30 individus.

Taille de la population	**Nombre de générations**		
	100	**300**	**500**
10	0.143	0.264	0.287
20	0.205	0.260	0.273
30	0.271	0.401	0.401
70	0.230	0.382	0.400
100	0.295	0.403	0.404

Tableau 5.1. Modularité en fonction de la taille de la population et du nombre de générations

5.2.2 Variation du taux de crossover et de la taille de la population

Partant du même principe que le test précédent, mais en fixant le nombre de générations à 300 et le taux de mutation à 0.2, on fait varier la taille de la population et le taux du crossover. A la fin du test les valeurs moyennes de la modularité sont reportées dans le tableau 5.2.

On constate que les résultats pour le crossover égal à 0.8 et à 0.9 sont assez similaires et sont meilleurs, comparés aux résultats avec les autres valeurs du crossover. Les meilleures valeurs de la modularité sont celles obtenues pour des populations de tailles 30, 70 et 100. Mais l'amélioration de la modularité pour une population supérieure à 30 n'est pas toujours obtenue. Par ailleurs, pour les cas de croisement égal à 0.5, 0.6 et 1, les résultats sont détériorés.

Taille de la population	**Cros 0.5**	**Cros 0.6**	**Cros 0.7**	**Cros 0.8**	**Cros 0.9**	**Cros 1**
10	0.119	0.187	0.203	0.13	0.172	0.122
20	0.143	0.169	0.316	0.269	0.287	0.317
30	0.233	0.28	0.321	0.398	0.387	0.301
70	0.181	0.312	0.319	0.382	0.378	0.31
100	0.208	0.347	0.337	0.4	0.391	0.352

Tableau 5.2. Modularité en fonction de la taille de la population et du taux de crossover

5.2.3 Variation du taux de mutation et de la taille de la population

Nous avons également adopté le même principe que dans les deux précédents tests (cf. sous-sections 5.2.1 et 5.2.2). Cette fois-ci, le taux de croisement est fixé à 0.8 et le nombre de générations à 300. Ainsi, à la fin de l'expérimentation, les différentes valeurs de la modularité sont recueillies dans le tableau 5.3.

On peut remarquer que les meilleurs résultats sont obtenus pour un taux de mutation de 0.1 et 0.2, et cela pour pratiquement toutes les tailles de population considérées. Cette situation reflète de manière réaliste le fonctionnement de la nature, où les mutations sont des évènements très rares donc avec un taux de mutation faible.

Taille de la Population	**Mut 0**	**Mut 0.1**	**Mut 0.2**	**Mut 0.5**	**Mut 1**
10	0.104	0.137	0.141	0.107	0.092
20	0.124	0.21	0.236	0.141	0.183
30	0.212	0.378	0.401	0.308	0.297
70	0.237	0.381	0.389	0.271	0.282
100	0.326	0.394	0.402	0.212	0.209

Tableau 5.3. Modularité en fonction de la taille de la population et du taux de mutation (Mut)

Au vu des tests précédents, nous avons choisi d'expérimenter notre approche de détection de communautés avec l'algorithme génétique pour la combinaison des paramètres qui a donné les meilleurs résultats, à savoir, taille de la population = 30, nombre de génération = 300, taux de crossover = 0.8 et taux de mutation = 0.2.

5.3 Métriques d'évaluation

L'évaluation dans le contexte de la détection de communautés consiste à vérifier dans quelle mesure la méthode de détection de communautés est susceptible de retrouver les communautés cibles. Ainsi, pour valider et situer notre approche par rapport à certains travaux existants, nous nous sommes particulièrement intéressés à deux types de métriques d'évaluation très largement utilisées dans les travaux sur la détection de communautés dans les réseaux :

- La première, dénommée modularité de Newman, nous l'avons utilisée dans le cas de notre approche afin de quantifier la partition des arêtes du graphe biparti.

- La deuxième métrique, appelée modularité de Mancoridis, a été adaptée dans [Bou Aub 09] aux décompositions chevauchantes. Nous l'avons exprimée sous une nouvelle forme adaptée aux graphes bipartis. Nous l'avons ensuite utilisée pour mesurer la modularité de la décomposition de nœuds, résultant du partitionnement des arêtes.

- Par ailleurs, nous avons utilisé une troisième métrique nommée "*information mutuelle normalisée*" pour mesurer la similarité entre la décomposition de nœuds obtenue à l'aide de notre méthode et une décomposition cible.

5.4 Tests de fiabilité de l'algorithme génétique

Le but des tests suivants est de mettre en évidence la capacité de notre algorithme à détecter efficacement les communautés disjointes, d'une part et, d'autre part, pour évaluer sa performance en fonction de son temps d'exécution. Le test de la fiabilité est réalisé sur des ensembles de données synthétiques et réelles. La performance, à savoir, le temps d'exécution est testé sur un autre jeu de données réelles. Les détails sur l'ensemble de ces tests et les résultats obtenus sont décrits dans les paragraphes suivants.

5.4.1 Ensemble de données synthétiques

Afin de vérifier la capacité de notre algorithme génétique à détecter la structure communautaire d'un réseau, nous utilisons le benchmark de référence proposé par Girvan et Newman dans [Gir New 02]. Le benchmark est généré par un logiciel mis en ligne par Fortunato [Fortuna 08]. Le réseau est constitué de 128 nœuds répartis en 4 communautés de 32 nœuds chacune. Les arêtes sont placées au hasard entre les paires de nœuds mais de telle sorte que $Zout + Zin = 16$, où Zin et $Zout$ représentent respectivement le degré interne et externe d'un nœud par rapport à sa communauté. Si $Zin > Zout$, les voisins d'un nœud à l'intérieur de sa communauté, sont plus nombreux que les voisins appartenant aux trois autres communautés.

Nous avons généré 30 réseaux pour chacun des cas de *Zout* allant de 1 à 8. Avec chacun de ces réseaux, nous avons exécuté notre algorithme génétique. Les résultats obtenus sont utilisés pour calculer les moyennes des fractions de sommets bien classés obtenues. Nous avons reportés ces moyennes sur la figure 5.2 ainsi que celles obtenues avec l'algorithme de Girvan et Newman [Gir New 02].

Nous avons remarqué que la difficulté à trouver une bonne solution de partitionnement augmente avec la valeur de *Zout* (cf. figure 5.1). Ainsi, jusqu'à $Zout = 6$, notre algorithme détecte les communautés à 85%. Au-delà de *Zout*=6, notre algorithme détecte en moyenne les communautés à 55%. Comparativement à l'algorithme de Girvan et Newman, nous avons obtenu un gain de 7% en termes de précision.

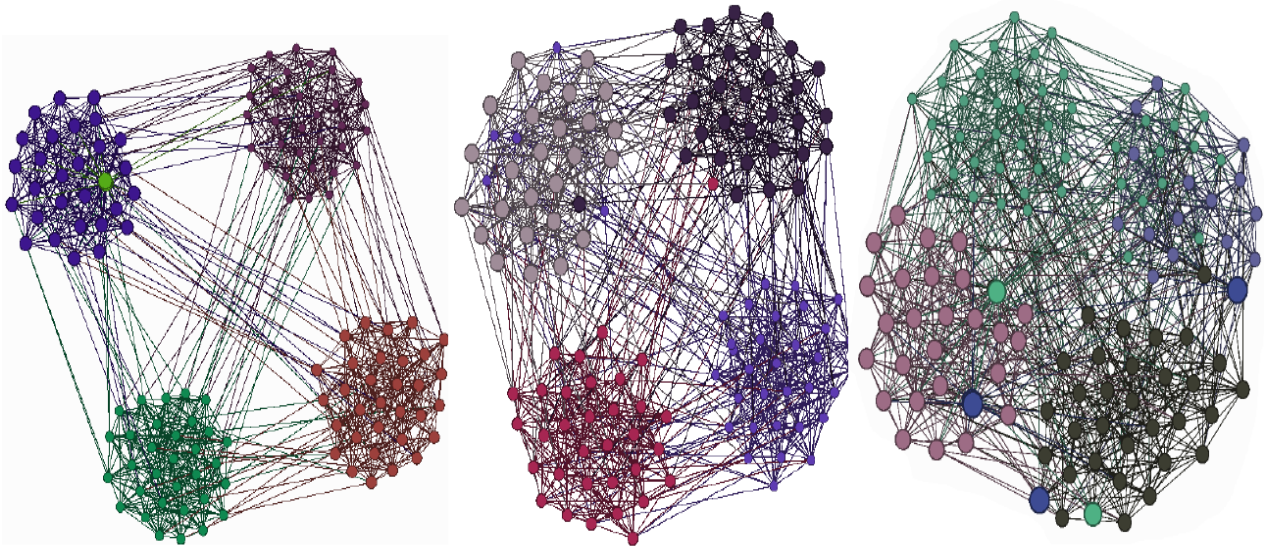


Fig 5.1. Captures d'écran des résultats obtenus par notre algorithme pour le benchmark de Newman et Girvan pour les cas : $Z_{out} = 2$, 4 et 6

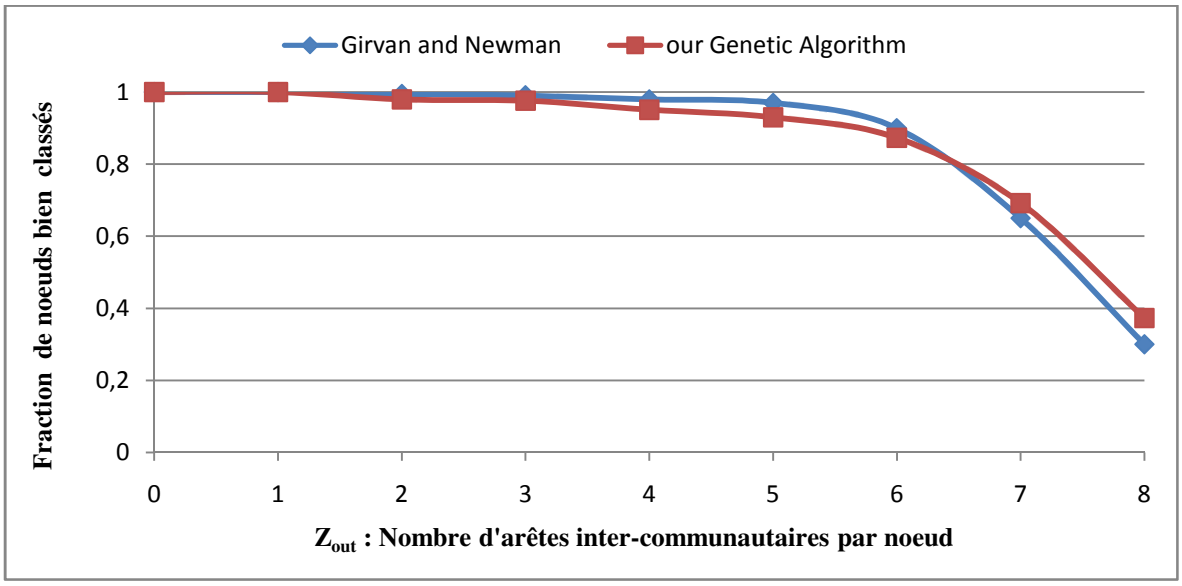


Fig 5.2. Moyennes des fractions de nœuds bien classés pour une trentaine d'exécution

5.4.2 Ensemble de données réelles

Pour mettre en évidence la fiabilité de notre algorithme d'optimisation de la modularité de Newman, nous avons effectué plusieurs tests afin de comparer notre algorithme génétique avec celui de l'approche Girvan et Newman décrit dans [Gir New 02]. Ces tests ont été effectués sur plusieurs réseaux réels connus : le réseau "Zackary Club", le réseau "American College Football", le réseau "Dolphins" et le réseau "Books about US Politics".

- **Le réseau Zackary Club** a été déjà présenté à la section 5.2

- **Le réseau American College Football** [Gir New 02] provient du collège de football des États-Unis. Le réseau représente le calendrier des matchs de la Division I lors de la saison 2000. Les nœuds du réseau représentent les équipes et les arêtes représentent les matchs de saison régulière entre les deux équipes qui ont joué ensemble. Les équipes sont réparties dans des conférences. En moyenne, les équipes ont joué 4 matchs inter-conférence et 7 matchs intra-conférence ; donc les équipes ont tendance à jouer entre les membres d'une même conférence. Le réseau est composé de 115 nœuds et de 616 arêtes regroupés en 12 équipes.

- **Le réseau Dolphins** [Lus et al. 03] est un réseau social non orienté qui représente les associations fréquentes entre 62 dauphins dans une collectivité vivant à Doubtful Sound, Nouvelle-Zélande.

- **Le réseau Books about US Politics** est un réseau de livres, sur la politique américaine, publiés au moment de l'élection présidentielle de 2004 et vendus par la librairie en ligne Amazon.com. Les nœuds du réseau sont les noms des livres et les arêtes entre les livres représentent les fréquents co-achats de livres parles mêmes acheteurs. Ce réseau est compilé par Valdis Krebs. Les sommets ont reçu des valeurs "l", "n" ou "c" pour indiquer s'ils sont «libéral», «neutre», ou «conservateurs». Ces classements ont été attribués séparément par Mark Newman sur la base de la lecture des descriptions et des critiques de livres publiés sur Amazon.

Ainsi, nous avons exécuté les deux algorithmes, une vingtaine de fois, sur chacun de ces réseaux et nous avons reporté les meilleurs résultats sur le tableau 5.4.

Réseau	**Modularité de Newman** Girvan-Newman Notre Algorithm		**Nombre de Communautés** Girvan Newman Notre Algorithm	
Zackary Club	0.409	0.417	4	4
American college Football	0.592	0.603	12	11
Dolphins	0.519	0.519	5	5
Books on Us Politics	0.517	0.526	5	5

Tableau 5.4. Les résultats concernant la modularité et le nombre de communautés obtenus avec notre algorithme et l'algorithme de Girvan et Newman sur les quatre réseaux réels considérés

- En ce qui concerne le réseau *Zackary Club*, la modularité de la partition obtenue à chaque exécution est supérieure à 0.378. La meilleure décomposition obtenue a une modularité de 0.417 avec 4 communautés. L'algorithme de Girvan et Newman trouve une modularité de 0.409, le nœud 9 y est placé différemment (cf. figure 5.3).

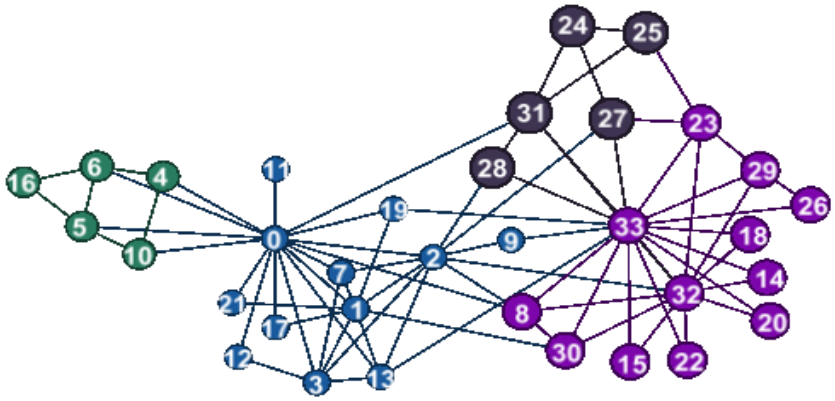


Fig 5.3. Capture d'écran de l'exécution de l'algorithme génétique sur le réseau *Zackary club*

- Au sujet du réseau *Dolphins*, notre algorithme identifie cinq communautés avec une modularité de 0.519. Les résultats trouvés sont identiques à ceux trouvés par l'algorithme de Girvan et Newman (cf. tableau 5.4).

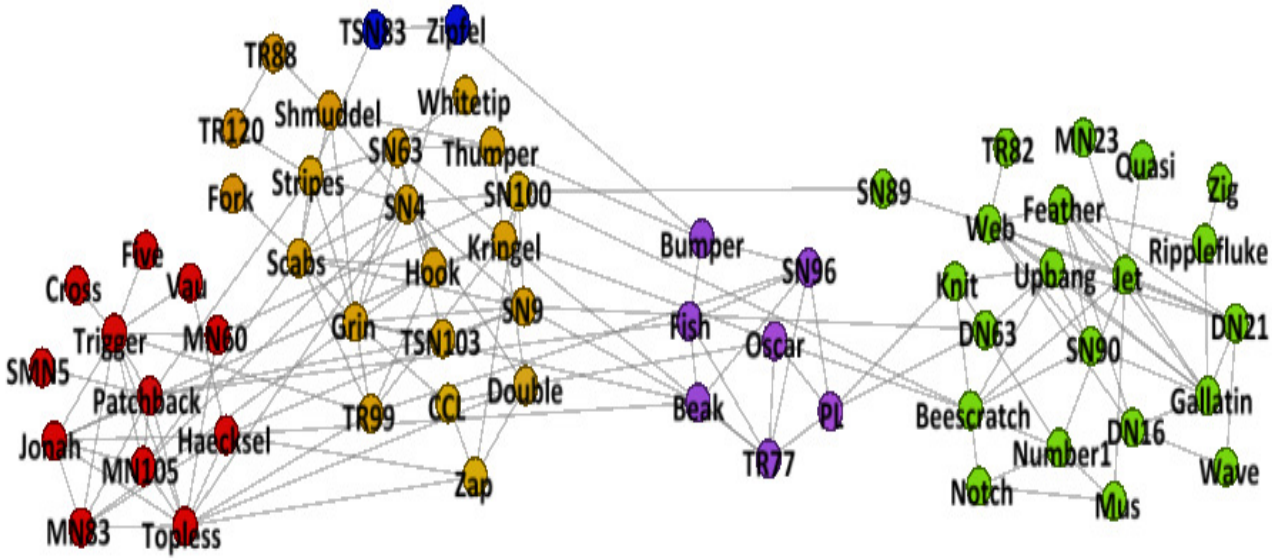


Fig 5.4. Capture d'écran de l'exécution de l'algorithme génétique sur le réseau *Dolphins*

- Pour le réseau *American College Football*, notre algorithme identifie pratiquement tous les groupes, il trouve onze groupes au lieu de douze avec une modularité de 0.603 (cf. figure 5.5). Le groupe "Independents" n'a pas été détecté. En effet, les nœuds de ce groupe qui sont au nombre de 5, se retrouvent dans les groupes " Mid-American, Big East et Sun Belt ". Ceci s'explique par le fait que ce groupe ne possède pratiquement pas de liens internes. Chacun de ses nœuds a été affecté au groupe avec lequel il est étroitement lié. Deux autres groupes n'ont pas été identifiés correctement. Il s'agit des groupes "Western Atletic" et "Sun Belt" ; ils ont formé dans la structure trouvée deux groupes, dans chacun de ces groupes on retrouve une partie de "Western Atletic" et de "Sun Belt".

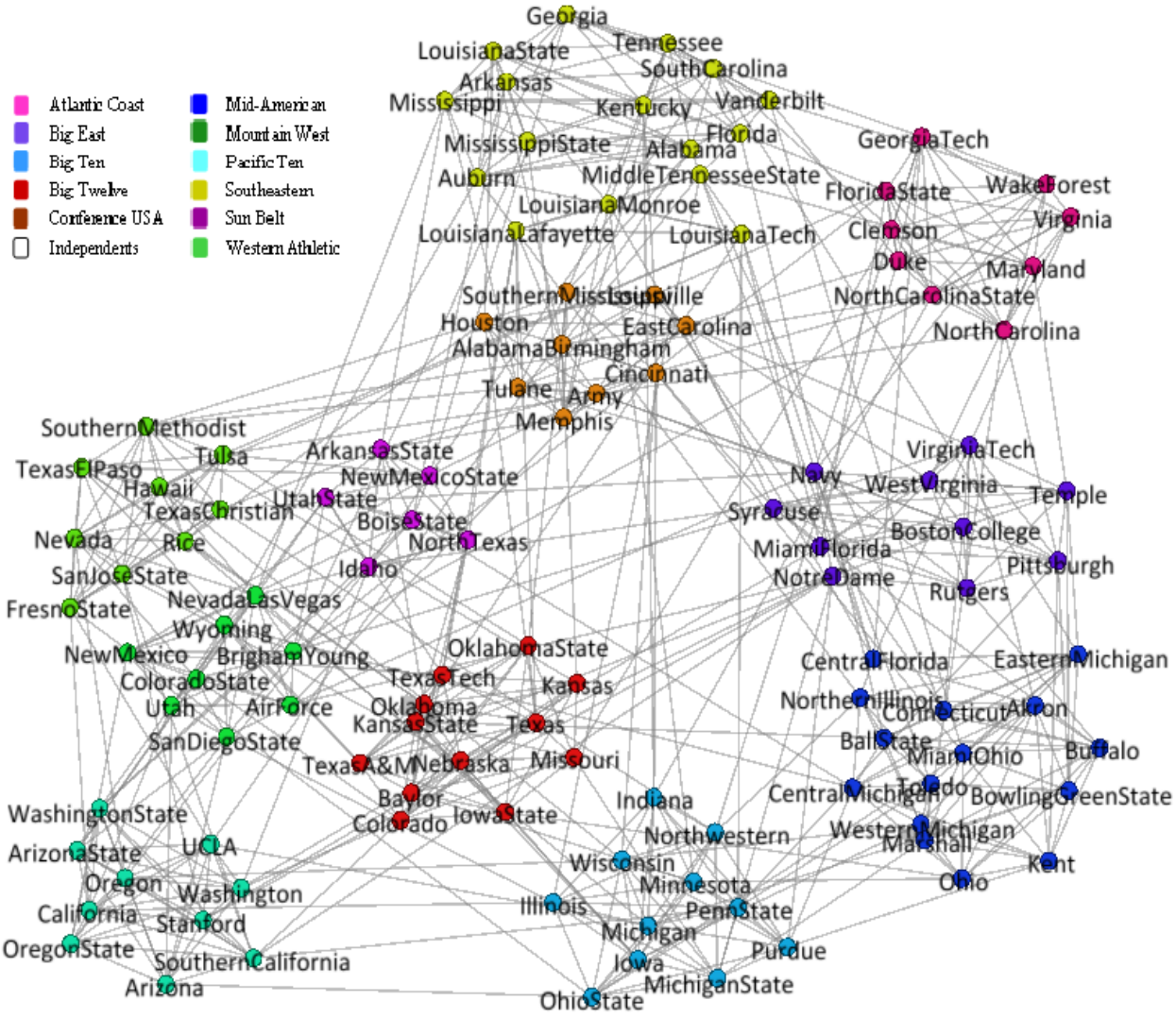


Fig 5.5. Capture d'écran de l'exécution de l'algorithme génétique sur le réseau *American College Football*

- Quant au réseau *Books on Us Politics*, notre algorithme trouve cinq communautés avec une modularité de 0.526 (cf. figure 5.6). Dans la structure détectée, les nœuds "l" sont répartis entre les communautés de couleurs *verte* et *violet*, les nœuds "c" sont dans la communauté de couleur *bleu*, les nœuds "n" sont répartis entre les communautés de couleur *rouge* et *jaune*. Nous remarquons une subdivision des groupes "l" et "n". Par conséquent, nous concluons que notre algorithme est capable de détecter de petites communautés qui peuvent être incluses dans une communauté plus large.

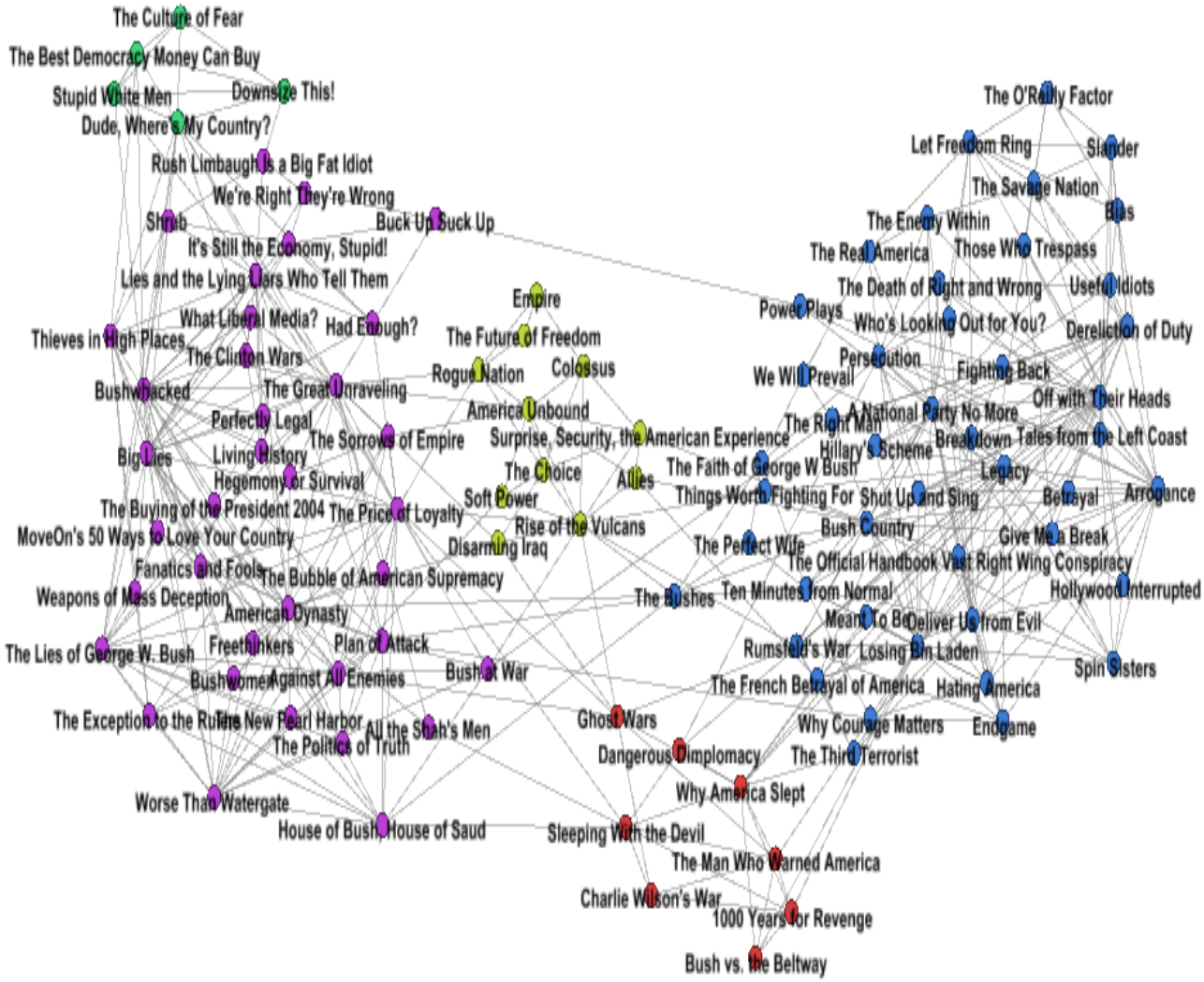


Fig 5.6. Capture d'écran de l'exécution de l'algorithme génétique sur le réseau *Books on Us Politics*

En vertu de cette expérimentation menée sur quatre réseaux réels décrits ci-dessus, nous pouvons dire que notre algorithme peut donner une bonne estimation de la modularité maximale.

5.4.3 Evaluation du temps d'exécution avec un ensemble de données réelles

Pour estimer le temps d'exécution de notre algorithme et le situer par rapport à celui des algorithmes de certains travaux existants, nous avons repris les réseaux vus précédemment et deux autres réseaux plus volumineux, à savoir le réseau Power Grid avec 4941 sommets et le réseau Internet avec 22962 sommets.

- **Le réseau Power Gird** [Wat Str 98], est un réseau non-orienté, non pondéré représentant la topologie de "Western Power" des États-Unis.

- **Le réseau Internet**, ce réseau est une capture instantanée de la structure d'Internet au niveau des systèmes autonomes[16], reconstruits à partir des tables BGP (protocole de routage externe: Border Gateway Protocol), mis en ligne par "University of Oregon Route Views Project".

Nous avons comparé le temps d'exécution de notre algorithme à ceux de trois autres algorithmes de détection de communautés disjointes, relatés dans l'état de l'art, énumérés comme suit :

1. G-N : L'algorithme de Girvan et Newman [Gir New 02] qui utilise la centralité d'intermédiarité ;
2. Fast : L'algorithme rapide proposé par Newman dans [Newman 04] qui optimise la modularité ;
3. D-M : L'algorithme de Donetti et Muñoz [Don Mun 05] utilisant la matrice Laplacienne ;
4. N-A : Notre algorithme génétique.

Les résultats concernant le temps d'exécution (en seconde) et la modularité obtenus avec ces tests sont rassemblés dans le tableau 5.5.

[16] *Système Autonome*, est un ensemble de réseaux informatiques intégrés à Internet et dont la politique de routage interne (routes à choisir en priorité, filtrage des annonces) est cohérente.

Réseau	**Nombre denœuds**	**Temps d'exécution / Modularité**			
		G -N	**D-M**	**Fast**	**N-A**
Zackary Club	33	0.01 / 0.409	0 / 0.41	0 / 0.39	0.09 / 0.417
Politics Books	62	0.17 / 0.517	0 / 0.451	0 / 0.51	0.16 / 0.526
Dolphins	105	0.32 / 0.519	0 / 0.483	0 / 0.5	0.192 / 0.519
Football	115	0.39 / 0.592	0 / 0.6	0 / 0.57	0.229 / 0.603
Power Gird	4941	> 24000	116 / 0.57	1.08 / 0.614	134 / 0.682
Internet	22962	----------	5100 / 0.692	156 / 0.632	1017 / 0.716

Tableau 5.5. Les résultats concernant le temps d'exécution et la modularité obtenus par notre algorithme et trois autres algorithmes (G-N, D-M, Fast et N-A) avec les six réseaux réels considérés. Les termes de la forme X/Y dans les colonnes G-N, D-M, Fast et N-A du tableau représentent respectivement le temps d'exécution en seconde et la modularité.

Durant cette expérimentation, nous n'avons pas pu exécuter l'algorithme G-N pour le réseau "Internet", le temps d'exécution s'est avéré très élevé. Du point de vue temps de calcul, notre algorithme est plus rapide que les algorithmes G-N et D-M et produit des structures communautaires plus satisfaisantes en terme de modularité. En revanche, notre algorithme est plus lent que Fast, mais détecte des structures communautaires présentant des scores de modularité sensiblement plus élevé. Cette comparaison directe montre que notre approche a un avantage évident en termes de qualité de la partition et présente un meilleur compromis entre la qualité et le temps d'exécution pour de grands réseaux.

5.4.4 Notre algorithme génétique versus certains algorithmes génétiques

Pour compléter l'évaluation de la performance et de la fiabilité de notre algorithme évolutionnaire, nous avons ajouté un test de comparaison de l'algorithme génétique que nous avons proposé, avec quelques algorithmes génétiques, dédiés à la détection de communautés disjointes, mentionnées dans l'état de l'art. Nous mettons l'accent, dans le tableau 5.6 sur les caractéristiques de notre algorithme avec ceux de trois

autres algorithmes génétiques, à savoir les algorithmes de [Tas Bin 06, Pizzuti 08, Lip Mil 09] :

- L'algorithme de [Tas Bin 06] est un algorithme génétique fondé sur la maximisation de la modularité réseau. Cet algorithme est évolutif pour de très grands réseaux.
- L'algorithme génétique de [Pizzuti 08] introduit la notion de *score de communauté* qu'il utilise comme fonction de qualité, et recherche un partitionnement optimal du réseau en maximisant la valeur du score de communauté.
- L'algorithme de [Lip Mil 09], est un algorithme de classification ascendante basé sur un algorithme génétique pour détecter des communautés dans des réseaux sociaux à grains fins (nombre de communautés très élevé).

Algorithme	Fitness	Type de réseaux traités	Détection automatique du nombre de communautés	Taille de la Population	Nombre de Génération	Taille des réseaux traités
Algorithme de Tasgin	Modularité de Newman	Réseaux Complexes	oui	250	500	<93 000
Algorithme de Pizzuti	Community Score	Réseaux Sociaux	oui	300	30	< 120
Algorithme de Lipczak	Modularité de Newman	Réseaux sociaux à grains fins	non	1	500	< 2500
Notre Algorithme	Modularité de Newman	Réseaux Complexes	oui	30	300	<23 000

Tableau 5.6. Notre algorithme génétique versus certains autres algorithmes génétiques cités dans l'état de l'art

Notre algorithme génétique et celui de Tasgin sont tous les deux conçus pour des réseaux complexes et optimisent la valeur de la modularité de Newman. Comparativement à celui de Tasgin où la valeur de la modularité de la partition trouvée n'a pas été rapportée dans l'étude, nous avons montré que notre algorithme donne des partitions avec des scores de modularités assez élevés (cf. section 5.4.2). Par ailleurs, notre algorithme est un algorithme génétique à état stationnaire (SSGA), la taille de sa population est réduite par rapport à celui de Tasgin et de Pizzuti (cf. tableau 5.6). L'algorithme de Pizzuti quant à lui est destiné pour des réseaux sociaux, il a été testé seulement pour des réseaux de petites tailles. D'un

autre point de vue, il utilise une taille de population de 300 individus, ce qui exige une grande mémoire. Cette contrainte a sûrement été la cause s'il n'a pas été appliqué sur des réseaux de tailles importantes. Au sujet de l'algorithme de Lipczak, celui-ci est conçu spécialement pour des réseaux sociaux à grains fins, et exige la connaissance préalable du nombre de communautés recherchées. Cet algorithme a été testé seulement sur des réseaux synthétiques dont la taille ne dépasse pas 2500 nœuds.

En conséquence, les points forts de notre algorithme vis-à-vis de ces algorithmes peuvent être résumés par :

- notre algorithme est destiné sans restriction à tous types de réseaux,
- utilise une petite population d'individus,
- il ne requiert pas une connaissance préalable du nombre de communautés recherchées et
- il a été testé avec succès sur de grands réseaux.

5.5 Expérimentation de la méthode proposée

5.5.1 Ensemble de données réelles

Pour évaluer l'algorithme sur les réseaux du monde réel, il n'y a pas de moyen largement accepté, permettant de comparer et d'apprécier la qualité d'une décomposition. Cela se fait habituellement en calculant la modularité. D'autres mesures peuvent alors être envisagées :

- Vad (vertex average degree) qui mesure le degré moyen d'un sommet dans un ensemble de groupes S ; il est défini comme :

$$Vad(S) = \frac{2\sum_{C\in S}|E(C)|}{\sum_{C\in S}|C|}$$

  $E(C)$ représente l'ensemble des arêtes dont les sommets appartiennent à C.

- *Overlap* qui mesure le nombre moyen de communautés auxquelles appartient chaque sommet. C'est la somme des tailles de toutes les communautés (y compris les singletons), divisé par le nombre de sommets ($|V|$). Ceci est exprimé comme suit :

$$Overlap = \frac{\sum_{C \in S} |C|}{|V|}$$

Pour cette expérimentation, nous avons utilisé un réseau réel dont la structure est connue pour être difficile à comprendre, il s'agit du réseau Southern Women :

Le réseau Southern Women [Dav et al. 41], est un réseau social biparti comportant 18 nœuds femmes et 14 nœuds événements, avec 89 arêtes reliant les nœuds femmes et les nœuds événements. Un nœud femme est relié à un nœud événement si la femme a assisté à l'événement correspondant. Ce réseau a été beaucoup étudié par Davis dans le cadre d'une étude approfondie de classe et de race dans le Sud des USA.

Pour évaluer l'efficacité de l'algorithme d'optimisation locale de la modularité chevauchante, nous avons exécuté le logiciel avec ce réseau une dizaine de fois et avons reporté dans le tableau 5.7 les valeurs de la modularité de Newman du graphe-arête ($Q_{Graphe-arête}$), la modularité de Mancoridis (MQB_{Over} avant et après l'optimisation locale), ainsi que le nombre de communautés obtenues $Nbcom$ (avant et après l'optimisation locale). Nous y avons reporté également les valeurs du degré moyen Vad d'un nœud et le degré de chevauchement $Overlap$.

Les résultats obtenus (cf. tableau 5.7), indiquent une amélioration palpable sur la valeur de la modularité qui varie de 0.04 à 0.148. Les meilleures valeurs de la modularité sont obtenues avec des valeurs du Vad et de l'$Overlap$ ne dépassant pas 2.748 et 1.157 respectivement. Ceci indique qu'il n'y a pas un grand chevauchement dans le réseau considéré. En d'autres termes, particulièrement pour le réseau testé, les communautés significatives ne sont pas forcément très chevauchantes.

Q Graphe-arête	MQBOver avant	Nbcom avant	MQBOver après	Nbcom après	Vad	Overlap
0.412	0.442	10	0.560	4	3.953	1.343
0.337	0.480	11	0.521	4	3.333	1.125
0.334	0.318	9	0.431	3	4.410	1.219
0.471	0.449	11	0.508	4	3.189	1.156
0.384	0.437	10	0.580	3	4.160	1.125
0.391	0.410	9	0.477	4	3.953	1.343
0.395	0.352	6	0.401	4	3.333	1.218
0.457	0.472	10	0.620	4	2.748	1.157
0.503	0.538	8	0.567	4	3.426	1.213
0.561	**0.545**	**10**	**0.652**	**4**	**2.081**	**1.156**

Tableau 5.7. Résultats de dix exécutions avec le réseau *Southern Women*

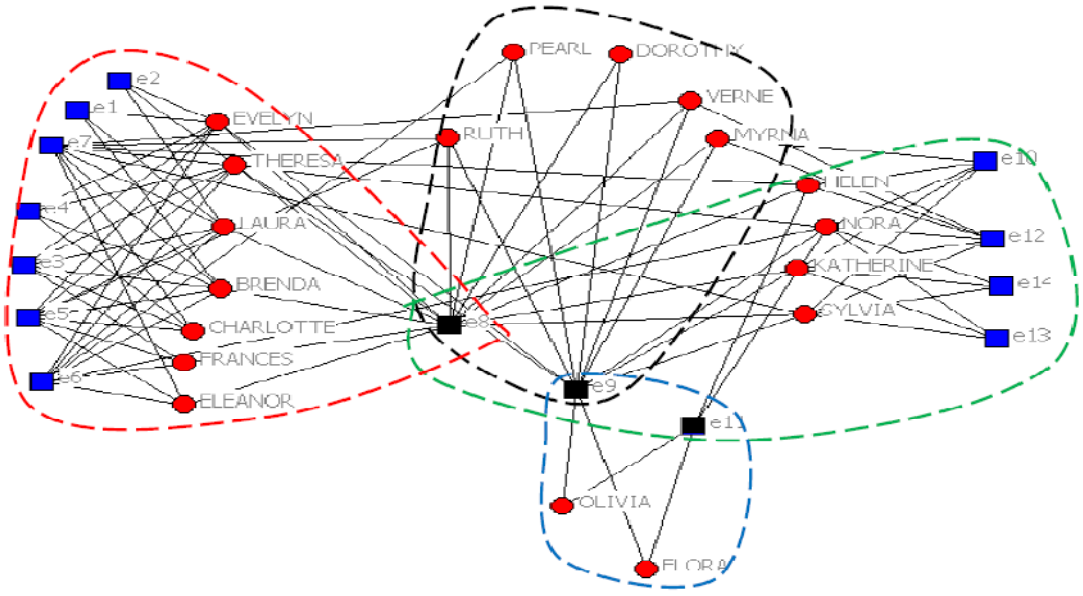


Fig 5.7. Structure communautaire du réseau *Southern Women* pour $MQB_{over} = 0.652$

La meilleure décomposition trouvée comporte quatre communautés et une modularité $MQB_{Over} = 0.652$, $Vad = 2.081$ et $Overlap = 1.156$ (cf. figure 5.7). Chaque communauté est encerclée par des pointillés colorés. Les nœuds qui chevauchent entre plusieurs communautés sont colorés en noir. On voit bien que l'événement e8 appartient à trois communautés (rouge, noir et verte) et e9 appartiennent à 2 communautés (noir et verte), donc ces événements attirent le plus de femmes (14 femmes pour e8 et 10 pour e9). Flora et Olivia préfèrent aller qu'aux deux clubs

e11 et e9, donc la communauté (bleu) : (Flora Olivia e9 et e11) forme un groupe cohérent. De même, la communauté colorée en noir, comporte des femmes qui sont reliées pratiquement qu'aux deux événements e8 et e9 appartenant à cette communauté. La communauté colorée en rouge est constituée des femmes qui sont pratiquement reliées qu'aux événements (e1, e2, e3, e4, e5, e6 et e8). La communauté en vert est constituée de femmes qui sont reliées presque qu'aux événements e10, e12, e13, e14, e11, e8 et e9.

L'analyse présentée précédemment concernant les résultats obtenus sur le réseau S*outhern Women*, montre la capacité de notre approche à détecter des communautés mixtes chevauchantes significatives. La structure communautaire ainsi découverte, suggère des particularités inattendues pour certains nœuds du réseau, et exprime clairement les traits distinctifs des communautés.

5.5.2 Ensemble de données synthétiques

Soient *D1* et *D2* deux décompositions d'un réseau en communautés recouvrantes (chevauchantes). On note par $|D1|$ et $|D2|$, les nombres de communautés dans *D1* et *D2*, respectivement.

Pour mesurer la similarité entre les deux décompositions *D1* et *D2*, on utilise *l'information mutuelle normalisée* notée $N(D1, D2)$ qui a été adaptée aux communautés chevauchantes dans [Man et al. 98] ; elle est définie par la formule suivante :

$$N(D1, D2) = 1 - \frac{1}{2}[H(D1|D2)_{norm} + H(D2|D1)_{norm}]$$

tel que $H(D1|D2)_{norm} = \frac{1}{2}\sum_{k=1}^{|D1|} \frac{H(D1_k|D2)}{-p_{D1}\log(p_{D1})}$ et $H(D2|D1)_{norm} = \frac{1}{2}\sum_{k=1}^{|D2|} \frac{H(D2_k|D1)}{-p_{D2}\log(p_{D2})}$

Le terme p_{D1} représente la fraction de nœuds contenus dans la décomposition *D1*. La sous-expression $-p_{D1}\log(p_{D1})$ représente l'entropie[17] de *D1*. Enfin, la sous-expression $H(D1_k|D2)$ indique l'entropie conditionnelle d'une communauté $D1_k \in D1$ par rapport à la décomposition *D2*.

[17] L'*entropie* caractérise le "désordre" possible d'un système.

Nous avons généré deux types de réseaux synthétiques bipartis (groupes A et B) dont les communautés sont représentées par des carrés dans la figure 5.8. Celles-ci seront considérées comme des communautés "cibles" dans cette expérimentation.

- pour le groupe *A*, le réseau est constitué de *4(m+r)/4(n+s)* nœuds [*4(m+r)* nœuds rouges et *4(n+s)* nœuds bleus], répartis en 4 communautés disjointes de *m+r /n+s* nœuds chacune, plus une cinquième communauté de *4r/4s* nœuds pris dans les 4 premières communautés (on prend r/s dans chacune des 4 communautés disjointes), avec bien sûr, $m \geq r$ et $n \geq s$.

- Pour le groupe *B*, le réseau est composé de *5m+4r /5n+4s* nœuds, répartis en 4 communautés disjointes de *m+r/n+s* nœuds, plus une cinquième communauté constituée de *m+4r/n+4s* nœuds dont *r/s* nœuds sont pris dans chacune des 4 premières communautés et *m/n* nœuds qui lui sont propres.

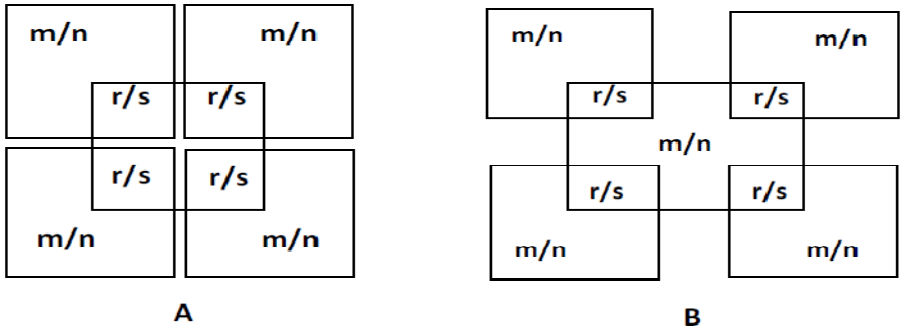


Fig 5.8. Schéma des deux types de réseaux bipartis utilisés

Nous avons implémenté une procedure qui génère automatiquement ces deux types de réseaux selon le procédé suivant : Pour chaque communauté on tire au hasard un nœud rouge et un nœud bleu, et ensuite on les relie par une arête avec une probabilité *pi*. Cette opération est réalisée un certain nombre de fois pour la même communauté. On obtient ainsi des réseaux bipartis qui présentent des communautés plus ou moins denses. Lorsque la probabilité *pi* tend vers 1, les graphes associés à ces communautés tendent à être complets. Un exemple de ces réseaux est présenté dans la figure 5.9 et 5.10.

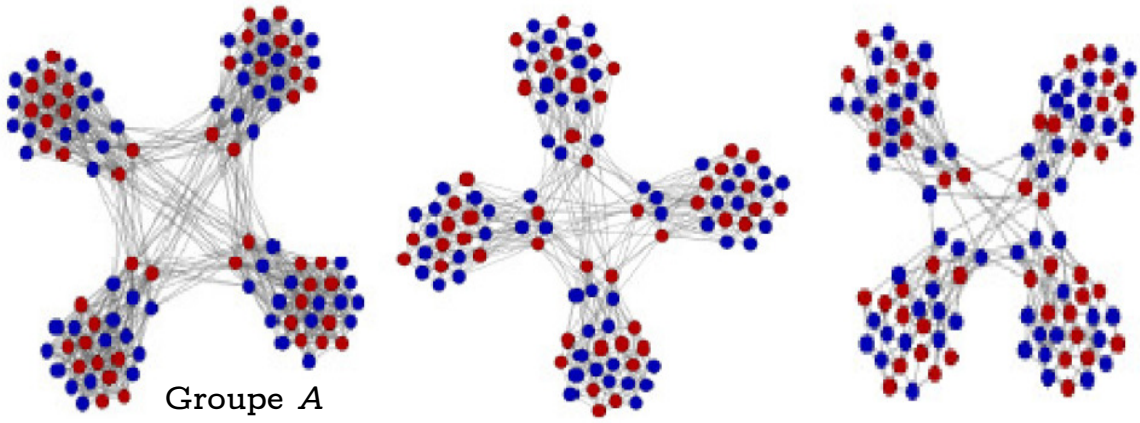


Fig 5.9. Réseaux synthétiques du groupe A pour les valeurs de m=12, n=10, r=3 et s=2 et pour une probabilité pi=1, 0.7 et 0.4

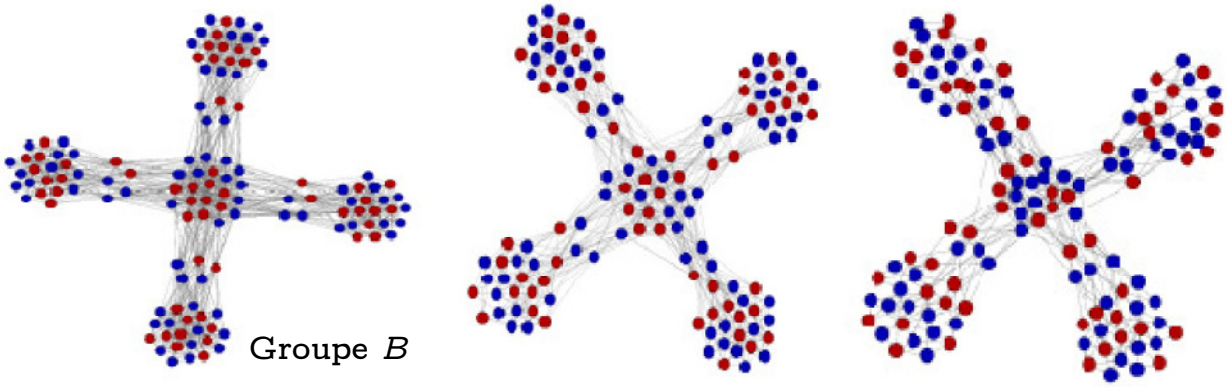


Fig 5.10. Réseaux synthétiques du groupe B pour les valeurs de m=12, n=10, r=3 et s=2 et pour une probabilité pi=1, 0.7 et 0.4

- Nous testons notre approche sur deux réseaux du groupe A : un réseau de 320/200 nœuds et un réseau de 140/140 nœuds, ce dernier possède des classes carrées (le nombre de nœuds rouges est le même que celui des nœuds bleus).

- De même, nous testons notre approche sur deux réseaux du groupe B : un réseau de 370/240 nœuds et un réseau de 160/120 nœuds. Ce dernier possède aussi des communautés carrées.

Durant cette expérimentation, nous avons généré des réseaux pour des valeurs de pi allant de 0,3 à 1. Sur chacun des réseaux, l'algorithme génétique a été exécuté une dizaine de fois. La meilleure partition a été prise, elle correspond à celle qui a donné la plus grande modularité de Newman. Nous appliquons alors sur cette partition

l'algorithme d'optimisation de la modularité de Mancoridis et nous obtenons ainsi une décomposition en communautés chevauchantes. Les résultats obtenus sont comparés aux communautés cibles en calculant la valeur de l'information mutuelle normalisée. Les résultats sont reportés sur le graphique des figures 5.11 et 5.12.

Nous remarquons que l'information mutuelle normalisée s'approche de 1 au fur et à mesure que la probabilité *pi* tend vers 1. De plus, l'information mutuelle normalisée est meilleure dans les réseaux du type *B*. Par ailleurs, dans les deux types de réseaux, l'information mutuelle normalisée est meilleure pour les réseaux constitués de communautés carrées (courbes rouges sur les figures 5.11 et 5.12).

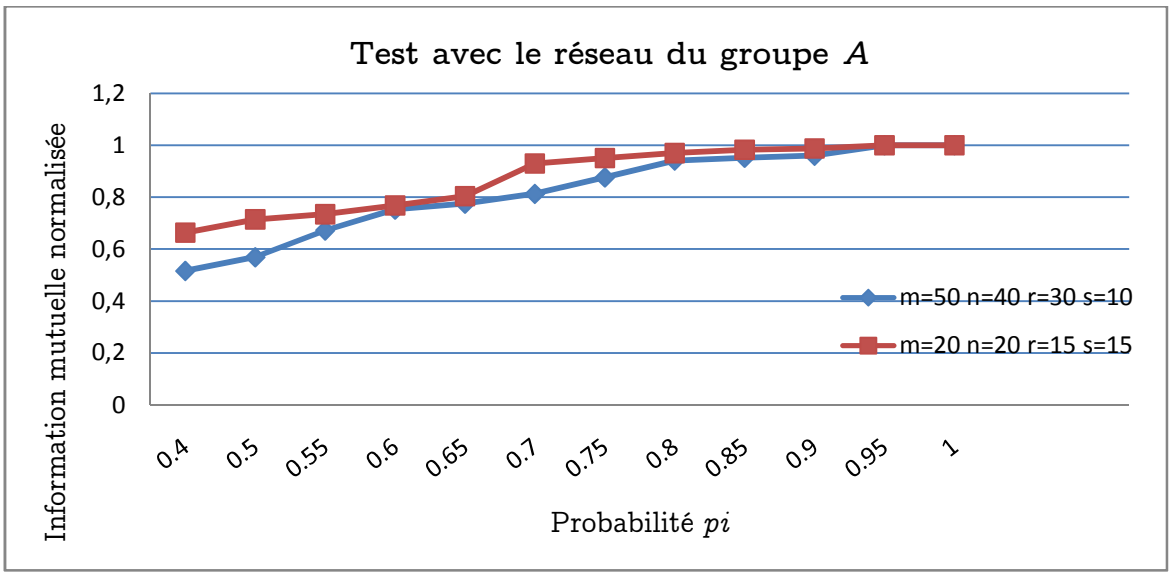


Fig 5.11. Résultats des tests effectués sur les réseaux bipartis synthétiques de type *A*

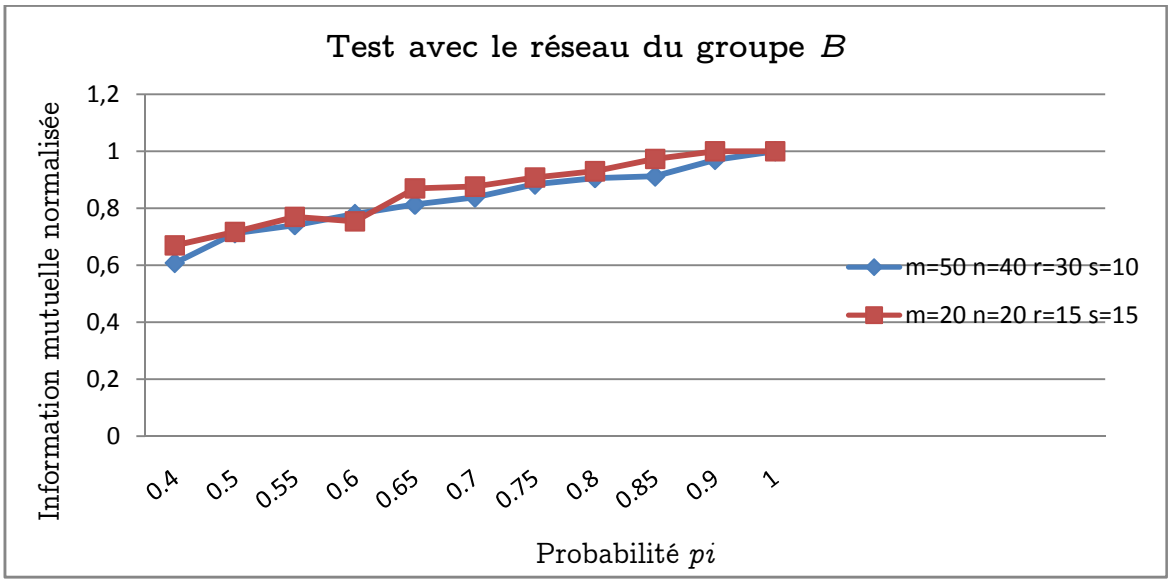


Fig 5.12. Résultats des tests effectués sur les réseaux bipartis synthétiques de type *B*

5.5.3 Mise en évidence de la viabilité de la méthode proposée

Une approche alternative à cette double optimisation de la modularité est de combiner les deux types d'optimisation (optimisation globale et locale) en une seule phase. En d'autres termes, à chaque génération de l'algorithme génétique on sélectionne le meilleur individu de la population. On applique alors une optimisation locale à la décomposition correspondant au meilleur individu. A la fin du processus évolutif nous choisissons la meilleure modularité parmi les modularités obtenues. Pour mesurer l'impact de la combinaison des deux phases, sur la modularité finale, on sauvegarde les modularités obtenues avant et après l'optimisation locale des différentes générations.

Pour ce nouveau test, nous réutilisons le réseau biparti *Southern Women*, décrit dans la sous-section 5.3.3. L'algorithme génétique a été exécuté avec les valeurs des paramètres définis dans le dernier paragraphe de la sous-section 5.2.3. L'objectif est de mettre en évidence l'évolution de la modularité après optimisation locale. Pour ce faire, nous avons utilisé les 10 meilleures modularités obtenues avant l'optimisation locale et les 10 modularités correspondantes après optimisation locale (ce qui est illustré graphiquement dans la figure 5.13). Notons que cette expérience a été réalisée avec 300 générations de la population.

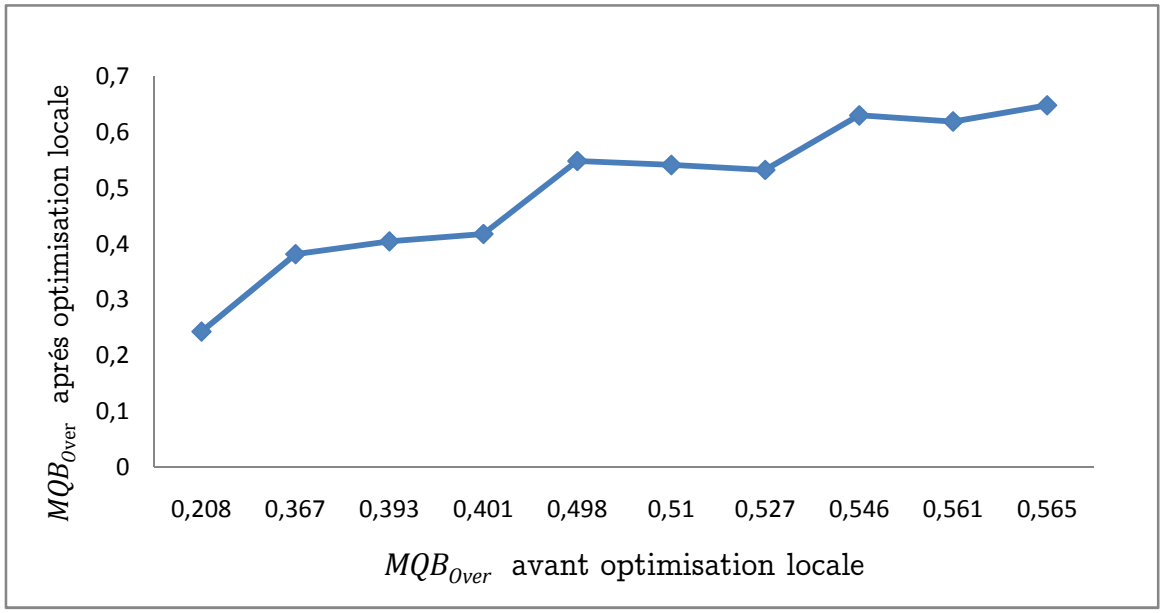


Fig 5.13. MQB_{Over} après l'optimisation locale en fonction de MQB_{Over} avant l'optimisation locale

En vertu de cette expérimentation, la meilleure modularité MQB_{Over} obtenue après l'optimisation locale est égale à 0.649. La modularité MQB_{Over} avant l'optimisation locale est égale à 0.565. Ce résultat est identique à celui que nous avons obtenu avec le procédé à deux phases. Par ailleurs, on observe sur la figure 5.13 que la modularité *après optimisation* (MQB_{Over} *après*) n'est pas une fonction monotone de la modularité *avant optimisation* (MQB_{Over} *avant*).

Nous avons aussi exécuté l'algorithme en se limitant seulement à 230 générations, au lieu de 300, le résultat ainsi retourné par la méthode basée sur deux phases est (MQB_{Over} *avant*, MQB_{Over} *après*) = (0.561, 0.62), alors que dans la méthode qui utilise une seule phase, nous obtenons (MQB_{Over} *avant*, MQB_{Over} *après*) = (0.546, 0.63) (ce qui est légèrement meilleur que le résultat précédent).

Du fait que la modularité *après optimisation* (MQB_{Over} *après*) n'est pas une fonction monotone de la modularité *avant optimisation* (MQB_{Over} *avant*), on peut effectivement parfois obtenir une légère amélioration de la modularité avec la méthode à une seule phase, mais ce n'est pas très significatif, vu la complexité temporelle relativement élevée de la méthode à une seule phase. Rappelons que la complexité de notre méthode (méthode basée sur deux phases) est $O((S+3\times Gen\_max)\times|E|+|V|)$ (cf. dernier paragraphe de la sous-section 4.6.3). Le procédé à une seule phase, possède, quant à lui, une complexité temporelle en

$O((S + 3 \times Gen\_max) \times |E| + Gen\_max \times |V|)$, puisque l'optimisation locale est appliqué à chaque génération de l'algorithme génétique. Par conséquent, avec les grands réseaux, le temps d'exécution est beaucoup plus long, et le résultat n'est pas significativement amélioré.

5.6 Conclusion

Dans ce chapitre, nous avons exposé les résultats des expérimentations que nous avons réalisées. Les tests ont porté sur les algorithmes proposés en utilisant deux jeux de données : réelles et synthétiques. Concernant l'algorithme évolutionnaire, les tests montrent clairement la capacité et la fiabilité de notre algorithme à détecter les communautés disjointes et placent notre algorithme parmi les meilleurs, en termes de temps de calcul et de la qualité des structures communautaires obtenues. En ce qui concerne l'approche globale, les tests réalisés ont porté sur des réseaux synthétiques et un réseau du monde réel, et ont montré l'efficacité et l'efficience de notre méthode à détecter des communautés chevauchantes mixtes dans des réseaux bipartis. L'analyse des communautés ainsi trouvées, montre que celles-ci sont porteuses de significations par rapport au réseau utilisé. Les communautés chevauchantes découvertes dévoilent ainsi des connaissances inconnues au préalable dans le réseau biparti étudié.

Conclusion et perspectives

Le travail réalisé dans le cadre de cette thèse s'inscrit dans le domaine de la détection de communautés dans les réseaux. Nous nous sommes particulièrement intéressés à la détection de communautés chevauchantes dans des réseaux bipartis. Notre approche est caractérisée par l'utilisation d'une optimisation globale suivie d'une optimisation locale.

Le premier type d'optimisation est l'œuvre d'un algorithme évolutionnaire permettant la détection de communautés disjointes de nœuds dans le graphe-arête. La détection est basée sur l'optimisation globale de la modularité de Newman. L'algorithme utilise les opérateurs génétiques classiques ainsi que deux autres opérateurs spécifiques. Il détermine de surcroit de manière automatique le nombre de communautés avec une complexité quasi linéaire par rapport au nombre total de liens dans le graphe. Nous avons expérimenté cet algorithme sur deux ensembles de données, en l'occurrence, un ensemble de données synthétiques et un ensemble de réseaux réels. Nous avons comparé nos résultats à ceux d'autres travaux existants dans le domaine. Les expérimentations effectuées ont montré l'efficacité et la capacité de cet algorithme à découvrir des structures communautaires ayant un score de qualité élevé. Nous avons constaté que l'algorithme présente un meilleur compromis qualité-temps d'exécution notamment dans le cas de grands réseaux.

Le second type d'optimisation consiste en un algorithme d'optimisation locale de la modularité de Mancoridis. Nous avons d'abord adapté cette modularité à la détection de communautés chevauchantes dans des réseaux bipartis, ensuite nous l'avons appliquée pour quantifier des structures communautaires chevauchantes. Nous avons, par ailleurs, pris en considération, dans l'algorithme, l'aspect sémantique pour aboutir à des communautés mixtes significatives. Nous avons, en outre, démontré l'efficience et l'efficacité de notre approche à détecter des communautés chevauchantes dans des réseaux bipartis, particulièrement sur des réseaux synthétiques dont les structures communautaires chevauchantes sont connues à priori. Nous avons aussi appliqué notre méthode à un réseau réel dont la structure est connue pour être difficile à comprendre. La structure communautaire ainsi découverte

pour ce réseau, suggère des particularités inattendues pour certains nœuds du réseau, et exprime clairement les traits distinctifs des communautés découvertes.

Enfin, les résultats expérimentaux présentés montrent que l'approche proposée dans cette thèse permet d'extraire des communautés chevauchantes mixtes. L'analyse des communautés ainsi découvertes montre que celles-ci dévoilent certaines significations non connues au préalable dans le réseau biparti étudié. Ces significations peuvent être qualifiées de connaissances relatives au réseau considéré.

Les travaux et résultats présentés dans cette thèse offrent naturellement de nombreuses perspectives. La première, en l'occurrence découle logiquement du fait que nous n'avons traité que des graphes non-orientés et non-pondérés, quoiqu'il existe d'autres types de graphes qui peuvent simuler d'autres situations. L'utilisation d'autres modèles de graphes, permettrait ainsi de considérer de nouveaux phénomènes, jusque-là non encore pris en compte par les algorithmes actuels de détection de communautés chevauchantes. Nous souhaitons dans un premier temps étendre l'approche proposée aux graphes pondérés et/ou orientés.

La deuxième perspective envisageable serait d'implémenter notre algorithme génétique avec une possibilité d'exécution parallèle. En effet, il serait souhaitable de trouver un modèle de parallélisation de cet algorithme pour pouvoir traiter des réseaux de très grandes tailles afin d'exhiber des solutions optimales en un temps raisonnable.

Une autre perspective, qui semble être, une continuité logique de ce travail, est l'adaptation de l'approche globale proposée pour détecter des communautés chevauchantes significatives dans des réseaux monopartis. Si l'intégration d'une certaine sémantique, qui se déduit du domaine auquel appartient le réseau étudié par exemple, s'avère possible, alors cela constituerait vraisemblablement une piste de recherche très intéressante.

Enfin, une piste qui nous paraît possible, serait de chercher comment exploiter la structure détectée dans une optique de réseaux dynamiques. En effet, on peut tirer profit de la structure communautaire chevauchante détectée, non seulement dans l'instant présent, mais également au cours de son évolution dans le temps. Les structures d'un réseau dynamique, en deux périodes consécutives de temps, possèdent généralement de grandes similitudes, et peuvent se déduire l'une de l'autre, vraisemblablement en exploitant les chevauchements découverts.

Liste de travaux scientifiques

Revues internationales

- Souam Fatiha, Aïtelhadj Ali, Baba Ali Riadh. (2013), "*Dual modularity optimization for detecting overlapping communities in bipartite networks*". In journal, Knowledge and Information Systems An International Journal, Online First, published April 17, (2013), 1-34, Springer-Verlag, DOI 10.1007/s10115-013-0644-8.

- Aîtelhadj Ali, Boughanem Mohand, Mezghiche Mohammed, Souam Fatiha. (2012), "*Using structural similarity for clustering XML documents*". In journal, Knowl Inf Syst, Volume 32, Issue 1, 109–139, Springer-Verlag.

Actes de conférences internationales

- Aïtelhadj Ali, Souam Fatiha, Mezghiche Mohammed. (2009), "*XML Documents Clustering Based on Structural Similarity*". In Proceedings of the Internatonal Conference IADIS, WWW/Internet, Rome, Italy, 19-22 November, pages 559-566.

- Aïtelhadj Ali, Mezghiche Mohamed, Souam Fatiha. (2009), "*RI structurée, RI et XML, RI précise*". Dans, Actes de la 6eme Conférence en Recherche d'Information et Applications Coria'2009, Presqu'île de Giens, France, 5–7 mai, pages 301–317.

Bibliographie

[Agg et al. 10] Aggarwal C.C and Wang H, (2010) *A Survey of Clustering Algorithms for Graph Data. Managing and Mining Graph Data*. A. K. Elmagarmid, Springer US, 40: 275-301

[Ahn et al. 10] Ahn Y.Y, Bagrow J.P and Lehmann S, (2010) *Link communities reveal multiscale complexity in networks*. Nature 466, 761–764.

[Ait et al. 09a] Aïtelhadj A, Mezghiche M, Souam F, (2009) *RI structurée, RI et XML, RI précise*. Actes de la 6^{eme} Conférence en Recherche d'Information et Applications Coria'2009, Presqu'ile de Giens, France, pages 301–317,

[Ait et al. 09b] Aïtelhadj A, Souam F and Mezghiche M, (2009) *XML documents based on clustering structural similarity.* Proceedings of the Internatonal Conference IADIS WWW/Internet, Rome, Italy, pages 559–566, ISBN: 978-972-8924-93-5.

[Ait et al. 12] Aïtelhadj A, Boughanem M, Mezghiche M, Souam F, (2012) *Using structural similarity for clustering XML*. Knowledge and Information Systems, An International Journal, Volume 32, Issue 1, 109–139 Springer-Verlag.

[Aub et al. 03] Auber D, Chiricota Y, Jourdan F, and Melançon G, (2003) *Multiscale visualization of small world networks*. Proceedings of the 9th IEEE Symposium on Information Visualization (InfoVis 2003), page 10, Seattle, USA, IEEE Computer Society.

[Aup et al. 03] Aupetit S, Monmarché N, Slimane M, Guinot C, and Venturini G, (2003) *Clustering and Dynamic Data Visualization with Artificial Flying Insect*. Genetic and Evolutionary Computation Conference (GECCO).

[Bâc Sch 93] Bâck T and Schwefel H.P, (1993) *An overview of evolutionary algorithms for parameter optimization*. Evolutionay Computation 1(1): 1-24.

[Bal et al. 95] Baluja S and Caruana R, (1995) *Removing the genetics from the standard genetic algorithm*. A Prieditis and S Russel, editors. The Int. Conf. on Machine Learning 1995, pages 38-46, San Mateo, CA, Morgan Kaufmann Publishers.

[Barber 07] Barber M.J, (2007) *Modularity and community detection in bipartite network*. Phys. Rev. E, vol. 76, no. 6, 066102, doi: 10.1103/PhysRevE.76.066102.

[Barnes 82] Barnes E.R, (1982) *An Algorithm for Partitioning the Nodes of a Graph*. SIAM Journal on Algebraic and Discrete Methods, 3(4):541-550.

[Bar Alb 99] Barabasi A-L and Albert R,(1999). *Emergence of scaling in random networks*. Science, 286: 509.

[Bau et al. 05] Baumes J, Goldberg M and Magdon-Ismail M, (2005) *Efficient Identification of Overlapping communities*. ISI 2005. LNCS, vol. 3495, pp. 27--36. Springer, Heidelberg

[Bea et al. 93] Beasley D, Bull D.R, and Martin R, (1993) *An overview of genetic algorithms: Part 2, research topics*. University Computing, 15(4):170 181.

[Berge 70] Berge C, (1970) *Graphes et hypergraphes*, Dunod Paris.

[Ber et al. 09] Berghman L, Goossens D and Leus R, (2009) *Efficient solutions for mastermind using genetic algorithms*. Computers & Operations Research, 36: 1880–1885.

[Blo Sen 02] Blondel V and Sennelart P, (2002) *Automatic extraction of synonyms in a dictionary*. Proceeding of the SIAM Workshop on Text Mining.

[Bra Erl 05] Brandes U and Erlebach T, (2005) *Network Analysis*. Methodological Foundations, volume 3418 of Lecture Notes in Computer Science. Springer.

[Bro et al. 89] Brown D.E, Huntley C.L and Spillane A.R, (1989) *A parallel genetic heuristic for the quadratic assignment problem*. Proceedings of the third international conference on Genetic algorithms, pages 406-415. Morgan Kaufmann Publishers Inc.

[Bor Sch 03] Bornholdt S and Schuster H.G, (2003) *Handbook of Graphs and Networks: From the Genome to the Internet*. Edition Wiley.

[Bou Aub 09] Bourqui R and Auber D, (2009) *Analysis of 4-connected Components Decomposition for Graph Visualization*. Technical report, LaBRI (http://www.labri.fr/).

[Cerf 94] Cerf R, (1994) *Une théorie asymptotique des algorithmes génétiques*, Thèse de Doctorat, Université de Montpellier II.

[Cic Smi 01] Cicirello V and Smith S, (2001) *Wasp-like Agents for distributed Factory Coordination*. Technical Report CMU-RI-TR-01-39, Robotics Institute, Carnegie Mellon University, Pittsburgh.

[Che Cha 09] Chen Y.H and Chang F.J, (2009) *Evolutionary artificial neural networks for hydrological systems forecasting*. Journal of Hydrology.

[Chu et al. 09] Chung J.W, Oh S.M and Choi I.C, (2009) *A hybrid genetic algorithm for train sequencing in the korean railway*. Omega, 37: 555–565.

[Cla et al. 04] Clauset A, Newman M.E.J and Moore C, (2004) *Finding Community Structure in Very Large Networks*. Phys. Rev. E 70, 066111. Availlable at http://link.aps.org/doi/10.1103/PhysRevE.70.066111

[Col et al. 92] Colorni A, Dorigo M, and Maniezzo V, (1992) *Distributed Optimization by Ant Colonies*. In Varela F. and Bourgine P, editors, Proceedings of ECAL'91 - First European Conference on Artificial Life, pages 134-142, Paris, France. Elsevier Publishing.

[Coo Hol 06] Cook D.J and Holder L.B, (2006) *Mining Graph Data*. Edition Wiley.

[Dav et al. 41] Davis A, Gardner B.B and Gardner M.R, (1941) *Deep South;a Social Anthropological Study of Caste and Class*. The University of Chicago Press, Chicago.

[Deb Gol 89] Deb K and Goldberg D.E, (1989) *An investigation of niche and species formation in genetic function optimization.* Proceedings of the third international conference on Genetic algorithms, pages 42-50. Morgan Kaufmann Publishers Inc.

[DeC Von 99] De Castro L and Von Zuben F, (1999) *Artificial Immune Systems Part I : Basic Theory and Applications*. Technical Report TR-DCA 01/99, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil.

[DeJ Spe 93] De Jong K.A and Spears W, (1993) *On the state of evolutionary computation*. Proceedings. of International Conference on Genetic Algorithms (ICGA'93), p. 618-623.

[Don Mun 05] Donetti L and Munoz M.A, (2005) *Improved spectral algorithm for the detection of network communities*. Modeling cooperative behavior in the social sciences, volume 779 : 104–107.

[Dré et al. 03] Drép J, Pétrowski A, Siarry P and Taillard E, (2003) *Méta-heuristiques pour l'optimisation difficile*. Edition Eyrolles.

[Dréo 04] Dréo J, (2004) *Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical.* Thèse de Doctorat de l'Université de Paris 12.

[Dongen 00] Dongen S.V, (2000) *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000.

[Dou et al. 07] Dourisboure Y, Geraci F and Pellegrini M, (2007) *classification of dense communities in the web*. Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada, ACM, pages 461-470.

[Duc Are 05] Duch J and Arenas A, (2005) *Community detection in complex networks using extremal optimization*. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 72(2): 027104, 2005.

[Durand 04] Durand N, (2004) *Algorithmes Génétiques et autres méthodes d'optimisations appliquées à la gestion de trafic aérien*. Thèse de doctorat, Thèse d'habilitation INPT.

[Ebe et al. 02] Ebel H, Mielsch L.I and Bornholdt S, (2002) *Scale-free topology of e-mail networks*. Physical Review E, 66.

[Ebe et al. 01] Eberhart R, Kennedy J, and Shi Y, (2001) *Swarm Intelligence*. Evolutionary Computation. Morgan Kaufmann.

[Eib et al. 95] Eiben A.E, van Kemenade C.H.M and. Kok J.N, (1995) *Orgy in the computer: Multi-parent reproduction in genetic algorithms*. F. Moran, A. Moreno, J. J; Merelo, and P. Chacon, editors, Procedings of the Third European Conference on Artificial life : Advances in Artificial Life, volume 929 of LNAI, pages 934 945 Springer-Verlag.

[Eib et al. 99] Eiben Á.E, Hinterding R and Michalewicz Z, (1999) *Parameter Control in Evolutionnary Algorithms*. IEEE Transaction on Evolutionary Computation, 3(2):124–141.

[Erd Rén 59] Erdös P, Rényi A, (1959) On Random Graphs. Publicationes Mathematicae, (6): 290–297.

[Esh et al. 89] Eshelman L.J, Caruana R.A and Shaffer J.D, (1989) *Biases in the crossover landscape*. Proceedings of the international conference on Genetic algorithms, pages 10-19.Morgan Kaufmann Publishers Inc.

[Esh Sch 91] Eshelman L.J and Schaffer J.D, (1991) *Preventing premature convergence in genetic algorithms by preventing incest.* In Fourth International Conference on Genetic Algorithms (ICGA), pages 115 122, San Mateo, Morgan Kaufmann.

[Eva Lam 09] Evans T.S and Lambiotte R, (2009) *Line graphs, link partitions and overlapping communities.* Phy. Rev. E, vol.80, p.016105.

[Fal et al. 99] Faloutsos M, Faloutsos P and Faloutsos C, (1999) *On power-law relationships of the internet topology*. Proceedings og the internatonal ACM conference SIGCOMM, pages 251-262.

[Far et al. 07] Farkas I, Abel D, Palla G and Vicsek T, (2007) *Weighted network modules*. New J. Phys. 9, 6, 180.

[Fei Fel 63] Feigenbaum E.A and Feldman J, (1963) *Computers and Thought.* McGraw-Hill Book Company, New York

[Fer et al. 01] Ferrer R, Cancho I and Solé R.V, (2001) *The small-world of human language*. Technical report, Santa Fe Working paper 01-03-016.

[Fla et al. 00] Flake G.W, Lawrence S and Giles C.L, (2000) *Efficient identification of Web communities.* Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, Massachusetts, United States, ACM, pages 150-160.

[Fog et al. 66] Fogell J, Owens A.J and Walsh M.J, (1966) *Artificial Intelligence through Simulated Evolution*. Edition Wiley.

[For et al. 04] Fortunato S, Latora V and Marchior M, (2004) *Method to find community structures based on information centrality*. Phy. Rev. E, 70(5): 056104.

[Fortuna 10] Fortunato S, (2010) *Community detection in graphs*. ArXiv:0906.0612v2 [physics.soc-ph], Physics Reports, 486(3-5): 75–174. Availlable http://arxiv.org/abs/0906.0612

[Fortuna 08] Fortunato S, (2008) http://sites.google.com/site/santofortunato/inthepress2/

[Gal Hao 99] Galinier P and Hao J.K, (1999) *Hybrid evolutionary algorithms for graph coloring*. Journal of Combinatorial Optimization.

[Gir New 02] Girvan M and Newman MEJ, (2002) *Community Structure in Social and Biological Networks*. Proceedings of the National Academy of Science (PNAS). USA 99: 7821–7826.

[Glover 86] Glover F, (1986) *Future paths for integer programming and links to artificial intelligence*. Computers and Operations Research 13: 533-549.

[Goldber 89] Goldberg D, (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Welsey.

[Goldber 94] Goldberg D.E, (1994) *Algorithmes génétiques. Exploration, optimisation et apprentissage automatique.* Addison-Wesley France.

[Gregory 07] Gregory S, (2007) *An algorithm to find overlapping community structure in networks*. Proc. PKDD Conf. 91–102.

[Gregory 08] Gregory S, (2008) *A Fast Algorithm to Find Overlapping Communities in Networks*, PKDD 2008. LNAI, vol. 5211, pp. 408--423. Springer, Heidelberg.

[Gregory 09] Gregory S, (2009) *Finding Overlapping Communities Using Disjoint Community Detection Algorithms*. Complex Networks: results of the 1st International Workshop on Complex Networks (CompleNet 2009), pp. 47–61 Springer.

[Gregory 10] Gregory S (2010) *Finding overlapping communities in networks by label propagation.* New Journal of Physics. 12 103018 2010. Availlable at http://arxiv.org/abs/0910.5516v3

[Gui Lat 06] Guillaume J-L and Latapy M. (2006) *Bipartite graphs as models of complex networks*. Physica A, 371 :795-813.

[Gui Ama 05] Guimera R and Amaral Nunes L.A, (2005) *Functional cartography of complex metabolic networks*. Nature, 433: 895-900.

[Gui et al. 07] Guimera R, Sales-Pardo M and Amaral L. A. N, (2007) *Module identification in bipartite and directed networks*. Phys. Rev. E, vol. 76, no. 3, 036102, doi: 10.1103 /PhysRevE. 76.036102.

[Hasting 06] Hastings M.B, (2006) *Community detection as an inference problem*. Physical Review E, 74(3):035102.

[Hir et al. 90] Hirata H, Koakutsu S and Sugai Y, (1990) *Block placement by improved simulated annealing based on genetic algorithm*. Institute of Electronics, Information and Communication Engineers of Japan, volume 173 A. pages 87 94.

[Hof Bac 91] Hoffmeister F and Back T, (1991) *Genetic algorithms and evolution strategies: Similarities and differences*. Hans-Paul Schwefel and Reinhard Manner, editors, Parallel Problem Solving from Nature, volume 496 of Lecture Note in Computer Science, pages 455 470, Springer.

[Holland 75] Holland J, (1975) *Adaption in Natural and Artificial Systems, An Introductory Analysis with Applications to Biology,* Control and Artificial Intelligence. The University of Michigan, 1st edition.

[Hu et al. 08] Hu Y, Chen H, Zhang P, Li M, Di Z and Fan Y, (2008) *A new comparative definition of community and corresponding identifying algorithm.* ArXiv: 0802.0242v1 [physics.soc-ph].

[Jaw Bal 09] Jawahar N and Balaji A.N, (2009) *A genetic algorithm for the two-stage supply chain distribution problem associated with a fixed charge.* European Journal of Operational Research, 194: 496–537

[Jeo et al. 00] Jeong H, Tombor B, Albert R, Oltvai Z, and Barabasi A, (2000) *The large-scale organization of metabolic networks.* Nature, 407, 651.

[Kelley 09] Kelley S, (2009) *The existence and discovery of overlapping communities in large-scale networks.* Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY.

[Kel et al. 11] Kelley S, Goldberg M, Magdon-Ismail M, Mertsalov K, and Wallace A, (2011) *Handbook of Optimization in Complex Networks.* Springer, Chapter 6.

[Ker Lin 70] Kernighan B.W and Lin S, (1970) *An efficient heuristic procedure for partitioning graphs.* Bell System Technical Journal, 49(2):291-308.

[Kle et al. 99] Kleinberg J, Kumar R, Raghavan P, Rajagopalan S and Tomkins A, (1999) *The Web as a Graph: Measurements, Models, and Methods.* Computing and Combinatorics. Springer-Verlag, 1627: 1-17.

[Kim et al. 08] Kim J.S, Lee J.W, Noh Y.K, Park J.Y, Lee D.Y, Yang K.A, Chai Y.G, Kim J.C, and Zhang B.T, (2008) *An evolutionary monte carlo algorithm for predicting dna hybridization.* Biosystems, 91: 69–75.

[Koa et al. 96] Koakutsu S, Kang M, and Dai W.W.M, (1996) *Genetic simulated annealing and application to non-slicing floorplan design.* 5*th* ACM/SIGDA Physical Design Workshop, pages 134 141, Virginia, USA.

[Koza 92] Koza J.R, (1992) *Genetic Programming : on the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, Massachusetts.

[Kum et al. 8] Kumpula J.M, Kivelä M, Kaski K, and Saramäki J, (2008) *Sequential algorithm for fast clique percolation.* Phys. Rev. E 78, 2, 026109.

[Lan et al. 09] Lancichinetti A, Fortunato S, and Kertesz J, (2009) *Detecting the overlapping and hierarchical community structure in complex networks.* New Journal of Physics, vol. 11, no. 3, p. 033015.

[Lar Loz 01] Larrañaga P and Lozano J.A, (2001) *Estimation of Distributions Algorithms, a new tool for evolutionary computation*, Kluwer Academic Publishers.

[Lip Mil 09] Lipczak M, Milios E (2009) *Agglomerative Genetic Algorithm for Clustering in Social Networks Genetic and evolutionary computation.* Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO 2009 Montral Canada, pp 1243–1250.

[Liu et al. 07] Liu X, Li D, Wang S, and Tao Z, (2007) *Effective algorithm for detecting community structure in complex networks based on GA and clustering*. Proceedings of the 7th international conference on Computational Science ICCS '07, Part II (Springer-Verlag, Berlin, Heidelberg), pp. 657-664.

[Liu Mor 09] Liu X and Murata T, (2009) *Community Detection in Large-scale Bipartite Networks*. IEEE/WIC/ ACM–International Conference on Web Intelligence and Intelligent Agent Technology.

[Lus et al. 03] Lusseau D, Schneider K, Boisseau O.J, Haase P, Slooten E and Dawson S.M, (2003) *The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations*. Behavioral Ecology and Sociobiology 54, 396-405.

[Madelin 02] Madeline B, (2002) *Algorithmes évolutionnaires et résolution de problèmes de satisfaction de contraintes en domaines finis*. Thèse doctorat de l'université de Nice-Sophia Antipolis.

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

[Mau Res 00] Mauricio G, and Resende C, (2000) *Detecting dense subgraphs in massive graphs*. 17th international Symposium on Mathematical Programming.

[Mic Fog 98] Michalewicz Z and Fogel D.B, (1998) *How to Solve It*. Modern Heuristics. Springer, 2nd edition.

[Mohar 97] Mohar B, (1997) *Some applications of Laplace eigenvalues of graphs*. Graph Symmetry: Algebraic Methods and Applications, volume 497 de NATO ASI Series C.

[Moh et al. 08] Mohebbi M, Barouei J, Akbarzadeh T.M.R, Rowhanimanesh A.R, Habibi-Najafi M.B and Yavarmanesh M, (2008) *Modeling and optimization of viscosity in enzyme-modified cheese by fuzzy logic and genetic algorithm*. Computers and Electronics in Agriculture, 62: 260–265.

[Mos Nor 89] Moscato P and Norman M, (1989) *A competition- cooperative approach to complex combinatorial search*. Technical Report 790, Caltech Concurrent Computational Program.

[Nan et al. 08] Nan Du, Bin Wu, Bai Wang, and Yi Wang, (2008) *Overlapping Community Detection in Bipartite Networks*. arXiv : 0804.3636v1 [physics.soc-ph].

[Newman 01] Newman M.E.J, (2001) *Scientific collaboration networks*: *Ii. shortest paths, weighted networks, and centrality*. Phys. Rev. E, 64.

[New Gir 04] Newman M.E.J, Girvan M, (2004) *Finding and evaluating community structure in networks,* Physical Review E, 69:026113.

[Newman 04] Newman M.E.J, (2004) *Fast algorithm for detecting community structure in networks*. Phys. Rev. E 69(6): 066133.

[New et al. 57] Newwell A, Shaw J.C and Simon H.A, (1957) *Elements of a Theory of Human Problem Solving*. Rand Corporation.

[Pal et al. 05] Palla G, Derényi I, Farkas I, and Vicsek T, (2005) *Uncovering the overlapping community structure of complex networks in nature and society*. Nature 435, 814–818.

[Pizzuti 08] Pizzuti C, (2008) *Ga-net: A genetic algorithm forcommunity detection in social networks.* PPSN, volume 5199 of Lecture Notes in Computer Science, pages 1081–1090. Springer.

[Pol et al. 08] Poli R, Langdon W.B and Freitag McPhee N, (2008) *A Field Guide to Genetic Programming*. ISBN 978-1-4092-0073-4

[Pon Lat 05] Pons P and Latapy M, (2005) *Computing Communities in Large Networks Using Random Walks.* Computer and Information Sciences - ISCIS 2005. P. Yolum, T. Güngör, F. Gürgen and C. Özturan, Springer Berlin / Heidelberg. **3733:** 284-293.

[Pons 07] Pons P, (2007) *Détection de communautés dans les grands graphes de terrain*. Thèse de doctorat, Université Paris 7.

[Rad et al. 04] Radicchi F, Castellano C, Cecconi F, Loreto V, and Parisi D, (2004) *Defining and identifying communities in networks*. Proceedings of the National Academy of Sciences, vol. 101, no.9, 2658-2663.

[Rag et al. 07] Raghavan U.N, Albert R, Kumara S, (2007) *Near linear time algorithm to detect community structures in large-scale networks*. Phys. Rev. E 76 036106 Availlable http://arxiv.org/abs/0709.2938.

[Ram Gre 93] Ramsey C.L and Grefenstette J.J, (1993) *Case based initialization of genetic algorithms*, Stephanie Forrest, editor, Proc, of the Fifth Int. Conf. on Genetic Algorithms, pages 84 91, San Mateo, CA,. Morgan Kaufmann.

[Rechen 73a] Rechenberg I, (1973) *Evolutionsstrategie werkstatt bionic und Evolutionstechnik*, ISBN: 9783772816420.

[Rechen 73b] Rechenberg I, (1973) *Evolutions strategies*. Friedrich Frommann, Verlag (Günther Holzboog KG), Stuttgart.

[Rib Mac 94] Ribeiro C.C and Maculann N, (1994) *Applications of combinatorial optimization*. Annals of Operations Research 50, 1994.

[Rom Slo 09] Rom W.O and Slotnick S.A, (2009) *Order acceptance using genetic algorithms*. Computers & Operations Research, 36: 1758–1767.

[Schaeff 07] Schaeffer S.E, (2007) *Graph clustering*. Computer Science Review, 1(1):27-64.

[Schwef 81] Schwefel H.-P, (1981).*Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA.

[Sch Mic 97] Schoenauer M and Michalewicz Z, (1997) *Evolutionary computation: an introduction*. Control and Cybernetics 26(3): 307-338.

[Scott 00] Scott J.P, (2000) *Social Network Analysis: A Handbook*. Sage Publications Ltd.

[Seb et al. 97] Sebag M, Schoenauer M and Ravise C, (1997) *Revisiting the memory of evolution*. Thomas Bäck, editor, Prroceedings of the Seventh International Conference on Genetic Algorithms (ICGA97), San Francisco. CA, 1997. Morgan Kaufmann.

[Sil et al. 09] Silva T.P.C, Silva de Moura E, Cavalcanti J.M.B, da Silva A.S, Gomes de Carvalho M and Gonçalves M.A, (2009) *An evolutionary approach for combining different sources of evidence in search engines*. Information Systems, 34: 276–289.

[Smi Tim 08] Smith S.L and Timmis J, (2008) *An immune network inspired evolutionary algorithm for the diagnosis of parkinson's disease*. Biosystems, 94: 34–46.

[Sol et al. 09] Soleimani H, Golmakani H.R, and Salimi M.H, (2009) *Markowitz-based portfolio selection with minimum transaction lots, cardinality constraints and regarding sector capitalization using genetic algorithm*. Expert Systems with Applications, 36: 5058–5063.

[Sou et al. 13] Souam F, Aïtelhadj A and Baba Ali R, (2013) *Dual modularity optimization for detecting overlapping communities in bipartite networks.* Knowledge and Information Systems, An International Journal, online First published, avril 17, 2013, Springer-Verlag, DOI 10.1007/s10115-013-0644-8.

[Syswer 89] Syswerda G, (1989) *Uniform crossover in genetic algorithms*, J.D Shaeffer, editor, Proceeding of Third International Conference on Genetic Algorithms and Their Applications, pages 29.

[Tas et al. 07] Tasgin M, Herdagdelen A and Bingol H, (2007) *Community detection in complex networks using genetic algorithms*. eprint arXiv: 0711.0491.

[Ter Gas 01] Terfloth L and Gasteiger J, (2001) *Neural networks and genetic algorithms in drug design*. Drug Discovery Today, 6: 102–108.

[Tho et al. 00] Thomsen R, Rickers P and Kring T, (2000) *A religion based special model for evolutionary algorithms*. Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yaho, Evelyne Lutton, Juan Julian Merelo, Hans Paul Schwefel, editor, Parallel Problem Solving from Nature PPSN VI 6th International Conference, Paris, France, 2000, Springer Verlag.

[Wak Tsu 07] Wakita K and Tsurumi T, (2007) *Finding Community Structure in a Mega-scale Social Networking Service.* Proceedings of the International Conference on WWW/Internet 2007 IADIS'2007, pp 153–162.

[Wat Str 98] Watts D. J and Strogatz S.H, (1998) *Collective dynamics of 'small-world' networks*. Nature 393, 440-442.

[Wei et al. 11] Weihua Z, Zhongzhi Z, Jihong G and Shuigeng Z, (2011) *Evolutionary method for finding communities in bipartite networks*. arXiv :1011.3315v3 [Physics.data-an] availlable http://arxiv.org/pdf/1011.3315v3.pdf

[Yang 02] Yang S, (2002) *Adaptive Non-Uniform Crossover Based on Statistics for Genetic Algorithms*. Proceedings of the Genetic and Evolutionnary Computation Conference, pages 650–657.

[Yen et al. 05] Yen L, Wouters F, Fouss F, Verleysen M, and Saerens M, (2005) *Clustering using a random walk based distance measure*. Proceedings of the 13th Symposium on Artificial Neural Networks (ESANN 2005), pages 317-324.

[Zachary 77] Zachary W, [1977] *An information flow model for conflict and fission in small groups*. Journal of Anthropological Research, vol. 33, no.4, pp. 452–473.

[Zha et al. 07] Zhang S, Wang R and Zhang X, (2007) *Identification of Overlapping Community Structure in Complex Networks Using Fuzzy C-means Clustering.* Physica A 374, 1, 483-490.

[Zha et al. 09] Zhang Y, Wang J, Wang Y, and Zhou L, (2009) *Parallel community detection on large networks with propinquity dynamics*. Proceding SIGKDD Conf. 997–1006.

[Zho Lip 04] Zhou H and Lipowsky R, (2004) *Network Brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities*. International Conference on Computational Science, pages 1062-1069.

[Zio et al. 09] Zio E, Baraldi P and Pedroni N, (2009) *Optimal power system generation scheduling by multi-objective genetic algorithms with preferences*. Reliability Engineering & System Safety, 94: 432–444.