Community detection : computational complexity and approximation

Thomas Pontoizeau

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Université Paris-Dauphine

Community detection: computational complexity and approximation

École doctorale de Dauphine – ED 543

Spécialité INFORMATIQUE

Soutenue par **Thomas PONTOIZEAU**
le **04/06/2018**

Dirigée par **Cristina BAZGAN**

COMPOSITION DU JURY :

Cristina BAZGAN
Université Paris-Dauphine
Directrice de thèse

Michel HABIB
Université Paris Diderot
Rapporteur

Ioan TODINCA
Université d'Orléans
Rapporteur

Henning FERNAU
Universität Trier
Examinateur

Rodolphe GIROUDEAU
Université de Montpellier
Examinateur

Aline PARREAU
Université Claude Bernard Lyon 1
Examinatrice

Yann VAXES
Université d'Aix-Marseille
Président du Jury

LAMSADE UMR CNRS 7243 — DAUPHINE UNIVERSITÉ PARIS — PSL RESEARCH UNIVERSITY PARIS

A mon frère Arnaud,

Remerciements

Après avoir écrit une thèse sur les communautés, il fallait que je remercie toutes celles qui ont partitionné mon entourage, durant les trois ans et demi qu'a demandé son écriture.

Je voudrais remercier en tout premier lieu Cristina Bazgan, qui a eu confiance en moi et qui m'a accompagné du début à la fin de cette thèse. Merci beaucoup de m'avoir soutenu, écouté, suivi. Merci pour ta bienveillance, ta confiance et ta présence.

I also thank Janka Chlebíková who welcomed me warmly in Portsmouth. Thank you for your support, your (very) rigorous follow-up, and especially your kindness. [1]

Je remercie Ioan Todinca et Michel Habib pour avoir lu attentivement l'ensemble de ma thèse. Merci beaucoup pour toutes vos remarques. Je remercie également les autres membres du jury, Henning Fernau, Rodolphe Giroudeau, Aline Parreau et Yann Vaxès, pour l'intérêt qu'ils ont porté à mes travaux.

Je veux remercier aussi Daniel Vanderpooten pour m'avoir accueilli au sein du Master MODO et de m'avoir accompagné jusqu'à la thèse avec bienveillance.

Je remercie toute l'équipe du LAMSADE au sein de laquelle je me suis senti bien durant toutes mes années de thèse. Vous croiser dans les couloirs chaque jour fut toujours un plaisir.

J'ai une pensée particulière pour Anita Souissi, Stéphane Boucheron et Henning Bruhn-Fujimoto qui ont tous les trois été des professeurs formidables durant mon parcours scolaire et qui ont sans doute contribué indirectement à la réalisation de cette thèse.

Je remercie Clément Dallard, avec qui j'ai eu l'occasion de travailler régulièrement. Merci de m'avoir fait partager ta passion pour le snooker, pour ton amitié et pour ton humour si singulier. Des bisous à Angie. Grazie ad Alessio Petrozziello e Francesca Picariello per aver condiviso tanto buon umore (e buona cucina !) durante il mio soggiorno a Portsmouth. [2]

Merci à toi Ian mon clé, qui a supporté avec une grande patience mes cocasseries. Merci d'avoir apporté cette atmosphère si particulière en C605. Si ces années furent le feu, c'est sans doute grâce à toi.

أتقدم بالشكر إلى مريم، الصديقة المخلصة و أتمنى أن هذه الصداقة ستستمر طويلا بعد إنتهاء الأطروحة. أشكر
أيضا ياسين ،إبن بلدي، على طيبته،عفويته، و صدقه. شكرا لكما لإعطائي صورة جميلة لتونس و الإستظافة الطيبة [3]

1. *Je remercie également Janka Chlebíková qui m'a accueilli chaleureusement à Portsmouth. Merci pour ton soutien, ton suivi (très) rigoureux, et surtout ta gentillesse.*
2. NB : tous les mots sont transparents.
3. *Je remercie Meriem, une amie sincère que j'espère continuer de voir après la thèse, ainsi que Yassine mon compatriote, merci pour ta gentillesse, ta spontanéité et ton honnêteté. Merci à vous deux qui m'ont donné une si bonne image de la Tunisie et qui m'y ont si bien accueilli* (un grand merci à Raja pour sa traduction !).

Je tiens également à remercier personnellement mes autres collègues du bureau C605 auxquels je me suis attaché. Khalil, pour ton amitié et toutes ces longues discussions sur tes projets de permaculture au Kirghizistan. Céline, pour ton sourire toujours présent et tes conseils bienveillants. Ioannis, για τιν καλί σας διάτηεσι και ειδικά για το σηιούμορ σας. [4] Boris, bientôt Diamant I, pour toutes ces parties de jeux de société.

Je n'oublie pas bien sûr mes autres collègues qui resteront dans mes souvenirs. Je tiens à remercier Anaëlle, que je connais depuis maintenant cinq ans, pour son rayonnement naturel et son sourire si communicatif. Fabien le vrai président, pour son mental imperturbable et sa gentillesse indéfectible. Marcel pour son marxo-gauchisme, Satya pour son crypto-anarchisme, Tom pour son néocommunisme, que je n'ai pas su droitiser. J'embrasse fort Michelle, Noah et Luca. Olivier pour ses blagues d'une qualité aussi grande que son âge. Diana, pour son sourire et son style quotidiennement travaillé et renouvelé. Sans oublier Ons, Lydia, Youcef, Nathanaël, Justin, Marek, Anne, Linda, Hiba, Feu Paul-Henri, Mehdi, Mehdi, Amine, Lamine, Saeed, Lyes, Raja, Louis, Raouia, Pedro, Sami, Georges, Renaud, Hossein, Oussama, Manel, Paul, Axel, Mayassa, Sonia avec qui j'ai visité la NASA, et Florian mon grand frère de thèse.

Je tiens à remercier Alain, Mathieu et Louise du groupe *Louise XIV* (ex- *Apostrophe*) avec qui j'ai pu partager tous ces moments amicaux et musicaux. Merci également à Antoine, avec qui j'ai toujours eu plaisir à discuter politique, à jouer de la musique et à écouter la sienne.

Je remercie également Thomas, maintenant docteur et que j'espère bientôt papa. Merci d'avoir été là, d'avoir toujours été de bon conseil, et d'avoir entretenu avec moi des rapports aussi authentiques. Je n'oublie pas Helena, za to, že jste tak dobrým velvyslancem pro Českou republiku : tak milí lidé ! [5]

Je tiens aussi à remercier Pierre, que je connais depuis maintenant douze ans, avec qui je partage une relation fusionnelle. Merci d'avoir supporté toutes mes dilemmes psychiques. Je n'oublie pas mes amis proches que j'ai toujours eu plaisir à côtoyer depuis des années : Nathan, Tifenn, Cécile, Sabrina, Anastasia, Max, Louis, Romain et Oana.

J'ai également une pensée pour tous les kheys qui ont toujours eu deux tours d'avance, merci d'avoir partagé mon quotidien.

Je souhaite finalement remercier ma famille. Mon père et ma mère, ainsi que Bernadette, qui m'ont toujours soutenu. Merci d'avoir été présents en toute circonstance.

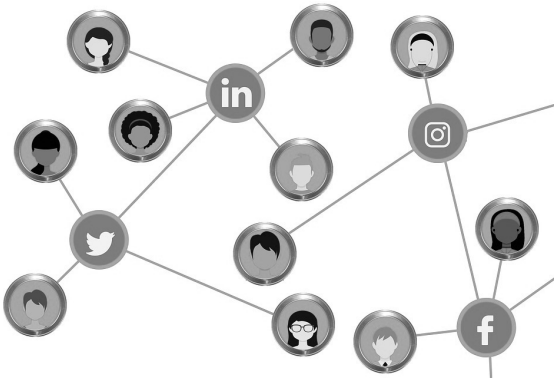
Je remercie enfin particulièrement mon frère Arnaud, dont ma relation avec lui fait miroir avec cette thèse : une histoire longue et compliquée qui finit bien à la fin. Cette thèse t'est donc dédiée.

4. *pour ta bonne humeur et surtout ton humour.*
5. *pour avoir été une si bonne ambassadrice de la République Tchèque : un peuple si gentil !*

Contents

Résumé de la thèse en français

Avec le récent développement de nombreux réseaux sociaux (Facebook, LinkedIn, Twitter...), l'étude de ces réseaux a particulièrement suscité l'intérêt de la communauté scientifique pour des raisons sociales ou économiques [57, 71, 129]. En particulier, un des enjeux importants dans l'étude des réseaux sociaux est la recherche de communautés, domaine qui a beaucoup été étudié ces dernières années.



La première question qui vient naturellement lorsque l'on parle de détection de communautés est : qu'est-ce qu'une communauté ? La première intuition qui semblerait la plus naturelle est de considérer qu'une communauté est un groupe de personnes qui se connaissent toutes. Dans un graphe, cela correspond à un ensemble de sommets joints deux à deux par une arête. Une telle structure est appelée clique. Trouver une clique d'une certaine taille k est un problème combinatoire très connu et difficile à résoudre. Cependant, un ensemble de sommets dont tous les couples seraient joints par une arête, sauf un couple, ne serait pas considéré comme une communauté. Cette observation pousse naturellement à chercher d'autres définitions qui seraient pertinentes pour décrire une certaine cohésion, sans être aussi restrictive qu'une clique. A partir de là, le choix des contraintes qui décrivent la cohésion d'un ensemble de sommets est très arbitraire, et dépend du type de cohésion

que l'on cherche. En ce sens, il n'existe pas de définition absolue pour une "communauté", mais la pertinence de diverses définitions peut être discutée.

Dans cette thèse, nous étudions quatre définitions pour décrire une communauté: les structures en communautés, les communautés, les clubs et les communautés potentielles. Cette dernière définition essaiera d'étendre l'intérêt du domaine à la recherche de groupes de personnes qui ne se connaissent pas, mais dont la cohésion est évaluée par les connaissances communes.

Communautés et structures en communautés (Chapitre 4)

Dans le Chapitre 4, nous étudions les structures en communautés. Une *structure en k communautés* dans un graphe connexe $G = (V, E)$ est une partition $\Pi = \{C_1, \ldots, C_k\}$ de V, $k \geq 2$, telle que $\forall i \in \{1, \ldots, k\}, |C_i| \geq 2$, et $\forall v \in C_i, \forall C_j \in \Pi$, $j \neq i$, on a :

$$\frac{|d_{C_i}(v)|}{|C_i| - 1} \geq \frac{|d_{C_j}(v)|}{|C_j|}$$

Une *structure en k communautés au sens faible* dans un graphe connexe $G = (V, E)$ est une partition $\Pi = \{C_1, \ldots, C_k\}$ de V, $k \geq 2$, telle que $\forall i \in \{1, \ldots, k\}, |C_i| \geq 2$, et $\forall v \in C_i, \forall C_j \in \Pi$, $j \neq i$, on a :

$$\frac{|d_{C_i}[v]|}{|C_i|} \geq \frac{|d_{C_j}(v)|}{|C_j|}$$

Nous appelons respectivement ces deux inégalités la condition de proportion et la condition de proportion au sens faible.

Informellement, une structure en k communautés est une partition des sommets en k parties telle que tout sommet a une plus grande proportion de voisins dans sa propre partie que dans toute autre partie. Cette définition nous a semblé pertinente, car elle prend en compte le nombre de connaissances qu'un individu peut avoir dans un groupe, mais aussi la taille du groupe. En effet, un individu n'a pas le même sentiment d'appartenance lorsqu'il connaît 2 personnes parmi 10 plutôt que 2 personnes parmi 100.

Nous nous intéressons alors au problème naturel suivant :

2-Community
Données : Un graphe $G = (V, E)$.
Question : Existe-t-il une structure en deux communautés dans G ?

Cette définition n'ayant été introduite que récemment, il existe peu de résultats dans la littérature à propos des structures en communautés. Olsen [133] a montré qu'étant donné un graphe, une structure en communautés peut toujours être trouvée (sans restriction sur le nombre de communautés) en temps polynomial si le graphe n'est pas une étoile. Plus récemment, Estivill-Castro *et al.* [59] ont montré que le problème de décider s'il existe une structure en k communautés, telle que chaque communauté doit être connexe et qu'elles soient toutes de tailles égales, est NP-complet dans les graphes généraux, mais résoluble

en temps polynomial dans les arbres. Dans [133], Olsen a également montré qu'il est NP-complet de décider s'il existe une structure en communautés dont l'une d'elles contient un ensemble de sommets défini en entrée.

Nous avons étudié le problème de trouver une structure en deux communautés dans un graphe, qui n'est pas une étoile, dans les classes de graphes suivantes :

- Pour les arbres, une structure en deux communautés existe et peut être générée en temps linéaire par un algorithme plus simple que celui donné dans [59]. Il existe des arbres avec une structure en deux communautés de même taille, mais sans structure en deux communautés connexes de même taille.

- Pour les graphes de degré maximum 3, il existe une structure en deux communautés connexes et celle-ci peut être générée en temps polynomial. De plus, il existe une structure en deux communautés de même taille au sens faible et celle-ci peut être générée en temps polynomial. Il existe des graphes sans structure de deux communautés de même taille. Il existe aussi des graphes ayant une structure en deux communautés de même taille, mais sans structure en deux communautés connexes de même taille au sens faible.

- Pour les graphes de degré minimum $|V| - 3$, les graphes complémentaires de graphes bipartis, les graphes de degré minimum $\lceil \frac{(c-1)\cdot|V|}{c} \rceil$ où c est la taille d'une clique maximale au sens de l'inclusion dans le graphe, une structure en deux communautés connexes existe et peut être générée en temps polynomial.

- Pour les graphes de largeur arborescente bornée, il existe des graphes sans structure en deux communautés de même taille, et décider si une telle structure existe (et si elle existe, la générer) peut être fait en temps polynomial.

Estivill-Castro *et al.* [58] ont prouvé que le problème de trouver une structure en deux communautés de même taille est NP-difficile dans les graphes généraux. Nous démontrons le même résultat en simplifiant drastiquement la preuve et en démontrant que ce problème est en fait équivalent à celui de trouver une partition en deux parties de même taille, telle que tout sommet a au moins autant de voisins dans sa propre partie que dans l'autre partie. Ce dernier problème a déjà été montré NP-complet [20].

En ce qui concerne le problème de trouver une structure en deux communautés de même taille au sens faible, la situation est légèrement différente. Tout graphe de degré maximum 3 a une structure en deux communautés de même taille au sens faible, tandis qu'il existe des graphes sans structure en deux communautés de même taille dans la même classe de graphes. En terme de complexité, générer une structure en deux communautés de même taille au sens faible peut être fait en temps polynomial dans les graphes de degré maximum 3, tandis qu'il est NP-difficile d'établir s'il existe une structure en deux communautés de même taille dans les graphes généraux, tout comme sa version au sens faible. Les résultats sont similaires lorsque la connexité des deux communautés est requise.

Par ailleurs, nous avons trouvé des graphes non triviaux dans lesquels il n'existe pas de structure en deux communautés. Cette observation a motivé l'étude d'un nouveau problème lié à 2-Community, que nous étudions dans le Chapitre 5.

Max Community (Chapitre 5)

Etant donné qu'il existe des graphes dans lesquels il n'y a pas de structure en deux communautés, il est intéressant de considérer une relaxation de la définition d'une structure en deux communautés, en acceptant que l'une des deux parties n'ait pas besoin de respecter la condition de proportion. L'autre partie est alors considérée comme une communauté. Ainsi, nous étudions le problème de trouver une communauté de taille maximum:

Max Community
Donnée : Un graphe $G = (V, E)$.
Résultat : Un ensemble C de taille maximale tel que pour tout sommet $v \in C$, $\frac{d_C(v)}{\lvert C\rvert -1} \geq \frac{d_{V\setminus C}(v)}{\lvert V\setminus C\rvert}$.

Nous montrons dans le Chapitre 5 que ce problème est NP-complet et APX-difficile, même dans les graphes split. De plus, étant donné un graphe $G = (V, E)$, il est toujours possible de trouver en temps polynomial une communauté de taille au moins $\frac{|V|}{2}$, ce qui prouve que le problème est 2-approximable en temps polynomial. Nous donnons également une borne supérieure pour la taille d'une communauté, ce qui permet de légèrement l'améliorer. Ensuite, nous montrons que Max Community est résoluble en temps linéaire dans les graphes cubiques Hamiltoniens si un cycle Hamiltonien est donné en entrée. Enfin, nous montrons qu'il est NP-complet de déterminer si, étant donnés un graphe et un ensemble S de sommets du graphe, il existe une autre communauté incluant S.

Clubs (Chapitre 6)

Comme nous l'avons indiqué précédemment, une communauté peut être vue au premier abord comme une clique dans un graphe. Une relaxation naturelle est de chercher un sous graphe de diamètre restreint: au lieu d'établir que la distance entre toute paire de sommets soit 1, nous acceptons qu'elle soit d'au plus k pour un certain entier k. A cet effet, nous étudions plusieurs problèmes liés à la recherche de sous graphes de diamètre 2 et 3, appelés respectivement 2-clubs et 3-clubs.

Dans le Chapitre 6, nous étudions ainsi les problèmes suivants :

k-Partition into s-clubs
Données : Un graphe $G = (V, E)$, deux entiers k, s.
Question : Existe-t'il une partition $\{P_1, P_2, \ldots, P_k\}$ de V telle que P_i est un s-club, pour tout $i \in \{1, \ldots, k\}$?

s-Club Edges Adding
Données : Un graphe $G = (V, E)$, deux entiers s, t.
Question : Existe-t'il un ensemble d'arètes E' de taille au plus t tel que V est un s-club dans le graphe $G' = (V, E \cup E')$?

Spanning s-club
Données : Un graphe $G = (V, E)$, un entier k.
Question : Existe-t'il un sous-ensemble d'arètes $E' \subset E$ de taille au plus k tel que le graphe $G' = (V, E')$ a pour diamètre s ?

Nous démontrons que partitionner un graphe en deux 2-clubs est NP-difficile même dans les graphes split. De plus, nous montrons que 2-Club Edges Adding est W[2]-difficile même dans les graphes split. Nous discutons sur l'éventuelle NP-difficulté du problème de Spanning 3-club dans les graphes split. Cette NP-difficulté impliquerait la NP-difficulté de Spanning $(2s+1)$-club dans les graphes généraux pour tout entier $s \geq 1$. De plus, si Spanning 2-club est NP-difficile, on peut montrer que Spanning s-club est NP-difficile pour tout entier $s \geq 2$.

Independent 2-Cliques (Chapitre 7)

Avec le développement récent des sites de rencontre dans lesquels les utilisateurs s'attendent à ne pas connaître les personnes qu'ils vont rencontrer (Meet-up, Couchsurfing...), nous définissons un problème de détection de communautés, dans le but de rapprocher des gens qui ne se connaissent pas a priori, mais qui sont liés par leurs connaissances communes.

Nous définissons un independent 2-clique comme un ensemble de sommets C tels que pour toute paire de sommets dans C, ces deux sommets sont non adjacents mais ont un voisin commun qui n'est pas dans C. Le problème étudié (Max Independent 2-Clique) est celui de trouver, étant donné un graphe, un independent 2-clique de taille maximale.

Max Independent 2-Clique
Donnée : Un graphe $G = (V, E)$.
Résultat : Un independent 2-clique de taille maximale.

Etant proche du problème de recherche d'un ensemble de sommets deux à deux non adjacents de taille maximum (Max Independent Set), nous comparons la difficulté des deux problèmes selon la classe de graphes dans laquelle ils sont étudiés. La Figure 1 résume ces résultats. Une flèche d'une classe à une autre indique que cette classe contient l'autre. Max Independent 2-Clique est NP-difficile dans la partie hachurée (en haut), résoluble en temps polynomial (en bas). Max Independent Set est NP-difficile dans la partie en pointillé (à gauche) et résoluble en temps polynomial (à droite).

Du point de vue de l'approximation, nous montrons que Max Independent 2-Clique n'est pas $n^{1-\epsilon}$- approximable en temps polynomial dans les graphes partout denses et dans les graphes split. De plus, ce problème n'est pas $n^{1/2-\epsilon}$- approximable en temps polynomial dans les graphes bipartis et est APX-difficile dans les line graphes.

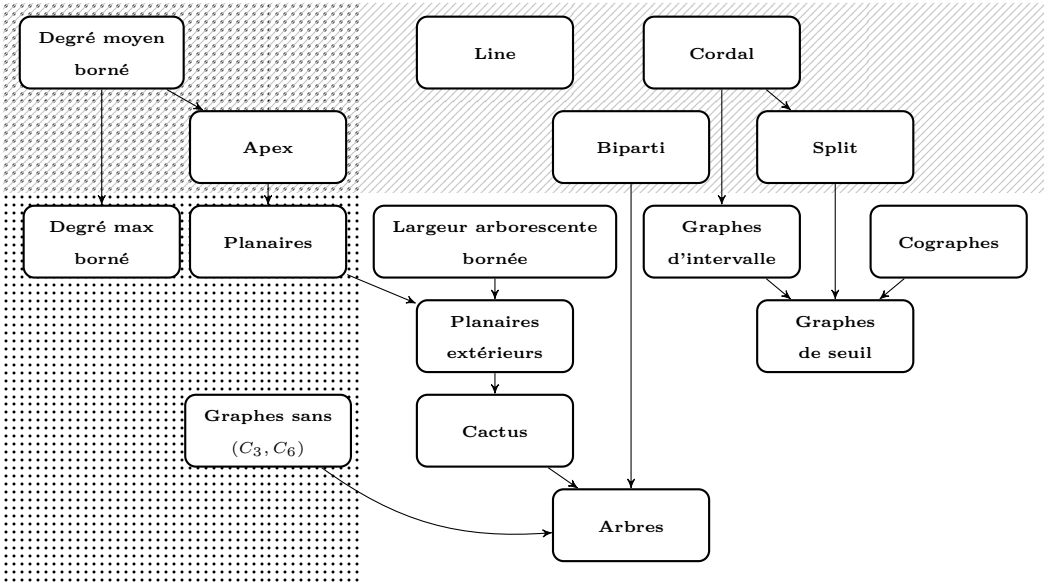
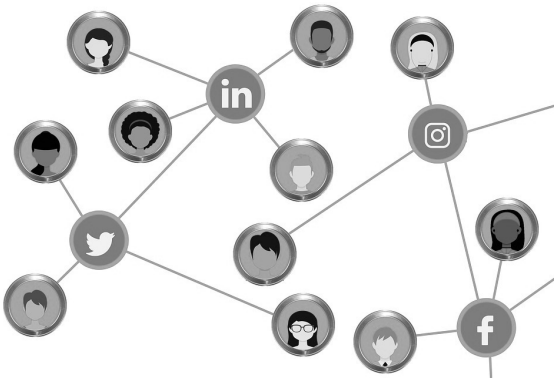


Figure 1: Comparaison des complexités de Max Independent Set et Max Independent 2-Clique

1 Introduction

The last decades have been marked by a significant explosion of communications with the developpement of the Internet. This turning point have led to the emergence of plenty of social networks like Facebook, Twitter, Snapchat, LinkdIn, Couchsurfing in which people can connect with each other and share information and messages.



In this way, studies about social networks have become a major stake in many different fields like economy, social science or marketing. In particular, one of the main questions in a social network analysis is to determine if there are communities, how many, and how the social network is organized by those communities.

Detecting communities has a lot of applications. Even before the existence of internet, people investigated problems around communities. In [159], Zachary presented the necessity of detecting communities in a karate club with 34 members, containing 78 pairwise links between members who interacted outside the club. Since a conflict arose between the administrator and the instructor, the club had to be split into two (see Figure 1.1). Many other various applications were investigated like identifying bitcoin users [34] or placing component of an electronic circuit into printed circuit cards [107]. Nowadays with massive social networks like Facebook, community detection became an even more significant stake with other applications. For instance, Facebook tries to suggest new relationships to its members. More generally, massive online stores like Amazon, or event organizers try

to find communities among their users in order to recommend them specific products or information.

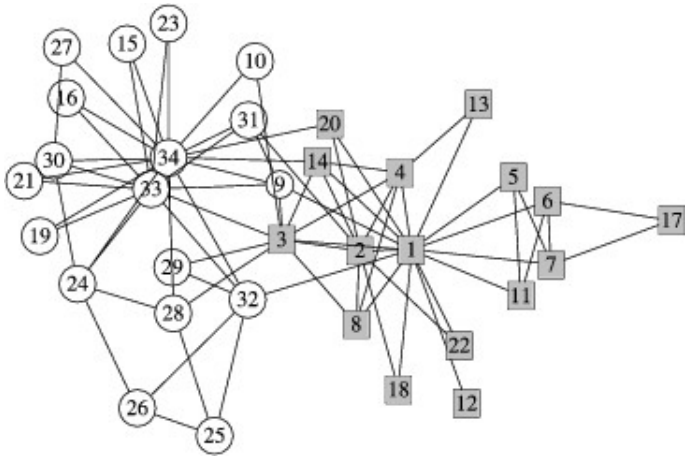
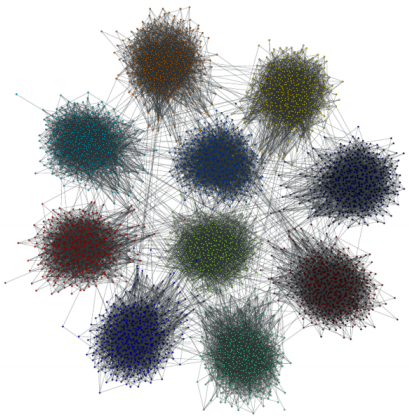


Figure 1.1: Zachary's karate club given in [71].

The first question you may ask is: "What is a community ?". If the term "community" refers to the intuition of some kind of cohesion in a social network, it is impossible to give a proper definition in absolute. Then, studying community detection is always made under a certain definition that is justified by specific criteria.

The standard structure of social networks is a group of individuals linked by relationships. A natural abstract model for those social network is a graph. A graph is a mathematical object constitued by a set of items called *vertices* and a set of links that join two vertices called *edges*.

In order to start with an intuitive way to apprehend a community, one simple definition that can be considered is a group of people in which everybody knows each other. In a graph, it corresponds to a subset of vertices such that any two vertices are linked by an edge. Such subset is called a *clique*. Looking for such community with the largest size could be a natural problem you may look at when you consider a social network.



Given a graph, finding a clique of maximum size should not be a mathematical problem in theory since it can be sorted by a simple exhaustive research. Indeed, it is sufficient to just consider all possible subsets of vertices, check if all edges occurs in this subset, and at the end, return the clique of maximum size you found. Suppose now that you have a computer able to check if a subset is a clique or not within 0.25 nanoseconds which is as fast as a processor of 4 GHz doing one operation. If you want to study a social network with only 70 members and try to find a clique of maximum size, you would have to check 2^{70} subsets, which would require more than 90 centuries to process... This leads to a natural question: is it possible to design an algorithm running in a reasonable time to solve this problem?

Most of computer scientists believe that finding a clique of maximum size is too *hard* to be solved efficiently (in the sense that it requires too much memory and time running to be solved in a reasonable time). More precisely, they proved that, under the conjecture $\mathsf{P}\neq\mathsf{NP}$, there is no algorithm running in polynomial time solving this problem.

The goal of this thesis is to study several definitions of a community that are relevant for different reasons that we discuss. For each definition, we study different problems around it and discuss their hardness, sometimes regarding specific graph structures. Either we prove that the problem can be solved by an efficient algorithm, or we prove that the problem is NP-hard and we discuss other options to approximate the problem efficiently. Approximation algorithms run in polynomial time and guarantee to output a solution near to the optimal solution by some factor.

The thesis is organized as follows. Chapter 2 gives the basic background in order to understand the content. Chapter 3 gives a state-of-art of several definitions of communities studied in the literature. Those definitions are grouped into two categories depending on the studied problem. First, only one community is considered as a subset of vertices in a graph. On the other hand, a community is described as a part of a partition in which each part is considered as a community. This thesis study four definitions for communities in the Chapters 4, 5, 6, and 7. The Chapters 4 and 5 investigate a similar definition of a community, but under different aspects. In Chapter 4 we consider a partition of the graph into communities (called *community structure*) such that each member of the graph knows a greater proportion of people in its part than in any other part. In Chapter 5, we investigate the same definition in which we only consider one community: a *community* is defined as a subset of vertices such that any vertex of the subset is linked by an edge to a greater proportion of vertices in its part than out of its part. The studied problem is to find such subset of maximum size. In Chapter 6, we consider a community as a subset of vertices such that there is a chain of relationship of size at most s between any two members within this subset, for a certain integer s. We call such structure an *s-club*. We study several problems related to that definition: partitioning a graph into two s-clubs, adding a minimum number of links in order to set the whole graph as an s-club, and removing the maximum number of edges in order to maintain an s-club structure. In Chapter 7, we do not try to study communities but *potential communities*. In the context of meetup services, we want to form a group of people that do not know each other but are related by another criterion than direct relationship. In this way, we define an *independent*

2-clique as a subset of vertices such that nobody knows each other, but any two members have a common acquaintance. We study the problem of finding such structure of maximum size. In Chapter 8 we give some conclusions and perspectives for future work.

2 Preliminaries

Contents

In this chapter, we give the basic background on graph theory, complexity, approximation and parameterized complexity.

2.1 Graphs

2.1.1 Basics of graphs

A graph is an ordered pair of sets (V, E) such that $E \subseteq V \times V$.

In this thesis, it is always assumed that we consider a graph $G = (V, E)$. Items of V are called *vertices* and items of E are called *edges*. In order to simplify the notations, we always note xy an edge $\{x, y\} \in E$ for any given two vertices x, y of V. Most of the time, we consider that for any graph $G = (V, E)$ and any $x \in V$, $xx \notin E$. Considering an edge $xy \in E$, we call x and y the *endpoints* of the edge.

All graphs considered in this thesis are finite (in the sense that the number of vertices is finite) and simple (at most one edge exists between two vertices).

A *directed graph* is a graph in which each edge is ordered. In this case, we note (x, y) the directed edge from the vertex x to the vertex y.

The *complement* $\overline{G} = (V, \overline{E})$ of a graph $G = (V, E)$ is the graph in which $uv \in E$ if and only if $uv \notin \overline{E}$, for all vertex pairs $u, v \in V$.

For any vertex $v \in V$, we say that v is *adjacent* to another vertex v' if $vv' \in E$. We call *neighborhood* of v the set $N(v) := \{x \in V : vx \in E\}$. We call closed neighborhood the set $N[v] := N(v) \cup \{v\}$. The size of a neighborhood of a vertex v is called *degree* of v and is noted $d(v)$. For any subset $H \subset V$ of the graph G, we note $N_H(v)$ the set of the neighbors of v in H, and similarly $N_H[v] := N_H(v) \cup \{v\}$ and $d_H(v) := |N_H(v)|$.

A graph is said to be *of maximum (resp. minimum) degree* d if and only if any vertex of the graph has degree at most (resp. at least) d. When all vertices have the same degree d, we say that the graph is d-regular. We sometimes call *cubic* a 3-regular graph.

A graph is said to be *complete* if for any two vertices $x, y \in V$, $xy \in E$.

A *subgraph* of G is a graph $H = (V(H), E(H))$ such that $V(H) \subset V$ and $E(H) \subset E$. We say that H is the subgraph of G *induced* by $V(H)$ if $E(H)$ contains all edges from E with two endpoints in $V(H)$. A *proper subgraph* of a graph G is a subgraph of G which is not G. A subgraph H is said to be *maximal* (or *inclusion-wise maximal*) under a certain property if there is no subgraph H' respecting this property such that $H \subset H'$. A *spanning subgraph* of G is a subgraph $H = (V(H), E(H))$ of G such that $V(H) = V$, $E(H) \subset E$, and H is connected.

Given a graph $G = (V, E)$, a *path* is a finite sequence of vertices $(v_1, v_2, \cdots, v_p)$ of V such that for any $i \in \{1, \cdots, p-1\}$, $v_i v_{i+1} \in E$. The *length* of a path is the number of edges involved in the sequence.

The *distance* between two vertices $x, y \in V$ is the length of a shortest path between them.

The *diameter* of a graph is the maximum distance that can occur between two vertices of the graph.

The k^{th} *power* $G^k = (V_k, E_k)$ of a graph $G = (V, E)$ is a graph such that $V_k = V$, and for any two vertices $x, y \in V_k$, $xy \in E_k$ if and only if $d(x, y) \leq k$ in G. For $k = 2$, we name such graph as a *squared graph*.

A graph is *connected* when for any two vertices $x, y \in V$, there exist a path between x and y. A *connected component* of a graph G is an inclusion-wise maximal connected induced subgraph of G.

A graph is *k-connected* if at least k vertices must be removed from the graph to make it disconnected.

A *pendant vertex* of G is any vertex of degree 1.

Given a graph $G = (V, E)$, a *cycle* is a path $(v_1, v_2, \cdots, v_p)$ of V such that $v_1 = v_p$. Given a cycle, we sometimes call *chord* an edge from E between two non consecutive vertices in the cycle. The *length* of a cycle is the number of edges involved in the sequence. A *k-cycle* is a cycle of length k. We sometimes call *triangle* a cycle of length 3. An *odd cycle* is a cycle with an odd length. A *Hamiltonian cycle* is a cycle such that its sequence contains all vertices of V exactly once.

A *partition* of V is a set of subsets $\{V_1, V_2, \ldots, V_p\}$ of V for some integer p such that $\cup_{i=1}^{p} V_i$ and for any two $i, j \in \{1, 2, \ldots p\}$, $i \neq j$, $V_i \cap V_j = \emptyset$. A *k-partition* of V is a partition of V with k sets. 2-partitions are sometimes called *cut*. The *size* of a cut (sometimes called

cut size) is the number of edges with one endpoint in each part. Given two vertices x, y of G, an *x,y-cut* is a 2-partition in which x and y do not belong to the same part. A partition is *connected* if the subgraphs induced by each part of vertices is connected. A partition is *balanced* if the sizes of each part differ by at most 1.

Given a partition of V, for any vertex v, we call *in-neighbor* (resp. *out-neighbors*) of v any neighbor of v which is in the same part of v (resp. not in the same part of v).

A *k-coloration* of V is a function $f : V \to \{1, 2, \ldots, k\}$ such that for any two vertices $x, y \in V$, if $f(x) = f(y)$ then $xy \notin E$.

In the following, we define some graph structures that will appear several times in the thesis.

A *star* is a connected graph in which all vertices have degree 1 except one vertex that can have any degree.

A *clique* K is a subset $K \subset V$ such that for any two vertices $x, y \in K$, $xy \in E$. We also call *triangle* a clique with 3 vertices. A *biclique* is a bipartite graph (see Subsection 2.1.3 for a definition of a bipartite graph) in which for any two vertices x, y from each independent set, $xy \in E$.

An *independent set* I is a subset $I \subset V$ such that for any two vertices $x, y \in K$, $xy \notin E$.

A *vertex cover* V' is a subset $V' \subset V$ such that for for any edge $e \in E$, there exist a vertex $v \in V'$ such that e contains v. Notice that, given a graph $G = (V, E)$, $I \subset V$ is an independent set of G if and only if $V \setminus I$ is a vertex cover of G.

A *dominating set* is a subset D of V such that for any vertex $v \in V$, either $v \in D$ or there exists a vertex $v' \in D$ such that $vv' \in E$.

2.1.2 Notations

Throughout the thesis, we use several notations that we regroup here. Given a graph $G = (V, E)$, a subset $H \subset V$ and two vertices v, w of G:

- $G[H]$ is the subgraph induced by H *i.e.* the graph $G' = (H, E')$ with E' the set of edges between two vertices of H in G.
- $G - \{v\}$ is an abuse of notation of $G[V \setminus \{v\}]$, *i.e.* the subgraph induced by $V \setminus \{v\}$.
- $\overline{H}$ is the complement of H in G, *i.e.* $V \setminus H$.
- $d(v)$ is the degree of v.
- $d_H(v)$ is the number of neighbors of v that belong to H.
- $d_{in}(v)$ is the number of in-neighbors of v, when a partition is given.
- $d_{out}(v)$ is the number of out-neighbors of v, when a partition is given.
- $\Delta(G)$ is the maximum degree of a vertex in G.
- $d(v, w)$ is the distance between v and w in G.

- $d_H(v, w)$ is the distance between v and w in $G[H]$. This notation is used when $v, w \in H$.
- K_p is a clique with p vertices. $K_{p,m}$ is a biclique constructed with two independent sets respectively of size p and m.
- S_p is a star with p vertices.

2.1.3 Graph classes

A graph class $\mathcal{G}$ is the set of all graphs respecting a certain property. We define all graph classes we study throughout this thesis.

A graph is a *tree* if it is connected and does not contain any cycle.

A *cactus* is a graph in which each edge occurs in at most one cycle.

A *bipartite* graph is a graph in which the set of vertices can be partitioned into two parts such that the subgraphs induced by each part are independent sets. We denote by $K_{p,m}$ the bipartite graph such that the sizes of the two independent sets are p and m, and each of the p vertices in the first one is adjacent to each of the m vertices in the second one.

A graph is *planar* if it can be embedded in the plane (drawn with points for vertices and curves for edges) without crossing edges. An equivalent definition given in [112] is that a planar graph does not contain a subgraph that is an expansion (*i.e.* some edges xy could have been subdivided into two edges xz and zy with an added vertex z) of K_5 or $K_{3,3}$.

A graph is *outerplanar* if it has a crossing-free embedding in the plane such that all vertices are on the same face. A graph is *k-outerplanar* if for $k = 1$, G is outerplanar and for $k > 1$ the graph has a planar embedding such that if all vertices on the exterior face are deleted, the connected components of the remaining graph are all $(k-1)$-outerplanar.

A graph G is *apex* if it contains a vertex v such that $G \setminus \{v\}$ is planar.

Given a graph H, a *H-free graph* is a graph not containing H as an induced subgraph.

An *interval graph* is a graph for which there exists a family of intervals on the real line and a bijection between the vertices of the graph and the intervals of the family in such a way that two vertices are joined by an edge if and only if the intersection of the two corresponding intervals is non-empty.

A graph is a *threshold graph* if it can be constructed from the empty graph by a sequence of two operations: insertion of an isolated vertex, and insertion of a dominating vertex (i.e., a vertex adjacent to all the other vertices). The original definition is that the graph admits a vertex labeling with positive real numbers, such that two vertices are adjacent if and only if the sum of their labels is at least a given 'threshold' t.

A *cograph* is a graph that can be generated from the single-vertex graph by (repeated applications of) complementation and vertex-disjoint union. These are precisely the graphs containing no induced paths on four vertices.

A *split graph* is a graph whose vertex set can be partitioned into two subsets, one inducing an independent set and the other one inducing a clique.

The *line graph* of a graph G (noted $L(G)$) is the graph whose vertices represent the edges of G, and two vertices of $L(G)$ are adjacent if and only if the corresponding two edges of G share a vertex.

A graph on n vertices is *δ-dense* if it has at least $\frac{\delta n^2}{2}$ edges and is *everywhere-δ-dense* if the minimum degree is at least δn.

A family of graphs is *dense* if there is a constant $\delta > 0$ such that all members of this family are δ-dense.

A family of graphs is *everywhere-dense* if there is a constant $\delta > 0$ such that all members of this family are everywhere-δ-dense.

2.2 Computational complexity

2.2.1 Decision and optimization problems

Given a problem, an *instance* is the data involved in the problem.

Algorithms. An *algorithm* is a procedure that takes an instance as an input and gives an output. We distinguish two kinds of algorithms: *deterministic algorithms*, that have a unique execution and a unique output for each input, and *non-deterministic algorithms*, that can produce different executions for the same input. The *running time* of an algorithm is the number of operations that the algorithm uses in the worst case scenario to give an output. We say that the running time is *polynomial* if the running time can be expressed as a polynomial function on the size of the input. Usually, instead of giving the exact number of operations, we use the big O notation in order to express the general running time.

Decision problems and the class NP. A decision problem is the data of a set of instances and a question whose answer is *yes* or *no* that can be asked for any instance of the set. When we introduce a new decision problem, we will make use of the following standard way to define it:

Decision Problem Name
Input : An instance.
Question : A *yes/no* question that depends on the input.

NP is the set of all decision problems such that there exists a non-deterministic algorithm solving the problem in polynomial time. Given two decision problems D_1, D_2, we say that that D_1 polynomial-time reduces to D_2 if there exists an algorithm that, taking as input any instance x_1 of D_1, gives in polynomial time an instance x_2 of D_2 such that x_1 is a *yes*-instance if and only if x_2 is a *yes*-instance. A decision problem D is *NP-hard* if any problem of NP polynomial-time reduces to D. If D is an NP-hard problem that belongs to NP, we say that D is *NP-complete*. As an example, we give the following famous NP-hard problem from Garey and Johnson [77]:

SAT
Input : A set U of variables, a collection C of clauses over U.
Question : Is there a satisfying truth assignment for C?

Cook proved in [42] that SAT is NP-hard by proving that for any problem A in NP, there is a polynomial reduction from A to SAT. Most of the time, in order to prove the NP-hardness of a problem, we reduce our problem from an already proved NP-hard problem.

P is the set of all decision problems such that there exists a deterministic algorithm solving the problem in polynomial time. Most of computer scientists believe that $\mathsf{P} \neq \mathsf{NP}$, even though $\mathsf{P} \subseteq \mathsf{NP}$.

Optimization problems and the class NPO. An optimization problem is the data of:

- a set of instances.
- for each instance, a set of feasible solutions with a size that is bounded by a polynomial on the size of the instance.
- a goal (either maximization or minimization).
- a cost function that takes an instance and a feasible solution as parameters and output a number (that is computable in polynomial time).

Given an optimization problem, an *optimal solution* is a feasible solution that fulfils the goal of the problem, *i.e.* maximizes (or minimizes) the value of the cost function. Given an optimization problem and an instance I of this problem, we denote by $|I|$ the size of I, by $opt(I)$ the value of an optimal solution for I, and by $val(I, S)$ the value of a feasible solution S for I. When we introduce a new optimization problem, we will make use of the following standard way to define it:

OPTIMIZATION PROBLEM NAME
Input : An instance.
Output : An optimal solution.

The link between optimization problems and decision problems is the following. If O is a maximization (resp. minimization) problem then we can define its decision version by introducing a parameter k and the question : "Is there a solution of value greater (resp. smaller) than k ?". We say that an optimization problem is NP-hard if its decision version is NP-hard.

NPO is the set of all optimization problems such that the associated decision problem is in NP. An optimization problem is polynomial-time solvable if there exists an algorithm that computes, for every instance of the problem, an optimal solution with a running time that is polynomial in the size of the instance. PO is the set of all optimization problems that are polynomial-time solvable.

2.2.2 Approximation and approximation-preserving reductions

Approximation. When an optimization problem is NP-hard, it is possible to study the approximation of this problem. The goal is to gives algorithms that run in polynomial time and output a solution which value is closed to the optimum value with a certain ratio called the performance ratio, which gives a guarantee for the quality of the given solution.

The *performance ratio* (or *approximation factor*) of a solution S for an instance I is $r(I,S) := \max\left\{\frac{val(I,S)}{opt(I)}, \frac{opt(I)}{val(I,S)}\right\}$. The closer the performance ratio is to 1, the closer the value of the solution is to the optimum value. The *error* of S, denoted by $\epsilon(I,S)$, is defined as $\epsilon(I,S) := r(I,S) - 1$.

For a function f, we say that an algorithm is an $f(|I|)$-*approximation*, if for every instance I of the problem, it returns a solution S such that $r(I,S) \leq f(|I|)$ in polynomial time. In this way we can define several classes of optimization.

The class PTAS is the set of optimization problems that allow polynomial-time approximation algorithms such that $val(S,I) \leq (1+\epsilon) \cdot opt(I)$ for a minimization problem or $val(S,I) \geq (1-\epsilon) \cdot opt(I)$ for a maximization problem, for any $\epsilon > 0$ given in input. Such algorithm is called a *ptas* (for polynomial-time approximation scheme). Usually, the running time of such algorithm is exponential in $\frac{1}{\epsilon}$. The optimization problems from NPO such that a polynomial-time approximation algorithm can be design with a running time that is polynomial in the size of the instance and in $\frac{1}{\epsilon}$ form the class FPTAS (See [155]).

The class APX is the set of optimization problems in NPO that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant. In the same way, the class *log*-APX (resp. *poly*-APX) is the set of optimization problems in NPO that allow polynomial-time approximation algorithms with approximation ratio bounded by $c \cdot log(|I|)$ (resp. $c \cdot p(|I|)$ with p a polynomial), for some constant c with $|I|$ the size of the instance.

At the end, we can set the following inclusions:

$$\mathsf{PO} \subseteq \mathsf{FPTAS} \subseteq \mathsf{PTAS} \subseteq \mathsf{APX} \subseteq \textit{log}\text{-}\mathsf{APX} \subseteq \textit{poly}\text{-}\mathsf{APX} \subseteq \mathsf{NPO}$$

Hardness of approximation. For proofs concerning the non-existence of a ptas, we use an approximation-preserving reduction, called L-reduction, which was introduced by Papadimitriou and Yannakakis in [137]. Let A and B be two optimization problems. Then A is said to be L-*reducible* to B if there are two constants $a, b > 0$ such that:

- there exists a function, computable in polynomial time, which transforms each instance I of A to an instance I' of B such that $opt_B(I') \leq a \cdot opt_A(I)$,
- there exists a function, computable in polynomial time, which transforms each solution S' of I' to a solution S of I such that $|val(I,S) - opt_A(I)| \leq b \cdot |val(I',S') - opt_B(I')|$.

An optimization problem is APX-hard if every problem of APX L-reduces to that problem. Then, if A is L-reducible to B and A is APX-hard then B is APX-hard.

The notion of an E-reduction (*error-preserving* reduction) was introduced by Khanna *et. al.* [108]. A problem A is called *E-reducible* to a problem B, if there exist polynomial-time computable functions f and g, and a constant β such that

- f maps an instance I of A to an instance I' of B such that $opt(I)$ and $opt(I')$ are related by a polynomial factor, i.e. there exists a polynomial p such that $opt(I') \leq p(|I|) \cdot opt(I)$,
- g maps any solution S' of I' to a solution S of I such that $\epsilon(I, S) \leq \beta \cdot \epsilon(I', S')$.

An important property of an E-reduction is that it can be applied uniformly to all levels of approximability; that is, if A is E-reducible to B and B belongs to $\mathscr{C}$ then A belongs to $\mathscr{C}$ as well, where $\mathscr{C}$ is a class of optimization problems with any kind of approximation guarantee (see [108]).

For more information about approximation algorithms, we recommand [99, 155].

2.2.3 Parameterized complexity

When we have to handle an NP-hard problem, it can be interesting to investigate parameterized complexity. Usually, we always express the running time of an algorithm depending on the size of the instance (in graphs, it is often the number of vertices or edges). The goal of this framework is to express the complexity of a decision problem by another parameter. In general, a natural parameter to study is the size of a solution.

A *parameterized problem* is a subset $Q \subset \Sigma \times \mathbb{N}$ where the first component is a decision problem and the second component is called the parameter of the problem. A lot of parameterized problems that are parameterized by the size k of the solution admit an algorithm with running time bounded by $c \cdot |I|^k$ for an instance I and a constant c. As an example, for the problem Clique parameterized by the size of the clique k, deciding if there is a clique of size at least k in a graph $G = (V, E)$ can be easily solved in $O(|V|^k)$ by checking any subset of size k in G. More generally, the set of parameterized problems that allow an algorithm that solves the decision problem with running time $O(f(k) \cdot |I|^k)$ for some computable function f is called XP.

However, it can be even more interesting to design algorithm with a running time that separates the size of the instance by the parameter. In that way, the class FPT contains every parameterized problem $Q \subset \Sigma \times \mathbb{N}$ for which the question "Does (x, k) belong to Q?" can be decided by an algorithm that runs in $f(k) \cdot |x|^{O(1)}$ time where $(x, k) \in \Sigma \times \mathbb{N}$ and f is a computable function.

Let $Q_1, Q_2 \subset \Sigma \times \mathbb{N}$ be two parameterized problems. We say that Q_1 *FPT-reduces* to Q_2 if there exists two computable functions f and g and an algorithm that takes as input an instance $(x_1, k_1) \in \Sigma \times \mathbb{N}$ and outputs a new instance $(x_2, k_2) \in \Sigma \times \mathbb{N}$ in $f(k_1) \cdot |x_1|^{O(1)}$ time such that:

- $(x_1, k_1) \in Q_1 \Leftrightarrow (x_2, k_2) \in Q_2$

- $k_2 \leq g(k_1)$

Downey and Fellows [53] introduced the W-hierarchy as different classes of complexity for parameterized problems. Before defining it, we need first to define preliminary concepts. A *boolean circuit* $C = (V, A)$ is a directed acyclic graph whose vertices V are called gates. The gates of in-degree 0 are called inputs. There is exactly one gate of out-degree 0 called output. Every gate that is neither an input nor an output is labeled by an element of $\{or,\ and,\ not\}$. A gate with label *not* has in-degree exactly one. A gate with in-degree bounded by a constant is said to be small, and otherwise it is called large. The weft of a boolean circuit is the maximum number of large gates on a path from an input to the output. The depth is the maximum number of all gates on a path from an input to the output. A truth assignment for a boolean circuit C is a function that associates the value true or false to each input gates. Given a truth assignment for C, the value of the output can be determined by computing the value of each gate according to their label and the values of the previous vertices. A truth assignment satisfies C if the value of the output gate is true. The weight of a truth assignment is the number of input gates set to true.

A parameterized problem (Q, k) belongs to $W[t]$, for a fixed $t > 0$, if (Q, k) FPT-reduces to Weft-t Circuit Satisfiability parameterized by k, where the latter problem is defined as follows:

Weft-t Circuit Satisfiability
Input : A boolean circuit C with constant depth and weft at most t, and an integer k.
Question : Is there a truth assignment of weight k that satisfies C?

A way to prove that a parameterized problem belongs to $W[t]$ is to construct an FPT-reduction from this problem to a problem known to be in $W[t]$. As an example we give an FPT-reduction from the following problem established in [53]:

Independent Set
Input : A graph $G = (V, E)$, an integer k.
Question : Is there a subset of vertices $S \subset V$ such that for any two vertices s_1, s_2 of S, $s_1 s_2 \notin E$?

Theorem 2.1 ([53])**.** Independent Set *belongs to* $W[1]$*.*

Proof. We construct an FPT-reduction from Independent Set to Weft-1 Circuit Satisfiability. Let $G = (V, E)$ be a graph as an instance of Independent Set. We construct a boolean circuit $C = (V, A)$ as an instance of Weft-1 Circuit Satisfiability as follows (see Figure 2.1 for an illustration). Introduce $|V|$ gates $a_i, i \in \{1, \cdots, |V|\}$ of in-degree 0, each gate corresponding to a vertex in G. For each gate a_i, introduce a gate b_i labeled "*not*" and add an edge (a_i, b_i) in A. For any two vertices v_i, v_j in V that are not linked by an edge, create a gate c_{ij} labeled "*or*" and add the edges (b_i, c_{ij}) and (b_j, c_{ij}).

Finally, add a gate d labeled "*and*" and add the edges (c_{ij}, d) for all c_{ij}. Now notice that since d is the only large gate, the boolean circuit has weft 1 and depth 4. Moreover, notice that there is an independent set of size at least k in G if and only if there is a truth assignment of weight k that satisfies C.

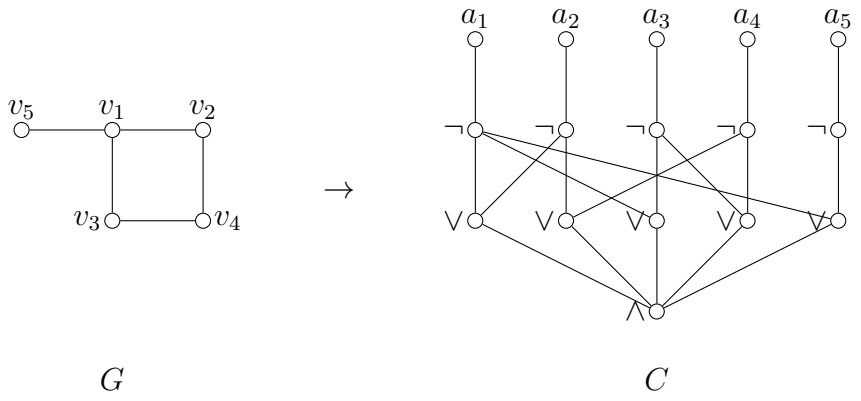


Figure 2.1: The construction of the boolean circuit C from G.

□

A parameterized problem is $W[t]$-*hard* if every problem of $W[t]$ FPT-reduces to it. It is $W[t]$-*complete* if it is $W[t]$-hard and belongs to $W[t]$.

At the end, we can set the following inclusions:

$$\mathsf{P} \subseteq \mathsf{FPT} \subseteq \mathsf{W[1]} \subseteq \mathsf{W[2]} \cdots \subseteq \mathsf{W[t]} \cdots \subseteq \mathsf{XP}$$

For more information about parameterized complexity, we recommand [54, 132].

3

Community detection in graphs : an overview

Contents

3.1 Community detection in social networks

The recent development of social networks such as Facebook or Linkedin and online meet-up services have motivated the investigation of community detection in such networks.

A standard abstract model for those networks are graphs in which a community should intuitively describe some cohesion, that often corresponds to some density in a subgraph (see Figure 3.1).

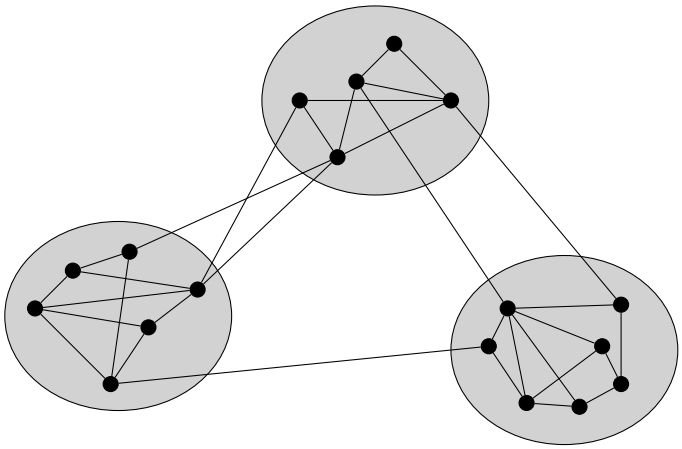


Figure 3.1: A social network partitioned into intuitive communities

It is natural to see that the problem of finding a community in a social network is closely related to the chosen definition of a community. Even if the idea of a community is intuitive, the number of ways to define it is huge. The goal of community detection is to determine, within a given definition of a community, how to find such structure (if it exists) respecting a certain goal (constraints on the size of the community, on the number of communities, on their quality...).

Since no definition can be considered in absolute, the definition is usually chosen depending on which aspect of a community is wanted to be captured.

In the following we give an overview of the definitions of a community that have been studied in the literature. Further information about community detection and more definitions can be found in [71, 72, 114].

3.2 Finding a single community

The first intuitive way to define a community is to look for a group of people in which all members of this group know each other. In graphs, it corresponds to a clique.

A classical problem related to finding a community as a subgraph is to look for a subgraph of inclusion-wise maximal size or maximum size. It has been proved in [128] that the number of maximal cliques in a graph with n vertices in bounded by $3^{\frac{n}{3}}$. It is possible to list all maximal cliques in a graph in polynomial time on the number of vertices, edges and the number of maximal cliques [154]. On the other hand, the problem of finding a clique of maximum size is a well known NP-hard problem [104].

However, considering communities as cliques is too restrictive: a subgraph with all possible internal edges except one would not be considered as a community under this assumption, even if it probably should be in real world social networks. For this reason, other definitions of a community have been studied in order to capture aspects of a community in social networks regarding different features that define a cohesion.

3.2.1 Restrictions on the distance between members

A first way to relax the restrictive condition of a clique is to consider *s-cliques*. Given a graph, an s-clique is a subset of vertices such that the distance between any pair of vertices is not larger than s. It is easy to see that an 1-clique is a clique.

This definition, more flexible than cliques, still has some limitations, deriving from the fact that the distance between two vertices in an s-clique may be ensured by vertices outside of it. In this way, there may be two disturbing consequences. First, the diameter of an s-clique may exceed s, even if the distance between any two vertices of the s-clique is at most s. Second, the subgraph induced by an s-clique may be disconnected, which is not consistent with the notion of cohesion a community should ensure. These problems for s-cliques have been studied in [4].

In order to take those drawbacks into account, Mokken suggested to introduce the concept of *s-club* in [126]. Given a graph, an s-club is a subset of vertices such that the subgraph induced by it has diameter at most s. Notice that any s-club is then included in an s-clique. See Figure 3.2 for an illustration taken from [4, 126].

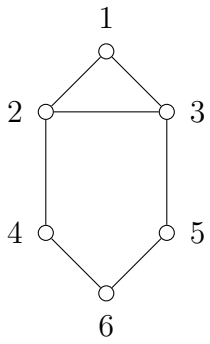


Figure 3.2: Graph illustrating s-cliques, s-clubs for $s = 2$. $\{1, 2, 3, 4, 5\}$, $\{2, 3, 4, 5, 6\}$ are 2-cliques meanwhile $\{1, 2, 3, 4\}$, $\{1, 2, 3, 5\}$, $\{2, 3, 4, 5, 6\}$ are 2-clubs.

Hence 1-clubs are exactly cliques, and every s-club is also an $(s+1)$-club by definition.

Notice the non-hereditary nature of s-clubs, which makes their behavior different from that of cliques for $s \geq 2$: although every subset of a clique is a clique, the same is not true for an s-club. In fact, given a graph $G = (V, E)$ and an s-club S_1 in G, deciding if there exists another s-club S_2 such that $S_1 \subset S_2$, $S_1 \neq S_2$ is NP-hard for every fixed $s \geq 2$ [134].

A natural problem with s-clubs is, given a graph G and an integer s, to find an s-club of maximum cardinality. The problem is trivial if G has diameter at most s, but is NP-hard for every fixed s, even on graphs of diameter $s + 1$ [12].

In this thesis, we investigate some aspects of s-clubs and prove several properties on problems related to them in Chapter 6.

A survey about various clique relaxation definitions for community detection can be found in [110].

3.2.2 Restrictions on the neighborhood

Instead of relating members by their distance, another way to ensure cohesion within a group of people is to directly deal with the amount of relationships for each members in their group. For instance, we can ask that every vertex has at least a certain amount of neighbors in their group. Seidman introduced the notion of a *k-core* in [150]. Given a graph G, a subgraph C of G is a k-core if any vertex of C has at least k neighbors in C. Notice that all 1-cores of a graph are all the connected components of it, and trees have no 2-cores. However, since k does not depend on the size of a k-core, the restriction on the number of neighbors inside C for vertices of C does not always ensure high cohesion in the graph, in particular when k is much smaller than the size of C.

Similarly to a k-core, Matsuda *et. al.* [124] introduced the notion of *p-quasi complete subgraph*, which is a subgraph such that the degree of each vertex is larger than $p(k-1)$, where p is a real number in $[0,1]$ and k the size of the subgraph. They proved that determining whether a graph includes a $\frac{1}{2}$-quasi complete subgraph of size at least k is NP-complete.

Instead of imposing a certain number of neighbors for each vertex, we can impose a certain amount of non neighbors. Imposing that any vertex of a community should be connected to any other vertex has already been studied under the name of clique, and relaxing this strong condition can be still relevant to find communities. In this way, Seidman and Foster introduced a clique-like structure in [151] under the notion of *k-plex*. Given a graph $G = (V, E)$, a subgraph C is defined as a k-plex if any vertex has at least $|C| - k$ neighbors inside C. Notice that cliques are k-plexes for any integer k, $1 \leq k \leq |C| - 1$.

Similarly, it is possible to set constraints on both (*i.e.* the number of neighbors inside and outside of the community). In this way, an (α, β)-*community* is defined in [125] as a subset of vertices C in which each vertex in C has at least $\beta \cdot |C|$ neighbors in C (including itself) and each vertex outside of C is linked to at most $\alpha.|C|$ vertices of C, with $\alpha < \beta \leq 1$. Given a graph $G = (V, E)$, the problem of finding a $(1 - \frac{1}{|V|}, 1)$-community is equivalent to finding a maximal clique in a graph, which is a well known problem we already discussed about. The $((1-\epsilon)\beta, \beta)$-communities, for small ϵ, have been studied under the name of quasicliques in [2]. In [95], they considered a similar definition by considering an (α, β)-community as a subset of vertices C in which each vertex in C has at least β neighbors in C and each vertex outside of C has at most α neighbors in C, $0 \leq \alpha < \beta$. The problem of finding an (α, β)-community of maximum size in a graph has been proved NP-hard in [156].

Instead of just studying the amount of neighbors inside or outside a community, it is also relevant to compare them and ask that the number of neighbors inside must be larger than the number of neighbors outside of the community in order to ensure the cohesion of the group. In this way, given a graph, an *alliance* is defined as a set of vertices C such that each vertex of C has at least as many neighbors in C as out of C. This definition has been motivated by the searching links in web graphs and introduced by Flake et al. [68], under

the name of *web community*. The term "alliance" has been introduced by Kristiansen *et. al.* [111] under the same definition.

It is easy to see that, given a graph G and two vertices x, y of G, it is possible to efficiently compute an alliance containing x but not y. Indeed, since of the well know theorem of "max flow/min cut" [70], we know that the value of the maximum flow through this network equals the minimum value of an (x, y)-cut. Then, considering a minimum (x, y)-cut in G which can be found in polynomial time [70], notice that any vertex of the set containing x given by the cut must have more neighbors in its own part than in the other part, otherwise the cut would not be of minimum value (see Figure 3.3 as an illustration).

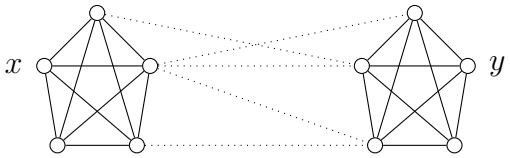


Figure 3.3: The minimum (x, y)-cut gives an alliance containing x

In [100], Jamieson *et. al.* showed that the problem of finding an alliance of minimum size in a graph is NP-hard even when restricted to split, chordal or bipartite graphs. In addition, finding an alliance of minimum size such that the alliance is also a dominating set is known to be NP-hard from [32].

Radicchi *et. al.* introduced a similar definition in [143] where a community in the strong sense is defined as a set of vertices C such that each vertex of C has strictly more neighbors in C than out of C. They also considered a community in the weak sense by considering subgraphs in which the condition for strong communities must be true only on average, *i.e.* the sum of the internal degree of each vertex of the community must be greater than the sum of the external degree of each vertex of the community.

In [153], Sigaretta *et. al.* studied the variant of *defensive k-alliance* in which any vertex must have at least k more neighbors in its part than out of its part. A survey and more information about alliances can be found in [65, 66].

If the cohesion of a set of vertices can be related to the number of in-neighbors of each vertex of a community, some studies tried to restrict the structure of the neighborhood of each vertex of a community. In this way, one can try to put vertices together according to the role that have by defining *role assignments*. This concept was in introduced in [61] under the name "role coloring". Given two graphs $G = (V_G, E_G)$ and $R = (V_R, E_R)$, an R-role assignment for G is a vertex mapping $r : V_G \to V_R$ such that the neighborhood relation is maintained, *i.e.* all roles of the image of a vertex appear on the vertex's neighborhood. Formally, for any vertex $u \in V_G$, $r(N_G(u)) = N_R(r(u))$. From the complexity point of view, several questions can be asked. Given two graph G and R, deciding if there is a role assignment r from G to R is NP-complete [145] even if R has 2 vertices. If R is not given in input, deciding if there is an assignment from G to R for some graph R is also NP-complete [67].

3.2.3 Maximizing the relationship density

Another way to express the cohesion within a subset of vertices is to deal with the density of relationships within a community, *i.e.* the density of edges occurring in a subgraph. A lot of densities, based either on high internal or low external connectivity have been studied in the literature. Given a graph $G = (V, E)$ and a set S of vertices in V considered as a community, we define a quality function $f(S)$ quantifying how community-like is the connectivity of vertices in S. We denote $E(S)$ the set of edges in E with two endpoints in S and $cut(S)$ the set of edges in E with one endpoint in S and one endpoint out of S.

Several definitions for densities can be found in the literature. The following non exhaustive list gives some quality functions that can be found:

- **Internal density**: $f(S) = \frac{|E(S)|}{|S|\cdot(|S|-1)/2}$ that is the internal edge density of S [122].
- **Internal-external density**: $f(S) = \frac{|E(S)|}{|S|\cdot(|S|-1)/2} - \frac{|cut(S)|}{|S|\cdot(|S|-1)/2}$ that is the difference between the internal edge density and the external edge density [122].
- **Edges inside**: $f(S) = |E(S)|$ which is the number of connexions in S [44, 63, 144]. A variant with weights on edges has been studied in [9].
- **Average degree**: $f(S) = \frac{2\cdot|E(S)|}{|S|}$ that is the average in-degree of vertices in S [83].
- **Fraction over median degree**: $f(S) = \frac{|\{v \in S : d_{in}(v) > d\}|}{|S|}$ is the fraction of vertices in S that have internal degree higher than the median value d of $d(u)$ in V [117].
- **Triangle Participation Ratio**: $f(S) = \frac{|\{v \in S : v \text{ belongs to a triangle in } S\}|}{|S|}$ that is the ratio of vertices belonging to a triangle in S [117].
- **Conductance**: $f(S) = \frac{|cut(S)|}{2\cdot|E(S)|+|cut(S)|}$ that is the fraction of edge contribution in S that have an endpoint out of S [116].

Most of those densities are NP-hard to maximize [146]. A comparison of heuristics for finding subsets of maximum density can be found in [117, 158].

3.2.4 Other features of communities

Other definitions in the literature tried to capture other features of communities. In [115], the notion of *LS set* was introduced to capture the idea that any subgraph of an LS set is less relevant as a community than itself. LS sets were first introduced by Luccio *et. al.* [121] under the name of "minimal groups", and Lawler [115] renamed them LS sets. An LS set is defined as follows. Given a graph $G = (V, E)$, a subset $H \subset V$ is an LS set if for any proper subgraph $H' \subset H$, the number of edges in E with one endpoint in H' and one endpoint in $V \setminus H'$ is always strictly greater than the number of edges with one endpoint in H and one endpoint in $V \setminus H$. In this way, any subset contained in an LS set has more

connections to the outside of it than the LS set itself. Hence, by being less connected to the rest of the graph, the LS set captures better the notion of cohesion.

Every singleton and the set of vertices V are trivial LS sets, but not interesting in our context. It can be noticed that any vertex of an LS set H has more neighbors in H than in $V \setminus H$, which seems an interesting property in the context of community detection, as we seen previously with alliances in Subsection 3.2.2.

An equivalent definition for an LS set given by Seidman in [149] is that an LS set is a subset of vertices H such that for any proper subset $H' \subset H$, the number of edges in E with one endpoint in H' and one endpoint in $H \setminus H'$ is strictly greater than the number of edges in E with one endpoint in H' and one endpoint in $V \setminus H$.

On the other hand, it has been showed in [121] that for any two LS sets H and H', either $H \cap H'$ is empty or one is included in the other. This property gives a good organization of the graph into communities (see Figure 3.4) that can be seen as a hierarchical clustering (see Section 3.3.7).

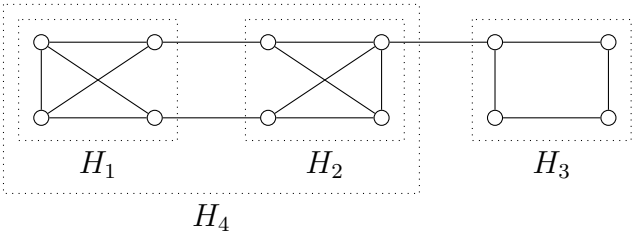


Figure 3.4: A graph in which all LS sets (H_1, H_2, H_3, H_4) are framed except trivial ones (singletons and the set of all vertices)

Denoting $\lambda(a, b)$ the minimum number of edges to remove in order to disconnect two vertices a, b in the graph, Borgatti *et. al.* showed in [28] that given a graph $G = (V, E)$ and an LS set $H \subset V$, for any $a, b, c \in H$ and $d \in V \setminus H$, $\lambda(a, b) > \lambda(c, d)$. This property highlights the cohesion of an LS set, and has been considered as a generalization of LS sets called *lambda sets* that has been defined as subsets of vertices satisfying this property [28].

Notice that the two definitions are not equivalent. In Figure 3.5, the subset of vertices H is not an LS set. Indeed, considering the proper subset $H' \subset H$ represented by the gray vertices, there are 4 edges with one endpoint in H' and one endpoint in $V \setminus H$ whereas there are only 2 edges with one endpoint in H' and one endpoint in H. On the other hand, H is a lambda set since the minimum number of edges to remove in order to disconnect any vertex of H from any vertex of $V \setminus H$ is 1, whereas H is 2-connected.

3.3 Finding a partition into communities

In this section, we focus on partitioning a graph into several parts that will be considered as communities. In the area of graph partitioning, results may find applications such

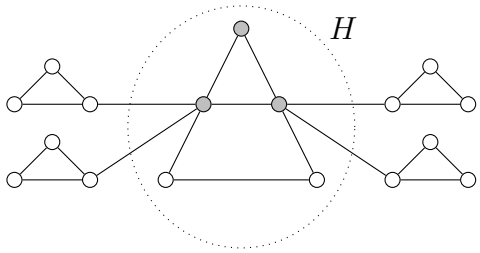


Figure 3.5: A graph in which the subset H is a lambda set but not an LS set.

as parallel-computing, VLSI-circuit design, route planning [51] and divide-and-conquer algorithms [152]. Several methods have been developed in order to find partitions that are relevant to describe a network partitioned into communities, like spectral clustering [27, 88, 123]. However, this section focuses on several definitions of a partition into communities in the literature and discusses their relevance in the context of community detection.

It is important to notice that Kleinberg showed in [109] that no clustering function (*i.e.* a function that take a graph $G = (V, E)$ and a distance function on $V \times V$ as an input and output a partition of V) can satisfy at the same time three natural properties for a partition into communities. This highlights the fact that there is no unique definition for a partition into communities, and such definition depends on the aspect of cohesion we want to capture.

3.3.1 Maximizing the number of links within the communities

A first natural way to partition a graph into communities is to maximize the number of links within communities. As discussed previously, the most cohesive communities we can found are cliques. Given a graph, deciding if there is a partition into k cliques in a graph is NP-complete from [104] and even NP-complete on planar cubic graphs [35]. However, finding a partition into k cliques can be done in polynomial time in some classes of graphs. In triangle-free graphs, such partition corresponds obviously to the union of a matching of maximum size and the remaining singletons. Finding a matching of maximum size can be done in polynomial time by solving a flow of maximum value in the adjacency graph. In perfect graphs[1], since the complement of a perfect graph is also a perfect graph [120], the minimum number of cliques in a partition into clique equals the size of an independent set of maximum size in the graph. Since finding an independent set of maximum size can be done in polynomial time in perfect graphs [87], it is the same result for finding a partition into a minimum number of cliques.

From the approximation point of view, finding a partition into a minimum number of cliques is not $n^{1-\epsilon}$-approximable in polynomial time unless P = NP [161]. The NP-

[1]Perfect graphs are graphs in which any subgraph satisfies that the minimum number of colors in a coloring equals the maximum size of a clique.

hardness of this problem and its approximation has also been studied in unit disk graphs[2] [36, 56, 139]. On the other hand, there is a polynomial-time $\frac{5}{4}$-approximation algorithm for finding a partition into a minimum number of cliques in graphs of maximum degree 3 [35].

3.3.2 Minimizing the number of links between the communities

Intuitively, a partition into communities should have a lot of links inside each community and few of them between communities. One can then think about partitioning a graph into k communities (for any integer k) by looking for a *k-cut* of minimum size. This problem has already been studied in the literature, and known to be solvable in polynomial time for $k = 2$ [70] and for any integer k [84]. More generally, if we put weights on edges (that would represent the strength of a relationship), one can look for a k-cut minimizing the sum of all costs of the edges of the cut. In this way, given a graph $G = (V, E)$ and an integer b, a *k-way partition* is a partition $P = (C_1, \cdots, C_k)$ of V such that $|C_i| \leq b$ for any $i \in \{1, \cdots, k\}$. A natural optimization problem is, given a graph $G = (V, E)$, to find a k-way partition P of V minimizing the total cost of all edges having the two endpoints in different parts of P. It has been proved in [78] that this problem is NP-hard, even for 2-way partitions with $b = \lceil \frac{|V|}{2} \rceil$. However, this problem and some variants have been widely studied and a lot of heuristic algorithms have been designed to solve it. An heuristic has been suggested for general k in [107] and for $k = 2$ in [39]. For general k and $b = \frac{|V|}{k}$, heuristics have been suggested in [41, 105]. A generalization with weights on vertices (the size of a subset is then the sum of the weights of its vertices) has been studied in [98] for $b = \frac{|V|}{k}$.

3.3.3 Imposing a certain neighborhood for members of communities

Similarly to Subsection 3.2.2, another aspect of communities is that each people should have more relationships in their community than out of it. In this way, we can design partitions into communities such that each vertex should have a certain amount of people they know in their own part rather than out of their part. Gerber and Kobler introduced in [80, 81] the problem of deciding if a given graph G has a vertex partition into two nonempty parts such that each vertex has at least as many neighbors in its part as in the other part. Given a graph G, we say that G is *partitionable* if G has such partition.

Some graphs are not partitionable, like complete graphs, stars, and complete bipartite graphs with at least one of the two vertex sets having odd size. On the other hand some other graphs are easily partitionable: cycles of length at least 4, trees which are not stars, and disconnected graphs.

Bazgan *et. al.* proved in [20] that the problem is NP-complete and polynomially equivalent to its balanced variant in which the partition is requested to be balanced (*i.e.*

[2] A unit disk graph is the intersection graph of some set of disks of diameter 1 in the euclidean plane.

both sets are requested to have the same size). The variant of finding a partition into two nonempty parts such that each vertex has at most as many neighbors in its part as in the other part has also been proved NP-complete in [21].

The previous definition of Gerber and Kobler gives a simple and natural way to define a partition into two communities. However, it could be more relevant to consider a condition involving the size of each part. In this way, Olsen [133] introduced the notion of *community structures*. A community structure is a partition of the vertex set into several parts such that for each vertex, the ratio between the number of neighbors in its own part and the size of its part (excluding the vertex itself) must be at least as large as the ratio between the number of neighbors in any other part and the size of this part. Informally, if someone belongs to a community structure, then he knows a greater proportion of his group than in any other group. In Figure 3.6, the white and black vertices form a satisfactory partition. However, the vertex x know only $\frac{1}{5}$ of its group whereas it knows half of the other group. It seems legit to consider that this member should not be happy to belong to the left community.

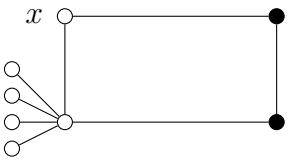


Figure 3.6: The black and white vertices form a satisfactory partition, but is not a community structure

This definition has been introduced very recently. Consequently, few results have been discovered. In Chapter 4, we investigate this definition further and prove that a community structure with two parts can be found in polynomial-time in some graph classes. We also investigate the problem of finding a community structure into two parts with the additional constraint of obtaining a balanced partition, *i.e.* a partition in which both parts are equal sized.

3.3.4 Ensuring stability

First introduced in social psychology in [97] by Heider, *structural balance* was defined in graphs theory by Cartwright and Harary in [33]. We consider a graph as a social network and all edges are labeled by "+" or "−", indicating if two members of the social network are friends or enemies. Such graph is called a signed graph.

The crucial idea with structural balance is the following. When we look at sets of three people at a time knowing each other (that we call a *triangle*), some configurations of +'s and −'s are socially and psychologically more stable than others, in the sense that the situation is unlikely to change in the future. In particular, there are four distinct ways

(up to symmetry) to label the three edges among those three people with +'s and −'s, see Figure 3.7.

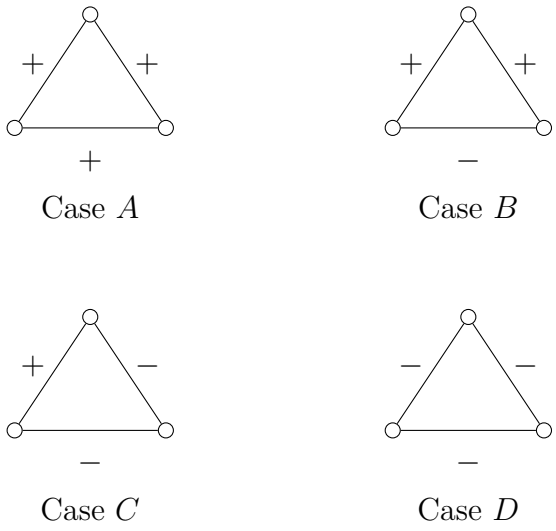


Figure 3.7: Structural balance: Each labeled triangle must have 1 or 3 positive edges

Within the four possible configurations, we distinguish two stable situations. A triangle having three "+" is a very stable situation: all three people are mutual friends (Case A). A triangle having only one "+" is also a stable situation since a group of two friends have a common enemy (Case C).

The other two possible labelings of the triangle introduce some amount of psychological stress or instability into the relationships. A triangle with two "+" and one "−" (Case B) corresponds to a situation in which a person have two friends who do not get along with each other. This situation is unstable since it pushes that person to choose one side and become an enemy of the other one, or could also push the two enemies to become friends. Finally, if the triangle has only "−", two of the three people could be motivated to team up against the third one, turning one of the three edge labels to a "+".

Now, a signed graph is said to be *balanced* (resp. *weakly balanced*) if it does not contain any triangle of case B and D (resp. case B). Such property ensures that the social network is stable in the sense that relationships are unlikely to change in the future. Checking if a graph is balanced can be useful in political contexts: by looking at the relationships between countries or populations, it is interesting to figure if some diplomatic tensions might appear or not. An appealing example of the evolution of alliances before World War I can be found in [8].

It is interesting to notice that a complete signed graph is balanced (resp. weakly balanced) if and only if the graph can be partitioned into two parts (resp. a certain number of parts) such that any edge with two endpoints in the same parts are labeled "+" and any edge with each endpoint in different parts are labeled "−" [33, 92]. If the graph

is not complete, then the graph is balanced if and only if it contains no odd cycle labeled "−" [33, 92].

On the other hand, checking if a graph can be partitioned into parts such that any edge with two endpoints in the same parts are labeled "+" and any edge with each endpoint in different parts are labeled "−" can be done easily in polynomial time: any pair of vertices x, y with an edge labeled "+" linking them can be merged into one vertex w preserving the original edges with the labels (*i.e.* for any edge xz or yz in the graph, we add the edge zw preserving the label). If an edge has two labels, then the graph is not partitionable. If not, the end of the editing will lead to a graph with only edges with a label "−" and the partition will be given by the merging.

The balanced property gives a good indicator of the stableness of a social network and gives a good partition into communities.

3.3.5 Evaluating the quality of a partition

Similarly to Subsection 3.2.3, we can design quality function to evaluate how relevant is a partition to reveal a community structure. The most famous one is *modularity* that has been a major measure in community detection and has been used in many works to evaluate the quality of partitions into communities [13, 24, 55].

The general idea of modularity is to assume that if the number of edges between vertices in a part is higher than what we would expect on the basis of random chance, then it would constitutes evidence of meaningful community structure. On the other hand, if the number of edges between groups is less than what we expect by chance, then it is reasonable to conclude that some structure is worth of interest.

Newman and Girvan [131] defined modularity, based on a previous measure proposed by Newman [130]. Modularity is defined as the fraction of the edges in the network that connect vertices within their community minus the expected value of the same quantity in a network with the same community divisions but edges are distributed randomly (preserving the degree of each vertex).

Optimizing the latter version of modularity has been proved NP-hard in [30]. Heuristics have been studied in the literature in order to give a partition with a good modularity [24, 31].

3.3.6 Overlapping partitions into communities

As we discussed before, finding a partition into communities is useful in lots of contexts like VLSI-circuits. However, regarding real world social network, asking for a partition may sometimes appear too restrictive. Indeed, some people might belong to several communities: it seems natural that some people have relationships with different groups of interest. In this way, given a graph $G = (V, E)$, we define an overlapping partition as a set $S = \{S_i\}_{i=1}^{p}$ of subsets of V such that $\cup_{i=1}^{p} S_i = V$. Contrary to regular partitions, we allow that $S_i \cap S_j \neq \emptyset$ for some i, j.

A first natural way to define an overlapping partition into communities is to ask for an overlapping partition into cliques, also called *clique cover*. However, the problem of finding a clique cover of size k is equivalent to finding a partition into k cliques: given a graph, it is easy to see that any clique partition is also a clique cover, and any clique cover of size k can be easily transformed into a partition into k clique by choosing arbitrarily only one set to belong to for any vertex that is in multiple parts.

On the other hand, one could argue that two cliques of size k sharing $k-1$ vertices should not be considered as two distinct communities. An easy solution consists in considering that two such sets should belong to a unique community. In this way, we can set that two cliques of size k are said to be adjacent if they share $k-1$ vertices. For a given integer k, a chain of adjacency $(C_1, \cdots, C_p)$ refers to a p-tuple of cliques of size k such that C_i and C_{i+1} are adjacent for any $i \in \{1, \cdots, p-1\}$. Then, Palla *et. al.* defined in [136] a *k-clique community* as a maximal union of cliques of size k such that for any two cliques C_1, C_2 of this union, there exists a chain of adjacency between C_1 and C_2. Building all k-clique communities gives an overlapping partition of the graph into communities that may share some members. To put it in another way, the overlapping partition corresponds to the connected component of the clique graph in which the vertices correspond to the cliques of size k in the original graph, and two cliques are linked by an edge if they are adjacent in the original graph (see Figure 3.8 as an illustration).

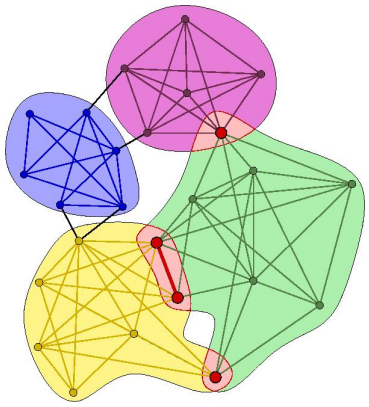


Figure 3.8: Example of an overlapping partition with 4-clique communities.

This definition has many applications, like in identification of protein communities involved in cancer metastasis [102, 103] and studies of social networks [86, 135].

Another way to define overlapping partition into communities is to partition the edges. In [60], Evans and Lambiotte investigated constructing a partition of the vertices of the line graph of the original graph. In this way, a partition of the line graph gives an overlapping partition of the vertices in the original graph, allowing vertices to belong to several communities. Then, such partition can be searched according to a certain criterion as it is discussed in Section 3.3. In [3], edge partition is used via hierarchical clustering for community detection applied in biological and social networks.

A specific state-of-art about overlapping partition into communities can be found in [157].

3.3.7 Finding a hierarchical clustering of communities

Another approach consists in constructing a hierarchical clustering, which gives another organization of a networks into communities. Given a graph $G = (V, E)$, a hierarchical clustering is a set S of subsets of V such that $V \in S$ and $\forall s_1, s_2 \in S$, $s_1 \subset s_2$ or $s_2 \subset s_1$ or $s_1 \cap s_2 = \emptyset$. This gives an organization of the vertices of G into communities, each set of S being considered as a community.

In order to express the hierarchy in a more clear way, we can define a hierarchical clustering in an equivalent way [29]: Given a graph $G = (V, E)$, and a community partition $P = \{C_1, C_2, \cdots, C_\ell\}$ of V, a sub-partition $P' = \{C'_1, C'_2, \cdots, C'_m\}$ of P is a partition of V such that $\forall C'_i \in P', \exists C_j \in P$ such that $C'_i \subseteq C_j$. A hierarchical community structure of G is defined as a series of partitions $P_k, P_{k-1}, \cdots, P_2, P_1, P_0$ with $P_0 = V$ and $P_k = \{v\}, v \in V$ such that P_i is a sub-partition of P_{i-1} for any $i \in \{1, \cdots, k\}$. Given a partition P_i, we can define i as the level of the partition P_i within the global tree of communities with $(k + 1)$ levels.

Such hierarchy can be illustrated by a dendrogram. Again, several legit criteria can be considered in order to construct this hierarchical clustering. A dendrogram can be used to visualize the final hierarchical clustering: any horizontal line gives a partition of the graph (See Figure 3.9 as an illustration given in [142]).

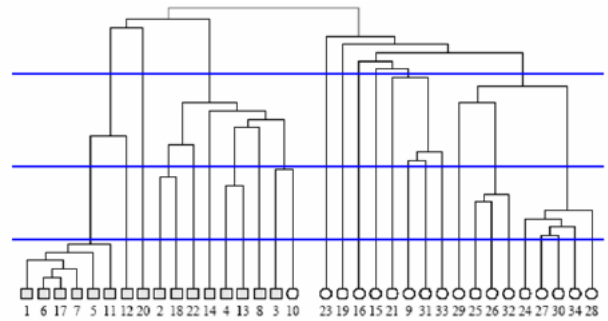


Figure 3.9: Dendrogram of the communities found in the Zachary Karate Club Network with the algorithm of Girvan and Newman in [82]. Each horizontal line gives a partition into communities.

Hierarchical clustering finds applications in bioinformatics and data analysis [62].

A very famous way to define a hierarchical clustering that describes a network into communities is the algorithm of Newman and Girvan [82] based on *edge betweenness*. Firstly, the notion of betweenness has been proposed by Freeman [74]. He defined the betweenness centrality of a vertex v as the number of shortest paths between any two vertices except v that contain v. This gives an indication of the influence of a vertex over

the flow of information between other vertices. This notion is particularly relevant when information flow over a network primarily follows the shortest available path. In a similar way, Newman and Girvan proposed to introduce the notion of edge-betweenness [82]. The edge-betweenness of an edge is defined as the number of shortest paths between any two vertices that contains it. If a network contains groups of people that are connected by a small amount of edges, then all shortest paths between those different groups must go along one of these few edges that will have then a high edge-betweenness (see Figure 3.10).

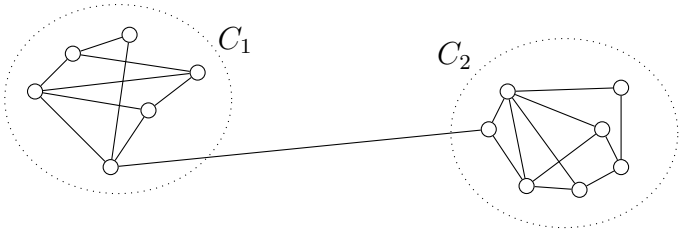


Figure 3.10: In this graph, the edge in the middle has a greater edge-betweenness than all other edges since all shortest paths connecting vertices from C_1 to C_2 run through it.

In [131], an efficient algorithm is proposed that consists in computing the betweenness of all edges first, and then repeating the two following steps in order to reveal a hierarchical clustering of the graph:

- Remove the edge with maximum edge-betweenness
- Compute again the edge-betweenness of all edges

Each step of the algorithm gives a partition of the graph into communities. Its final output is a hierarchical clustering containing all partitions given at each step of the algorithm.

On the other hand, it is also possible to describe a network of communities with a hierarchical clustering by using other features. As a first example, a method has been developed using random walk that is called *walktrap clustering*. It consists in assuming that a random walk starting from a vertex tends to stay in the community it belongs to. If we do a random walk starting from a vertex v, then the probability to reach a neighbor is $p = \frac{1}{d(v)}$, and then it is possible to compute the probability to reach a vertex j from a vertex i after k steps. Then we consider that two vertices are close if they have a similar probability to reach other vertices by a random walk of length k. The hierarchical clustering is made by first considering singletons communities, and then regrouping vertices that have the smallest distance between them, and compute again the distances. This way of describing communities has been studied in [79, 101, 141, 160], and has been used in order to study the similarity of words with applications in web navigation [79] or diffusion processes in small-world networks[3] [101].

[3]networks with a low diameter.

As a second example, we can consider that members of a network should be in the same community if they share the same neighborhood. Given a graph $G = (V, E)$, a *module* is a set of vertices $M \subset V$ such that for any two vertices $x, y \in M$, $N(x) \setminus M = N(y) \setminus M$. The concept of module has been introduced by Gallai in [75]. A strong module is a module that does not strictly overlap any other module. A graph is said to be prime if the only modules are the empty set, the set of all vertices and the singletons. Notice that given a graph $G = (V, E)$, if G is disconnected then all connected components of G are strong modules. In the same way, if $\overline{G}$ is disconnected, then then all connected components of $\overline{G}$ are strong modules.

If both G and $\overline{G}$ are connected, then Gallai showed in [75] that inclusion-wise maximal modules of G define a partition P of V which is called a modular partition. The quotient graph of G, in which each module of P is associated with one vertex, is prime.

Recursively, for each subgraph $G[M]$ induced by a module $M \in P$, either $G[M]$ (resp. $\overline{G[M]}$) is disconnected and thus can be partitioned into strong modules that are the connected components of $G[M]$ (resp. $\overline{G[M]}$), or G has a modular partition as seen before.

This recursion defines a modular decomposition that is a hierarchical clustering in which each partition is a partition of V into strong modules. Such hierarchical clustering can be represented by a dendrogram, called a modular decomposition tree. This tree is rooted by the set of vertices and all singletons are leaves.

A modular decomposition can be found in linear time [50]. A survey about modular decomposition can be found in [91].

4

Two-Community Structures

Contents

The content of this chapter is based on the following papers [14, 15, 16]:

- C. Bazgan, J. Chlebíková and T. Pontoizeau. *Structural and algorithmic properties of 2-community structures.* Algorithmica, 80(6):1890–1908, 2018.
- C. Bazgan, J. Chlebíková and T. Pontoizeau. *New Insight into 2-Community Structures in Graphs with Applications in Social Networks.* The 9th International Conference on Combinatorial Optimization and Applications (COCOA 2015), LNCS 9486, pp 236–250, 2015.

- C. Bazgan, J. Chlebíková, C. Dallard and T. Pontoizeau, *Family of graphs without 2-community structure*, The 10th International Colloquium on Graph Theory and combinatorics (ICGT 2018), 2018.

4.1 Introduction

In this chapter, we study the structural and complexity problems related to the recent definition of a community structure in graphs as defined in [58, 59, 133]. This definition reflects closeness between members of a community taking into account the number of neighbors of each member and the size of the communities. This new approach for communities is supported by the practical experiments showing the importance of capturing the sizes of communities for a better description of their properties [133].

Informally, a *community structure* in a graph is a partition of the vertex set such that each vertex has a greater proportion of neighbors in its part than in any other part. In this chapter, we focus on a partition with the restriction of outputting a community structure with two communities where the problems are already appealing. The presented techniques may offer some possibilities for an extension to a larger required number of communities.

We also introduce the concept of *weak community structure* in which the vertex itself contributes to the proportion of neighbors in its part. The ratio condition in the latter definition becomes weaker, but it reflects the reasonable requirement that each member should be considered as a part of its own community. Even if there are minor differences between the definitions, the structural and complexity results for the two problems are very different as it is presented in this paper. Both definitions are relevant to describe the community structures, the choice depends on the suitability of the model.

We also study the 2-community problems with additional constraints such as *connectivity* or *equality of sizes for both parts* (balanced partition). The connectivity request corresponds to the essential condition that each member in the community should 'indirectly know' all members in its own community, where the 'indirectly know' relation corresponds to a path between two vertices in the graph. The study of balanced communities is motivated by the practical interest for equal size of the communities. In general, the balanced graph partitions are well studied, e.g. due to its applications in the divide-and-conquer algorithms, see e.g. [40]. In the balanced partition problem, which can be seen as a generalization of the bisection problem to any given number of parts, the goal is to minimize the number of edges between partitions. It is known that the problem cannot be approximated within any finite factor in polynomial time in general graphs and it remains APX-hard even on trees of constant maximum degree [64]. It demonstrates that some graph partitions problems that are related to e.g. balanced communities are hard to solve even for restricted graph classes and indicates hardness of various problems related to a community structure too. Hence all positive results in community structure problems would be important to get better understanding of the differences between community and partition problems.

Furthermore, a community structure is in fact a graph partition with a restricted num-

ber of edges between parts, therefore the new results for communities may find applications in the areas similar to a graph partition such as parallel-computing, VLSI-circuit design, route planning [51] and divide-and-conquer algorithms [152].

There are only a few results related to our definition of a community structure. Olsen [133] proved that a community structure (without the condition on the exact number of communities) can be found in polynomial time in any graph with at least 4 vertices, except a star. Recently, Estivill-Castro *et al.* [59] claimed that the problem to find a k-community structure with restriction to all communities to be connected and equal size is NP-complete in general graphs, but polynomially solvable in trees. In [133] Olsen also proved that it is NP-complete to decide, whether there is a community structure in a graph in which a given set of vertices is included in a community. It is interesting to see that the problem of finding a community structure has been seen as a fractional hedonic game problem in [11].

The chapter is structured as follows. In Section 4.2 we introduce formally some notations and definitions of the studied problems and give some general observations about community structures. In Section 4.3 we show that in some well-studied graph classes a 2-community structure always exists and can be found in polynomial time, even with additional request for connectivity in both parts. In Section 4.4 we focus on the balanced 2-community structure and present the structural and algorithmic results in general graphs and some graph classes. Conclusions and open problems are provided in Section 4.6.

4.2 Preliminaries

In this section, we introduce definitions related to community structures and the related problems. Some general observations are also given in order to discuss the problem and give a better understanding of the concept of community structures.

4.2.1 k-community structures

We introduce Olsen's definition of a k-community structure from [133].

Definition 4.1. *A* k-community structure *for a connected graph* $G = (V, E)$ *is a partition* $\Pi = \{C_1, \ldots, C_k\}$ *of* V*,* $k \geq 2$*, such that* $\forall i \in \{1, \ldots, k\}, |C_i| \geq 2$*, and* $\forall v \in C_i, \forall C_j \in \Pi$*,* $j \neq i$*, the following holds:*

$$\frac{|N_{C_i}(v)|}{|C_i| - 1} \geq \frac{|N_{C_j}(v)|}{|C_j|}$$

The latter inequality will be called the *proportion condition* all along this chapter for easier reading.

In a more general sense, a community structure is a k-community structure for some integer k. The concept of community structure and the proportion condition of Definition 4.1 are quite intuitive to understand. As explained in the introduction, a community

structure corresponds to a partition of the vertex set such that each vertex has a greater proportion of neighbors in its part than in any other part. See Figure 4.1 for an example.

Figure 4.1: Two different 2-partitions for the same graph (given by the black and white colors) in which each part has at least 2 vertices. In the first partition, x does not satisfy the proportion condition of a community structure since the proportion of neighbors in the white part is $\frac{1}{3}$ but the proportion of neighbors in the black partition is $\frac{2}{3}$. The second partition gives a 2-community structure.

On the other hand, the proportion condition of the definition of a community structure can be relaxed. Indeed, it may have sense to consider that a member of a community considers itself as a part of his group. In this way, we define weak k-community structures as follows:

Definition 4.2. *A* weak k-community structure *for a connected graph $G = (V, E)$ is a partition $\Pi = \{C_1, \ldots, C_k\}$ of V, $k \geq 2$, such that $\forall i \in \{1, \ldots, k\}, |C_i| \geq 2$, and*

$$\frac{|N_{C_i}[v]|}{|C_i|} \geq \frac{|N_{C_j}(v)|}{|C_j|}$$

The latter inequality will be called the *weak proportion condition* all along this chapter for easier reading. In a more general sense, a weak community structure is a weak k-community structure for some integer k.

Notice that a k-community structure is obviously a weak k-community structure since for any partition $\{C_1, C_2, \ldots, C_k\}$ and any vertex v, $\frac{|N_{C_i}[v]|}{|C_i|} = \frac{|N_{C_i}(v)|+1}{|C_i|-1+1} \geq \frac{|N_{C_i}(v)|}{|C_i|-1}$, and thus if the proportion condition of Definition 4.1 is satisfied, it is also satisfied for the Definition 4.2. The other way direction is not always true, that is a weak k-community structure is not necessarily a k-community structure (see Figure 4.2).

Notice that if a graph has a community structure, it must have at least 4 vertices (since of the condition on the sizes of the parts) and not be isomorphic to a star. Indeed, any star never contains any community structure since all leaves need to be in the same part as the center of the star in order to satisfy the proportion condition of a community structure. Thus, any partition satisfying this condition could not have more than one part and thus would not be a community structure.

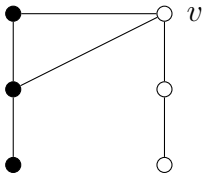


Figure 4.2: A weak 2-community structure of a graph (presented by the colors black and white) in which the vertex v does not satisfy the proportion condition of a 2-community structure but satisfies the weak proportion condition of a weak 2-community structure from Definition 4.2.

4.2.2 Studied problems

In this part we investigate community structures for a fixed number of two communities and also study some variants of the 2-Community problem:

2-Community **Input:** A graph $G = (V, E)$. **Question:** Does G have a 2-community structure?

A 2-community structure is a 2-partition $\{C_1, C_2\}$ of the vertex set V such that $|C_1|, |C_2| \geq 2$, and for each vertex $v \in C_i$, $i \in \{1, 2\}$,

$$\frac{|N_{C_i}(v)|}{|C_i| - 1} \geq \frac{|N_{C_{3-i}}(v)|}{|C_{3-i}|}$$

which is a re-writting of the proportion condition of Definition 4.1. Since a graph containing a 2-community structure must have at least 4 vertices and be non-isomorphic to a star, those assumptions are assumed in this chapter even without explicitly mentioning that in some informal parts.

In the Weak 2-Community problem we are looking for a weak 2-community structure in a graph where the proportion condition is replaced by its corresponding weak proportion condition:

$$\frac{|N_{C_i}[v]|}{|C_i|} \geq \frac{|N_{C_{3-i}}(v)|}{|C_{3-i}|}$$

Adding the balanced condition to the 2-Community problem, we obtain the Balanced 2-Community problem introduced by Estivill-Castro *et al.* [58]. Similarly we can define the Balanced Weak 2-community problem.

The additional constraint which asks for subgraphs induced by each part of the partition to be connected is a natural condition useful for the problems related to the connectedness. The Connected 2-Community problem is to decide if a graph has a connected 2-community structure, *i.e.* a 2-community structure $\{C_1, C_2\}$ such that the subgraphs induced by C_1, C_2 are connected. We can define analogous problems for weak and balanced versions.

4.2.3 General observations

An interesting result about community structure has been proved by Olsen in [133]. He showed that if there is no restriction on the number of parts, a community structure can always be found in polynomial time in any graph which is not a star.

Theorem 4.3 (Olsen 2013)**.** *A community structure can be computed in polynomial time for any connected graph $G = (V, E)$ containing at least four vertices, except stars.*

Proof. The proof consists in setting up a polynomial time local search among certain partitions of V and prove that the result of the local search is a community structure.

Specifying the search space $\mathscr{S}$. Considering a subset $C \subset V$, if we have a vertex $u \in C$ such that all vertices in $C \setminus \{u\}$ are neighbors of u, then we refer to u as a center of C. We now define the search space $\mathscr{S}$ as the set of all partitions Π of V with $|\Pi| \geq 2$ such that for all $C \in \Pi$, $|C| \geq 2$ and C has a center. The neighbors of a partition $\Pi \in \mathscr{S}$ are partitions in $\mathscr{S}$ that can be obtained from Π by moving one vertex from one set to another set of Π or by letting two vertices form a new set.

Generating one element of $\mathscr{S}$. We consider four different vertices u_1, u_2, u_3, u_4 such that $\{u_1, u_2\}, \{u_3, u_4\} \in E$. Now we expand the collection $\{\{u_1, u_2\}, \{u_3, u_4\}\}$ with one vertex at each step carefully making sure that each set contains at least two vertices and has a center. Sometimes we might have to form a new set containing the new vertex and another vertex. By using induction on the total number of vertices in the sets in the collection, it is always possible to add a vertex and still have a collection of sets with at least two vertices and a center. Thus an element $\Pi_1 \in \mathscr{S}$ can be computed in polynomial time.

The objective function. For a partition Π we define $g(\Pi)$ as the number of edges in the complement graph of G connecting vertices in different members of Π. More precisely, g is defined as follows:

$$g(\Pi) := \{uv \in \overline{E} : u \in C, v \in C', C \in \Pi, C' \in \Pi, C \neq C'\}$$

where $\overline{E}$ is the complement of E. We now use local search on $\mathscr{S}$ to maximize the number of parts and g, by defining the following objective function h:

$$h(\Pi) := |\Pi| + g(\Pi)$$

We start with an element Π_1 of $\mathscr{S}$ and pick any neighbor of Π_1 such that the value of h increases for that neighbor. The process is repeated until h cannot be increased, and the final partition Π' is returned as a community structure.

Analysis. We prove by contradiction that Π' is a community structure. Assume that there is a vertex $i \in V$ and $C \in \Pi'$ violating the proportion condition of a community structure, meaning that:

$$\frac{|N_i(C_i)|}{|C_i| - 1} < \frac{|N_i(C)|}{|C|} \tag{4.1}$$

where $C_i \in \Pi'$ is the part of Π' containing i. If i is a center then (4.1) cannot hold because the left hand side is 1. Consequently, the set C_i must contain at least three elements and i is not a center. We also note that i must have at least one neighbor in C, otherwise the right hand side would be 0. We now consider the following cases.

- If $|C| = 2$ and $|C_i| = 3$, the vertex i is connected to the center of C_i but not to the third vertex in C_i (otherwise the left hand side of (4.1) would be 1), and i must have both vertices in C as neighbors (otherwise the proportion condition would be satisfied for i) on C_i and C). If i is moved to C we would obtain a higher value of g without modifying the number of parts, thus the value of h would increase, contradiction.

- If $|C| = 2$ and $|C_i| \geq 4$, the vertex i has no neighbor in C_i except a center of C_i (otherwise i could establish a new set with such an additional neighbor and increase the number of parts without decreasing g). Then, there are at least two vertices in C_i that are not neighbors of i (denote the corresponding two non-edges $e', e'' \in \overline{E}$). Then, since there is at most one non-edge in $\overline{E}$ between $\{i\}$ and C which is contributing in $g(\Pi')$, moving i from C_i to C does not change the number of parts but increases the value of g by counting e' and e'', contradiction.

- If $|C| \geq 3$, similarly to the previous case, we can assume that i has a center of C and of C_i as neighbors but does not have any other neighbors in C_i or C. From (4.1), we obtain $|C_i| - 1 > |C|$ and g would increase if i is moved to C, contradiction.

The local search can be done in polynomial time since each step in the process can be carried out in polynomial time and $h(\Pi') \leq n + \frac{n(n-1)}{2}$. □

The previous algorithm uses local search in order to produce a community structure. Since a community may consist in two vertices linked by an edge, computing a community structure with the previous local search might lead to a community structure in which the number of communities is huge and the sizes of the communities are small (note that the algorithm split any clique into cliques of size 2 or 3). This is not always relevant for a proper community detection, especially when huge subsets of vertices that are strongly connected are never output as proper communities by the algorithm. A natural way to avoid this problem is to fix the number of communities in the community structure that we want to compute.

It can be noticed that there exist graphs in which it is easy to decide if there exists a k-community structure or not, for any integer k. In particular, as seen before, any star never contains any community structure. On the other hand, complete graphs on n vertices, $n \geq 4$, always contain a k-community structure for any integer $k \leq \lfloor \frac{n}{2} \rfloor$ since any k-partition in which each part contains at least 2 vertices is a k-community structure. Indeed, the proportion condition of a community structure is trivially satisfied since the left part of the inequation equals 1, *i.e.* all members knows everyone in its community (see Figure 4.3). More generally, for the same reasons, a partition of a graph in which each part is a clique is a community structure, and thus also a weak community structure.

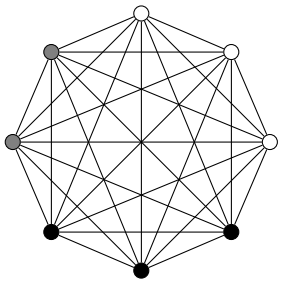


Figure 4.3: A complete graph in which a 3-community structure is given by the colors black, gray and white.

More generally, disconnected graphs can be an easy case when the number of connected components is at least 3 or if there are two connected components with at least 2 vertices. Indeed, any partition in which each part contains at least 2 vertices and corresponds to the union of some connected components is trivially a community structure since the right part of the proportion condition equals 0, *i.e.* each member does not know any members from the other communities.

However, because of the condition on the size of each community within a community structure, when there are only 2 connected components including an isolated vertex, there exists some graphs in which there is no community structure. As an example, in Figure 4.4, suppose first that x and y belongs to the same part of a community structure. Then, all vertices (except the isolated one) must belong to this part in order to fulfill the proportion condition since their only neighbors are x and y, and then the isolated vertex must be alone in a separated part, which is not possible because of the size restriction of communities. On the other hand, suppose that x and y are in different parts. Then, any neighbors of x (which are also neighbors of y) must be either with x or y as we discussed previously. Let be any partition such that x and y are not in the same part and each part is of size at least 2. Suppose that the isolated vertex belongs to the same part as x. Then, x does not satisfy the proportion condition of a community structure since the left part of the proportion condition is strictly less than 1 and the right part equals 1. By symmetry, if the isolated vertex belongs to the part containing y, the partition is not a community structure. Now if the isolated vertex belongs to a part not containing x nor y, then any other vertex of its part does not satisfy the proportion condition since this vertex has no neighbor in its part but has x and y as neighbors in the other parts.

Since such case is minor, we only investigate community structure in connected graphs in this chapter.

4.3 2-community structures in graph classes

This section contains our contribution about 2-community structures in some graph classes. In particular, we first show that if a graph has certain structural properties, then it has a connected 2-community structure which can be found in polynomial time. More precisely,

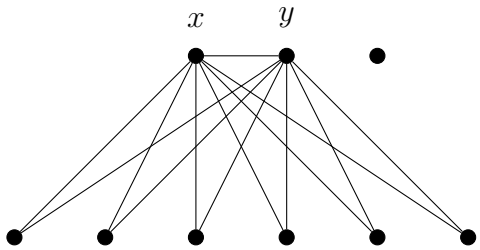


Figure 4.4: A disconnected graph with an isolated vertex in which there is no community structure.

we prove that such a statement is valid for complete bipartite graphs, complete split graphs, trees and graphs of high minimum or low maximum degrees.

4.3.1 Some graph classes in which the problem is easy to handle in linear time

Proposition 4.4. *Any complete bipartite graph has a connected* 2*-community structure. Moreover, such community structure can be found in linear time.*

Proof. Let $G = (V = V_1 \cup V_2, E)$ be a complete bipartite graph with V_1, V_2 the two independent sets. Wlog consider that $|V_1| \leq |V_2|$.

Suppose first that V_1 contains an odd number of vertices (let $|V_1| := 2k + 1$). If $|V_1| = |V_2|$, let k vertices from V_1 and k vertices from V_2 be in a set C_1 and let $C_2 := V \setminus C_1$. Then it can be checked that $\{C_1, C_2\}$ is a connected 2-community structure. Now assume that $|V_1| < |V_2|$.

We define a connected partition as follows: let k vertices from V_1 and $x := \lceil \frac{k|V_2|-k}{2k+1} \rceil$ vertices from V_2 be in a set C_1 and define $C_2 := V \setminus C_1$. Notice that $x \leq \lfloor \frac{k|V_2|-k}{2k+1} + \frac{2k+1}{2k+1} \rfloor = \lfloor \frac{k|V_2|+k+1}{2k+1} \rfloor$. We show that $\{C_1, C_2\}$ is a connected community structure.

Let a be a vertex of $V_1 \cap C_1$. Since $|V_2| \geq |V_1| \geq 2k = \frac{2k^2}{k} \geq \frac{2k^2}{k+1}$, we have $k|V_2| + |V_2| \geq 2k^2$. Thus $2k^2|V_2| - 2k^2 \geq 2k^2|V_2| + k|V_2| - 2k|V_2| - |V_2|$ and we obtain $\frac{k|V_2|-k}{2k+1} \geq \frac{k|V_2|-|V_2|}{2k}$. Thus, since $x \geq \frac{k|V_2|-k}{2k+1}$, we have $x \geq \frac{k|V_2|-|V_2|}{2k}$. Finally, we have $x|V_2| - x^2 + kx + x \geq k|V_2| + x|V_2| - |V_2| - kx - x^2 + x$ and we obtain $\frac{x}{x+k-1} \geq \frac{|V_2|-x}{|V_2|-x+k+1}$ and thus a satisfies the proportion condition of a community structure.

Let b be a vertex of $V_2 \cap C_1$. Since $x \leq \frac{k|V_2|+k+1}{2k+1}$, we have $|V_2|k + k^2 - xk + k \geq xk + k^2 - k + x + k - 1$. Thus $\frac{k}{x+k-1} \geq \frac{k+1}{|V_2|-x+k+1}$ and b satisfies the proportion condition of a community structure.

Let c be a vertex of $V_1 \cap C_2$. Since $|V_2| > |V_1|$, we have $|V_2| \geq 2k+2$. This implies that $\frac{k|V_2|+k+1}{2k+1} \leq \frac{|V_2|}{2}$. Since $x \leq \frac{k|V_2|+k+1}{2k+1}$, we have $x \leq \frac{|V_2|}{2}$. Thus, we have $|V_2|x + |V_2|k - x^2 - xk \geq |V_2|x - x^2 + kx$. Thus $\frac{|V_2|-x}{|V_2|-x+k} \geq \frac{x}{x+k-1}$ and c satisfies the proportion condition of a community structure.

Let d be a vertex of $V_2 \cap C_2$. Since $x \geq \frac{k|V_2|-|V_2|}{2k+1}$ (as seen previously), we have $kx+k^2+x+k \geq |V_2|k-kx+k^2$. Thus $\frac{k+1}{|V_2|-x+k} \geq \frac{k}{x+k-1}$ and d satisfies the proportion condition of a community structure.

Since $G[C_1]$ and $G[C_2]$ are clearly connected, $\{C_1, C_2\}$ is a connected 2-community structure.

Now suppose that V_1 contains an even number of vertices. Then consider $\frac{|V_1|}{2}$ vertices from V_1 and $\lfloor \frac{|V_2|}{2} \rfloor$ vertices from V_2 in a set C_1 and put $C_2 := V \setminus C_1$, and it is easy to check that $\{C_1, C_2\}$ is a connected 2-community structure. □

Proposition 4.5. *Any complete split graph has a 2-community structure. Moreover, such community structure can be found in linear time.*

Proof. The proof is similar to the one for bipartite graphs considering V_1, V_2 as the clique and the independent set of the split graph. The only difference is that any vertex in the clique part will always trivially satisfy the proportion condition since it has all of the other vertices of its part as neighbors. □

Theorem 4.6. *Every tree with at least 4 vertices (except a star) has a connected 2-community structure that can be found in linear time.*

Proof. Let $G = (V, E)$ be a tree not isomorphic to a star. We prove that there exists an edge $e \in E$ such that the two connected components of $G \setminus e$ form a 2-partition which is a connected 2-community structure.

Let $e = \{u, v\}$ be an edge in E such that $d(v)$, $d(u) \geq 2$ (due to the assumption about G such an edge e must exist). Consider a partition $\{X_u, X_v\}$ of V with X_u (resp. X_v) be the set of vertices of the connected component of $G \setminus e$ containing u (resp. v).

First we notice that only one of the vertices u and v may not satisfy the proportion condition. If this is not true then $\frac{d(u)-1}{|X_u|-1} < \frac{1}{|X_v|}$ and $\frac{d(v)-1}{|X_v|-1} < \frac{1}{|X_u|}$. Since $d(u), d(v) \geq 2$, it implies $|X_v| < \frac{|X_u|-1}{d(u)-1} \leq |X_u| - 1$ and $|X_u| < \frac{|X_v|-1}{d(v)-1} \leq |X_v| - 1$, which is not possible.

If both vertices u and v satisfy the proportion condition, then $\{X_u, X_v\}$ is obviously a 2-community structure. If not, then without loss of generality, let the vertex u satisfy the proportion condition and v do not. Then the UPDATE PROCEDURE is repeated and if no update is possible, a modified partition $\{X_u, X_v\}$ is already a 2-community structure as it is shown later.

The UPDATE PROCEDURE:

Let $v_1, v_2, \ldots, v_{d(v)-1}$ be the neighbors of v excluding u (there is at least one such a vertex due to our assumption $d(v) \geq 2$). For each i, $1 \leq i \leq d(v)-1$, and $e_i = \{v, v_i\} \in E$, let X_i be the set of vertices of the connected component in $G \setminus e_i$ containing v_i. Notice that if for all j, $1 \leq j \leq d(v) - 1$, $d(v_j) = 1$, then v must already satisfy the proportion condition in the partition $\{X_u, X_v\}$ at the beginning of the UPDATE PROCEDURE.

Hence from now we suppose that v has at least one neighbor of degree at least 2 excluding u. In the following we show that there exists j, $1 \leq j \leq d(v) - 1$, such that

$d(v_j) > 1$ and the vertex v satisfies the proportion condition in the partition $\{X_j, V \setminus X_j\}$. Indeed, suppose that for all j, $1 \leq j \leq d(v) - 1$, with $d(v_j) > 1$, this is not true. Notice that for each such j and the partition $\{X_j, V \setminus X_j\}$ must hold $\frac{d(v)-1}{n-|X_j|-1} < \frac{1}{|X_j|}$ which implies that:

$$d(v)|X_j| < n - 1. \tag{4.2}$$

Moreover, for any j, $1 \leq j \leq d(v) - 1$ with $d(v_j) = 1$ we have $|X_j| = 1$ and hence:

$$d(v)|X_j| < n - 1, \tag{4.3}$$

since G is not a star. Recall that v doesn't satisfy the proportion condition in the partition $\{X_u, X_v\}$, hence $\frac{d(v)-1}{|X_v|-1} < \frac{1}{|X_u|}$ and also:

$$d(v)|X_u| < n - 1, \tag{4.4}$$

Summing (4.2), (4.3) and (4.4) together, we obtain $d(v) \sum_{j=1}^{d(v)} |X_j| = d(v)(n-1) < d(v)(n-1)$, a contradiction.

Hence, there exists i, $1 \leq i \leq d(v) - 1$ such that $d(v_i) > 1$ and the vertex v satisfies the proportion condition in the partition $\{X_i, V \setminus X_i\}$. Then, relabel $u := v$ and $v := v_i$ and return to the beginning of the Update procedure.

Each time the labels of u and v are updated, the size of X_u strictly increases by at least one, hence the whole process always terminates. A final partition at the end of the process is a connected 2-community structure because both partitions correspond to two connected components of a tree obtained by removing an edge.

Notice that finding such an edge can be done in $O(|V|)$ operations. First, in constant time fix an edge $e = \{u, v\}$ such that $d(v), d(u) \geq 2$. Then, consider $G \setminus e$ as a union of two trees T_u and T_v, where T_u is a tree on the vertex set X_u rooted in u (and similarly for T_v on X_v rooted in v). For each vertex w of G calculate recursively the size of the subtree of T_u (or T_v) rooted in w which can be done in time $O(|V|)$. Finally, using the sizes of the subtrees, check if $\{X_u, X_v\}$ corresponds to a 2-community structure and if needed, update X_u, X_v according to the algorithm. The number of such updates is clearly at most $|E|$. Since G is a tree, the repetition of the Update procedure finishes with a connected 2-community structure in $O(|V|)$ time. □

Very recently, Estivill-Castro *et al.* proved in [59] the same result using different methods. Our approach is more structural and the proof for the existence of an edge that connects two communities results directly in a linear time algorithm.

4.3.2 Cubic graphs and graphs of maximum degree 3

Now we investigate other graph classes that have low densities. We first prove that 2-community structures always exist in cubic graphs, and then we extend the result to graphs of maximum degree 3. In both cases, it is possible to design an algorithm running in polynomial time to guarantee the connectedness of each part of the 2-community structure.

First, the restrictions on the size of partitions are discussed to ensure the vertices fulfil the proportion condition of a 2-community structure.

Lemma 4.7. *Let $G = (V, E)$ be a graph of maximum degree 3 of size n. Let $\{C_1, C_2\}$ be a partition of V such that $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$. Then each vertex of degree 3 in G with at most one out-neighbor fulfils the proportion condition of a 2-community structure.*

Furthermore, if for some $i \in \{1, 2\}$, $|C_i| = \lceil \frac{n-1}{3} \rceil$ (or also $|C_i| = \lceil \frac{n-1}{3} \rceil + 1$ in case $n \equiv 1 \mod 3$) then each vertex of degree 3 in C_i with two out-neighbors fulfils the proportion condition too.

Proof. Let $\{C_1, C_2\}$ be a fixed partition of G such that $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$. It is clear that the proportion condition is true for each vertex which has only neighbors in its own part. Firstly, suppose the vertex v from C_i, $i \in \{1, 2\}$ has exactly one out-neighbor.

Since $|C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, then obviously $|C_i| \leq n - \frac{n-1}{3}$ and $\frac{2}{|C_i|-1} \geq \frac{1}{n-|C_i|}$. Therefore the proportion condition is fulfilled for the vertex v.

Now suppose that for $i \in \{1, 2\}$ there is a vertex $v \in C_i$ with exactly two out-neighbors and $|C_i| = \lceil \frac{n-1}{3} \rceil$. Obviously, $\lceil \frac{n-1}{3} \rceil \leq \frac{n+2}{3}$ and hence $2\lceil \frac{n-1}{3} \rceil - 2 \leq n - \lceil \frac{n-1}{3} \rceil$ which implies $\frac{1}{\lceil \frac{n-1}{3} \rceil - 1} \geq \frac{2}{n - \lceil \frac{n-1}{3} \rceil}$. This corresponds to the proportion condition for the vertex v. Similarly if $|C_i| = \lceil \frac{n-1}{3} \rceil + 1$ and $n \equiv 1 \mod 3$: $n - 1 = 3\lceil \frac{n-1}{3} \rceil$ which implies $\frac{1}{\lceil \frac{n-1}{3} \rceil} \geq \frac{2}{n - \lceil \frac{n-1}{3} \rceil - 1}$. □

This lemma can be used to design a polynomial time algorithm which outputs a 2-community structure with a cubic graph in input.

Theorem 4.8. *Every 3-regular graph has a 2-community structure. Moreover it can be found in polynomial time.*

Proof. Let $G = (V, E)$ be a 3-regular graph of size n. The algorithm runs in two stages.

Stage 1: The algorithm finds a partition $\{C_1, C_2\}$ of V such that $|C_1| = \lceil \frac{n-1}{3} \rceil$ and at most two vertices from C_1 have more than one neighbor in C_2.

Stage 2: The algorithm moves some vertices between C_1 and C_2 until $\lceil \frac{n-1}{3} \rceil \leq |C_1| \leq n - \lceil \frac{n-1}{3} \rceil$ and each vertex of G has a restricted number of neighbors out of its own part in a such way that Lemma 4.7 can be applied.

Stage 1:
Let $u, v \in V$ be such that $uv \in E$ and put $C_1 = \{u, v\}$. Now repeat the following steps (S1) and (S2) until $|C_1| = \lceil \frac{n-1}{3} \rceil$:

(S1) Let w be a neighbor of u (or v) which is not in C_1, put $C_1 := C_1 \cup \{w\}$, $u := w$ (or $v := w$).

(S2) If there is no such vertex w, the degree of each vertex in the subgraph induced by C_1 is 2 or 3. In such a case let u be any vertex of degree 2 in the subgraph induced by C_1.

It is clear that at the end of the first stage the algorithm finishes with a set C_1 such that $|C_1| = \lceil \frac{n-1}{3} \rceil$. If it is not possible to apply (S1) and (S2) and $|C_1| < \lceil \frac{n-1}{3} \rceil$ then all vertices in C_1 must have all neighbors in C_1 which means that G is not connected.

Furthermore at most two vertices (u and v) from C_1 may have more than one neighbor outside C_1 and the subgraph induced by C_1 is connected. Define $C_2 = V \setminus C_1$.

Stage 2: We distinguish two major cases:

Case 1: If $\forall w \in C_2$, $d_{out}(w) \leq 1$ then all vertices in G except u and v have at most one neighbor out of its part. Using Lemma 4.7, these vertices fulfil the proportion condition. Moreover, $\lceil \frac{n-1}{3} \rceil$ equals $\lfloor \frac{n}{3} \rfloor$ or $\lceil \frac{n}{3} \rceil$ and according to Lemma 4.7, the proportion condition is also true if u or v have two neighbors out of C_1. Hence, $\{C_1, C_2\}$ is a 2-community structure.

Case 2: There exists a vertex $w \in C_2$, such that $d_{out}(w) \geq 2$.

Now we distinguish several subcases:

(A) $\forall x \in C_1, d_{out}(x) \leq 1$

(B) All vertices from C_1 which have more than one neighbor outside C_1 are adjacent to w (only u, $v \in C_1$ are possible candidates).

(C) No vertex from C_1 which has more than one neighbor outside C_1 is adjacent to w (only $u, v \in C_1$ are possible candidates).

(D) Both vertices $u, v \in C_1$ have more than one neighbor outside C_1, but only one of them is adjacent to w.

Case 2(A): Repeat the update step while it is possible before returning $\{C_1, C_2\}$ as a 2-community structure:

- if $\exists z \in C_2$, $d_{out}(z) \geq 2$, update C_1 and C_2 as follows: $C_1 := C_1 \cup \{z\}, C_2 := C_2 \setminus \{z\}$.

After each update step, two neighbors of z in C_1 have degree three by assumption of Case 2(A) (See Figure 4.5). After repeating the update step k times, C_1 has $\lceil \frac{n-1}{3} \rceil + k$ vertices and at least $2k$ vertices in C_1 have degree three in the subgraph induced by C_1. Hence, we can repeat the update step at most $\lceil \frac{n-1}{3} \rceil - 1$ times. Otherwise the degree of each vertex in C_1 is three which implies G is not connected. Thus $|C_1| \leq 2\lceil \frac{n-1}{3} \rceil - 1 \leq n - \lceil \frac{n-1}{3} \rceil$. Note that now every vertex in G has at most one neighbor out of its part and thus applying Lemma 4.7, $\{C_1, C_2\}$ is a 2-community structure.

Case 2(B): Wlog we suppose that $d_{out}(u) = 2$ and u is adjacent to w. Furthermore, if $d_{out}(v) = 2$ then also v is adjacent to w.

Now using C_1, C_2 we define a 2-community partition. After the initial update: $C_1 := C_1 \cup \{w\}, C_2 := C_2 \setminus \{w\}$, all vertices in C_1 have at most one neighbor in C_2 and $|C_1| = \lceil \frac{n-1}{3} \rceil + 1$. Then repeat the update step until $|C_1| = 2\lceil \frac{n-1}{3} \rceil - 1$ or if there is no such a vertex z to update C_1:

- if $\exists z \in C_2, d_{out}(z) \geq 2$, define $C_1 := C_1 \cup \{z\}, C_2 := C_2 \setminus \{z\}$,

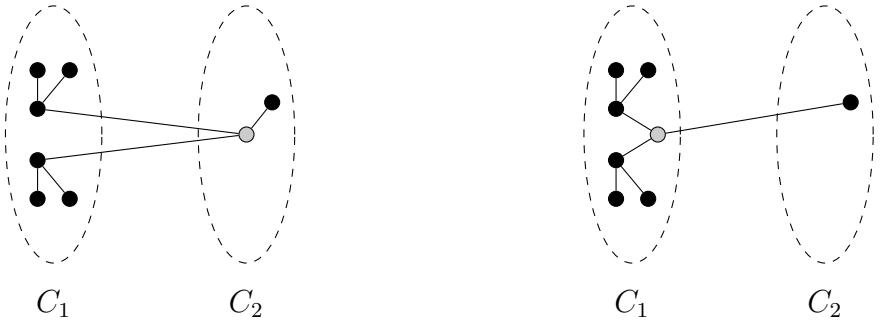


Figure 4.5: Applying one step in Case 2(A) on the gray vertex decreases the size of the cut by one and creates two vertices in C_1 with 3 in-neighbors.

There are two possible scenarios:

(i) if $|C_1| \leq 2\lceil\frac{n-1}{3}\rceil - 1$ and there is no such vertex $z \in C_2$ such that $d_{out}(z) \geq 2$, then each vertex in G has at most one neighbor out of its own part. Obviously, $|C_1| \leq n - \lceil\frac{n-1}{3}\rceil$ and due to Lemma 4.7, $\{C_1, C_2\}$ is a 2-community structure.

(ii) $|C_1| = 2\lceil\frac{n-1}{3}\rceil - 1$ and $\exists z \in C_2$ such that $d_{out}(z) \geq 2$: the update step has been repeated $\lceil\frac{n-1}{3}\rceil - 2$ times and in each step the number of vertices $x \in C_1$ with $d_{out}(x) = 1$ is decreased by at least 2 (the neighbors of z in C_1). It means every vertices in V has all neighbors in its own part, except at most three vertices, each having one neighbor out of its part.

– If $n \equiv 2 \mod 3$, then the size of $|C_2| = n-(2\lceil\frac{n-1}{3}\rceil-1) = \lceil\frac{n}{3}\rceil$. Because $|C_1| \leq n-\lceil\frac{n-1}{3}\rceil$, due to Lemma 4.7, all vertices with at most one neighbor out of its own part fulfil the proportion condition. If a vertex of C_2 is adjacent to exactly two vertices from C_1 then the proportion condition is true according to Lemma 4.7. A vertex of C_2 cannot be adjacent to all three vertices in C_1, otherwise $C_1 \cup \{w\}$ is a disconnected part of G. Hence, $\{C_1, C_2\}$ is a 2-community partition.
– If $n \equiv 0 \mod 3$ or $n \equiv 1 \mod 3$, then define the last update: $C_1 := C_1 \cup \{z\}, C_2 := C_2 \setminus \{z\}$. Now only one vertex of C_1 has one neighbor in C_2. Because $|C_1| = 2\lceil\frac{n-1}{3}\rceil \leq n - \lceil\frac{n-1}{3}\rceil$, the proportion condition is true for all vertices of G because of Lemma 4.7. Hence the updated partition $\{C_1, C_2\}$ is a 2-community structure.

Case 2(C): Without loss of generality, we suppose that $d_{out}(u) = 2$. Update $C_1 := C_1 \cup \{w\} \setminus \{u\}, C_2 := V \setminus C_1$.

Notice that after the update, $|C_1| = \lceil\frac{n-1}{3}\rceil$ and there may be at most two vertices in C_1 which have two neighbors in C_2. Hence we are again in one of the cases (A)-(D) of second stage, but each time we apply this update the size of the cut between C_1 and C_2 is decreases by two. Therefore the process is finite.

Case 2(D): Without loss of generality, suppose that u is adjacent to w and $d_{out}(u) = 2$. Update $C_1 := C_1 \cup \{w\} \setminus \{v\}$, $C_2 := V \setminus C_1$.

Notice that after the update, $|C_1| = \lceil \frac{n-1}{3} \rceil$. Moreover, u has two neighbors inside C_1 since u is obviously not adjacent to v. Hence we are in one of the previous cases of stage 2, since there is at most one vertex in C_1 (the neighbor of v) which could have two neighbors in C_2. □

Now we investigate the problem of finding a 2-community structure in 3-regular graphs with additional condition of connectivity for each part. Using the algorithm from Theorem 4.8 as a tool we extend the result in 3-regular graphs for a connected 2-community structure, but with many fine details in the proof. We first introduce a new lemma to make the construction more simple to read.

Lemma 4.9. *Let G be a 3-regular graph and $\{C_1, C_2\}$ a connected 2-partition of G with $\lceil \frac{n-1}{3} \rceil \leq |C_1| \leq n - \lceil \frac{n-1}{3} \rceil$ such that each part has at most one vertex with two neighbors out of its own part. Then G has a connected 2-community structure which can be found in polynomial time.*

Proof. The main idea is to move selected vertices between two parts in such a way that it preserves connectivity and offers the option to use Lemma 4.7.

We discuss four cases depending on which vertices have two neighbors out of its own part. Notice that transferring a vertex which has two neighbors out of its part does not compromise the connectivity of the partition.

(a) If there is no vertex in C_1 and C_2 with two neighbors out of its own part, then using Lemma 4.7 the partition $\{C_1, C_2\}$ is already a connected 2-community structure.

(b) If the only vertex with two neighbors out of its own part is in C_2, then update C_1, C_2 using the following loop:

- While $|C_1| < n - \lceil \frac{n-1}{3} \rceil$ and there exists a vertex z in C_2 which has two neighbors in C_1, update $C_2 := C_2 \setminus \{z\}$, $C_1 := C_1 \cup \{z\}$.

Obviously after each run of the while loop both parts of the partition remains connected. At the end of the while loop
– if $|C_1| < n - \lceil \frac{n-1}{3} \rceil$, then all vertices in G have at most one neighbor out of their parts and satisfy the properties of 2-community structure due to Lemma 4.7,
– if $|C_1| = n - \lceil \frac{n-1}{3} \rceil$ then $|C_2| = \lceil \frac{n-1}{3} \rceil$ and hence all vertices in C_2 with two neighbors out of the own part satisfy the properties of 2-community structure due to Lemma 4.7, the rest of vertices also satisfy Lemma 4.7.

(c) The only vertex with two neighbors out of its own part is in C_1. Then the case is similar to (b) by symmetry swapping the roles between C_1 and C_2.

(d) There are two vertices:
– $v_1 \in C_1$ with two neighbors in C_2 and let v_1^0 be the neighbor of $v_1 \in C_1$;
– $v_2 \in C_2$ with two neighbors in C_1 and let v_2^0 be the neighbor of $v_2 \in C_2$.

Now we need to distinguish two cases:

(i) If $v_1v_2 \in E$, then we update the partition as follows. If $|C_1| < n - \lceil \frac{n-1}{3} \rceil$ then define a new partition $C_1 := C_1 \cup \{v_2\}$; $C_2 := C_2 \setminus \{v_2\}$, otherwise $C_1 := C_1 \setminus \{v_1\}$; $C_2 := C_2 \cup \{v_1\}$. Obviously, $\{C_1, C_2\}$ is a connected partition which fulfil initial conditions of lemma, so we can apply case (a), (b), (c) or (d) again. Notice that the case (d) can be repeated only finite number of times since the cut size between C_1 and C_2 decreases each time the case is applied.

(ii) If $v_1v_2 \notin E$ we define the following update $C_1 := C_1 \cup \{v_2\} \setminus \{v_1\}$, $C_2 := C_2 \cup \{v_1\} \setminus \{v_2\}$. The new partition is a connected partition with no change in sizes and the following options are possible:

- If $d_{out}(v_1^0) = 0$ or $d_{out}(v_2^0) = 0$ before the update, then update removes at least one of the vertices with two outgoing edges. Now we can again apply one of cases (a), (b) or (c) which leads to a connected 2-community structure.
- If $d_{out}(v_1^0) > 0$ and $d_{out}(v_2^0) > 0$ then we can apply case (d) again. This process is finite because each time the size of the cut size between C_1 and C_2 is decreased by 2.

Obviously, the whole procedure can be run in polynomial time. □

Now we can prove the following theorem:

Theorem 4.10. *Every 3-regular graph has a connected 2-community structure. Moreover it can be found in polynomial time.*

Proof. The algorithm runs in two stages similarly to the algorithm in Theorem 4.8.

Stage 1: The algorithm finds either a connected partition $\{C_1, C_2\}$ such that $|C_1| = \lceil \frac{n-1}{3} \rceil$ and at most two vertices from C_1 have two neighbors in C_2 or ends up with a connected 2-community structure.

Stage 2: Apply directly Stage 2 from Theorem 4.8.

The difference to the approach from Theorem 4.8 is that C_2 remains connected until the end of the first stage, where C_1 is connected in both approaches.

Then we apply the second stage of the algorithm from Theorem 4.8. Since moving a vertex which has 2 neighbors in the other part never disconnect any part and all transfers only affect such vertices, the final partition $\{C_1, C_2\}$ remains connected at the end of the second stage.

Stage 1: (for a connected partition)
Choose any vertices $u, v \in V$ such that $\{u, v\} \in E$ and the subgraph induced by $V \backslash \{u, v\}$ is connected. Label the vertices u, v and define $C_1 := \{u, v\}$, $C_2 := V \backslash \{u, v\}$.

The initial construction:

While $|C_1| < \lceil \frac{n-1}{3} \rceil$ and one of the updates (S1), (S2) (in this order) can be applied do:

(S1) If there exists a vertex $x \in C_2$ such that $d_{out}(x) = 2$, then update $C_1 := C_1 \cup \{x\}$, $C_2 := C_2 \backslash \{x\}$. If all labelled vertices have three neighbors in C_1, then removes all labels and label one vertex in C_1 which has one neighbor in C_2.

(S2) If there exists a vertex $x \in C_2$ such that x is a neighbor of a labelled vertex w in C_1 and $C_2 \backslash \{x\}$ remains connected then update $C_1 := C_1 \cup \{x\}$, $C_2 := C_2 \backslash \{x\}$, label the vertex x and remove label from w. If all labelled vertices have three neighbors in C_1, then removes all labels and label one vertex in C_1 which has one neighbor in C_2.

Obviously after each update we can have at most two labelled vertices in C_1.

Now there are two possibilities how the initial construction can finish:

(1) The algorithm finishes with $|C_1| = \lceil \frac{n-1}{3} \rceil$. Due to the properties of the construction, the partition $\{C_1, C_2\}$ is connected and at most two vertices from C_1 may have two neighbors in C_2. In such a case we can move directly to Stage 2.

(2) If none of the updates (S1), (S2) can be applied and $|C_1| < \lceil \frac{n-1}{3} \rceil$ then we redefine the partition $\{C_1, C_2\}$ using the major update construction to obtain a new partition which leads to a connected 2-community structure.

The major update construction:

Step A: Let $N \subseteq C_2$ be the set of neighbors of all labelled vertices from C_1, hence $1 \leq |N| \leq 4$. Step A consists in splitting C_2 by defining subsets Q, Q', Z of C_2 and vertices q, $q' \in N$ (q, q' are not necessarily distinct) such that $\{Q, Q', Z, \{q\}, \{q'\}\}$ is a connected partition of the graph induced by C_2 and each vertex from $Q \cup Q'$ has at most one neighbor in C_1. Furthermore, the entire set Q (resp. Q') has exactly one neighbor outside Q (resp. Q') in C_2 and this is the vertex $q \in N$ (resp. $q' \in N$).

We show that such Q, Q', Z exist and are always connected. Consider a vertex $v_1 \in N$. The vertex v_1 has necessarily two neighbors in C_2 and the subgraph induced by $C_2 \setminus \{v_1\}$ is disconnected (otherwise (S1) or (S2) from Stage 1 could be applied). Let C_2^1 and C_2^2 be the two connected components of $C_2 \setminus \{v_1\}$. Define $Q := C_2^1$, $Q' := C_2^2$ and $q = q' = v_1$. Moreover, $Z = \emptyset$ is trivially connected. Hence in case $|N| = 1$ we can now move directly to Step B.

If $|N| > 1$, select another vertex $v_2 \in N$. Such vertex must be in Q or Q' defined above. Consider wlog that $v_2 \in Q$. If $v_1 v_2 \in E$, then update $Q := Q \setminus \{v_2\}$, $q := v_2$, $q' := v_1$ and $Z := \emptyset$. Notice that Q is still connected since v_2 has only one neighbor in Q. If $v_1 v_2 \notin E$, $Q \setminus \{v_2\}$ must be disconnected into two part Q^1 and Q^2. Name Q^2 the set which contains

a neighbor of v_1 (Q^1 obviously cannot have a neighbor of v_1, since the other two neighbors are in Q' and C_1, respectively). Update $Q := Q^1$, $q := v_2$ and $Z := Q^2$. Hence in case $|N| = 2$, the construction in Step A is over and we can continue directly with Step B. Indeed, Q has only $v_2 \in N$ as a neighbor in $C_2 \setminus Q$ and similarly for Q' and $v_1 \in N$.

Suppose now that $|N| \geq 3$. Then select another vertex $v_3 \in N$, the vertex v_3 must be in Q, Q' or Z defined above. If $v_3 \in Z$, then Q, Q', Z, q and q' already satisfy the properties. Otherwise, $v_3 \in Q \cup Q'$ and wlog we suppose $v_3 \in Q$. If $v_3v_2 \in E$, then update $Q := Q \setminus \{v_3\}$, $q := v_3$ and $Z := Z \cup \{v_2\}$ which is trivially connected. Otherwise, $Q \setminus \{v_3\}$ must be disconnected, let Q^1 and Q^2 be its connected parts. Denote Q^2 the set which has a neighbor of v_2 (then Q^1 cannot have a neighbor of v_2). Update $Q := Q^1$, $q := v_3$, $Z := Z \cup Q^2 \cup \{v_2\}$. Again, if $|N| = 3$, the construction is over. Indeed, Q has only $v_3 \in N$ as a neighbor in $C_2 \setminus Q$ and there are no changes for Q'. Moreover, Z is connected. Hence in case $|N| = 3$ we can move to Step B. If $|N| = 4$, the construction is similar to the discussion for $|N| = 3$ (see Fig. 4.6 as an example for $|N| = 4$).

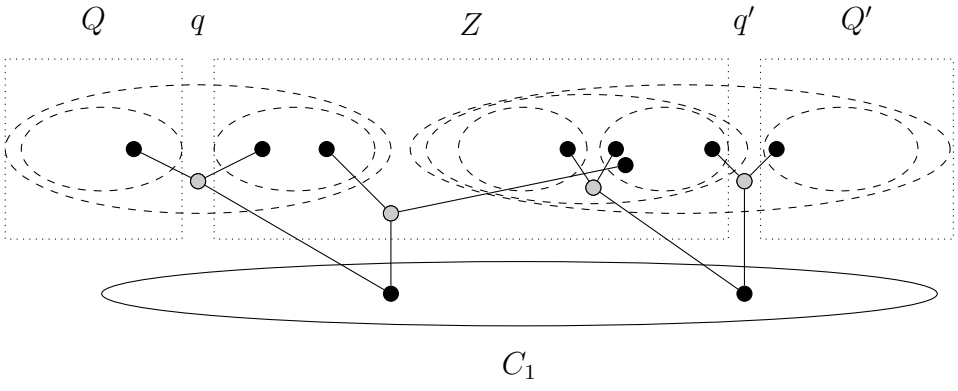


Figure 4.6: Splitting C_2 when $|N| = 4$ (vertices in N are in gray)

It is also important to notice that if $|N| > 2$ then following the construction $N \setminus \{q, q'\} \subseteq Z$.

Step B: This step consists in looking at the size of the sets $Z, Q, Q', Z \cup Q \cup \{q\}, Z \cup Q' \cup \{q'\}$ and update $\{C_1, C_2\}$ depending on the size of those sets which are known to be connected by moving entire sets of vertices instead of moving them one by one.

(i) If $|Q| > n - \lceil \frac{n-1}{3} \rceil$ or $|Q'| > n - \lceil \frac{n-1}{3} \rceil$ then define $\widetilde{C}_1 := V \setminus Q$, $\widetilde{C}_2 := Q$ (similarly for Q') and remove all labels. Then label q and apply again the updates of the initial construction on the partition $\{\widetilde{C}_1, \widetilde{C}_2\}$.

Note. The size of $\widetilde{C}_1$ is still strictly less than $\lceil \frac{n-1}{3} \rceil$ but it has strictly grown and the partition $\{\widetilde{C}_1, \widetilde{C}_2\}$ is connected, so the update is correct.

(ii) If $\lceil \frac{n-1}{3} \rceil \leq |Q| \leq n - \lceil \frac{n-1}{3} \rceil$ or $\lceil \frac{n-1}{3} \rceil \leq |Q'| \leq n - \lceil \frac{n-1}{3} \rceil$, define $\widetilde{C}_1 := V \setminus Q$, $\widetilde{C}_2 := Q$ (similarly for Q'). The vertices in $\widetilde{C}_1$ have at most one neighbor in $\widetilde{C}_2$ and there

may be only one vertex in Q ($=\widetilde{C}_2$), the neighbor of q in Q, with two neighbors out of its part. Following the definition of Q, all other vertices in Q may only have at most one vertex in $\widetilde{C}_1$ (their neighbors were only in C_1 and the vertex q). Obviously both parts of the partition are connected. Hence, we can use Lemma 4.9 to find a connected 2-community structure.

(iii) If $|Q| = \lceil \frac{n-1}{3} \rceil - 1$ or $|Q'| = \lceil \frac{n-1}{3} \rceil - 1$. The construction below is based on $|Q| = \lceil \frac{n-1}{3} \rceil - 1$, but it can be easily modified for $|Q'| = \lceil \frac{n-1}{3} \rceil - 1$.

If $|Q| = \lceil \frac{n-1}{3} \rceil - 1$ then

– in case $|N| > 1$: update $\widetilde{C}_1 := Q \cup \{q\}$, $\widetilde{C}_2 := V \backslash (Q \cup \{q\})$ (similarly in case of Q'). The subgraphs induced by $\widetilde{C}_1$ and $\widetilde{C}_2$ are obviously connected and $|\widetilde{C}_1| = \lceil \frac{n-1}{3} \rceil$. All vertices have at most one neighbor in the other part except q which has two neighbors in $\widetilde{C}_2$. Then applying Lemma 4.7, the partition $\{\widetilde{C}_1, \widetilde{C}_2\}$ is a connected 2-community structure.

– in case $|N| = 1$: update $\widetilde{C}_1 := Q'$, $\widetilde{C}_2 := V \backslash Q'$

Due to the sizes of the sets C_1 and Q ($|C_1| < \lceil \frac{n-1}{3} \rceil$ and $|Q| = \lceil \frac{n-1}{3} \rceil - 1$) obviously $|Q'| \geq n - 2\lceil \frac{n-1}{3} \rceil + 1$ which means $|Q'| \geq \lceil \frac{n-1}{3} \rceil$. But similarly also $|Q'| \leq n - \lceil \frac{n-1}{3} \rceil$, hence $\lceil \frac{n-1}{3} \rceil \leq |Q'| \leq n - \lceil \frac{n-1}{3} \rceil$.

Furthermore, there is at most one vertex from each part $\widetilde{C}_1$, $\widetilde{C}_2$ which may have two vertices out of its own part. Hence Lemma 4.9 can be used to find a connected 2-community structure.

(iv) If $|Q| < \lceil \frac{n-1}{3} \rceil - 1$ and $|Q'| < \lceil \frac{n-1}{3} \rceil - 1$ (then necessarily $|Z| > 0$).

- If $|Z| \geq n - \lceil \frac{n-1}{3} \rceil$, define $\widetilde{C}_1 := V \setminus Z$, $\widetilde{C}_2 := Z$ and remove all labels. If there exists a vertex in $\widetilde{C}_1$ which has two neighbors in Z then label it (it can be at most one such vertex), else label q. Then continue with the updates of the initial construction for $\{\widetilde{C}_1, \widetilde{C}_2\}$.

  Note. $|\widetilde{C}_1| \leq \lceil \frac{n-1}{3} \rceil$ and the size of $\widetilde{C}_1$ has strictly grown compared to C_1. Moreover, there is no vertex which has two neighbors out of its own part, except one vertex in C_1 which may have two neighbors in Z (in that case such a vertex is labeled). Besides, the partition is connected, so the step is correctly defined.

- If $\lceil \frac{n-1}{3} \rceil \leq |Z| < n - \lceil \frac{n-1}{3} \rceil$, define $\widetilde{C}_1 := V \setminus Z$, $\widetilde{C}_2 := Z$. If there is no vertex x in $\widetilde{C}_1$ such that $d_{out}(x) = 2$, then move any vertex $z \in \widetilde{C}_2$ such that $d_{out}(x) = 2$ to $\widetilde{C}_1$ until you can apply Lemma 4.7. So now we consider that there exist some vertex $x \in \widetilde{C}_1$ such that $d_{out}(x) = 2$ (notice that there are at most two vertices in $\widetilde{C}_2$ which may have two neighbors in $\widetilde{C}_1$).

  While there exist one vertex $z \in \widetilde{C}_2$ and one vertex $y \in \widetilde{C}_1$ such that $d_{out}(z) = d_{out}(y) = 2$ and $\{z, y\} \notin E$, update $\widetilde{C}_2 := \widetilde{C}_2 \cup \{y\} \setminus \{z\}$, $\widetilde{C}_1 := V \setminus \widetilde{C}_2$.

If there is at most one vertex $x \in \widetilde{C}_1$ such that $d_{out}(x) = 2$ and at most one vertex $z \in \widetilde{C}_2$ such that $d_{out}(z) = 2$, then apply Lemma 4.9 to obtain a connected 2-community structure.

Else if there exist $z, z' \in \widetilde{C}_2$ and $x \in \widetilde{C}_1$ such that $d_{out}(z) = d_{out}(z') = d_{out}(x) = 2$, $zx \in E$ and $z'x \in E$, then move x into $\widetilde{C}_2$. Notice that we still have $|\widetilde{C}_2| \leq n - \lceil \frac{n-1}{3} \rceil$ and now every vertex in $\widetilde{C}_2$ has at most one neighbor in $\widetilde{C}_1$ so we can move every vertex $z \in \widetilde{C}_1$ into $\widetilde{C}_2$ until we can use Lemma 4.7 and obtain a connected 2-community structure.

Else, every vertex in $\widetilde{C}_2$ has at most one neighbor in $\widetilde{C}_1$ so we can move every vertex $z \in \widetilde{C}_1$ into $\widetilde{C}_2$ until we can use Lemma 4.7 and obtain a connected 2-community structure.

- If $0 < |Z| < \lceil \frac{n-1}{3} \rceil$ then $|Z \cup Q \cup \{q\}| \leq 2\lceil \frac{n-1}{3} \rceil - 2 < n - \lceil \frac{n-1}{3} \rceil$.
  Notice that $\lceil \frac{n-1}{3} \rceil \leq |Z \cup Q \cup \{q\}|$ (indeed, $|Z \cup Q \cup \{q\}| < \lceil \frac{n-1}{3} \rceil$ is not possible since $|C_1| + |Z| + |Q| + |Q'| + 2 = n$). Then define $\widetilde{C}_1 := Z \cup Q \cup \{q\}$ and $\widetilde{C}_2 := V \setminus (Z \cup Q \cup \{q\})$. Now the only vertices which may have two neighbors out of their own part are one labeled vertex in the old C_1 and a neighbor of q' in Z. Hence we can apply Lemma 4.9 to obtain a connected 2-community structure.

Each case leads to a connected 2-community structure. □

This approach is very structural and efficient but is a little fastidious. It is possible to design a more general algorithm for graphs of maximum degree 3 which uses convergence properties to obtain a 2-community structure in polynomial time.

Similarly to vertices of degree 3, the restrictions on the size of partitions are discussed to ensure the vertices of degree 2 fulfil the proportion condition of a 2-community structure.

Lemma 4.11. *Let $G = (V, E)$ be a graph of maximum degree 3 of size n. Let $\{C_1, C_2\}$ be a partition of V such that $\lceil \frac{n-1}{3} \rceil \leq |C_1| \leq \lfloor \frac{n}{2} \rfloor$. Then each vertex of degree 2 in C_1 with at most one out-neighbor fulfils the proportion condition of a 2-community structure.*

If the partition is balanced, then each vertex of degree 2 in G with at most one out-neighbor fulfils the proportion condition.

Proof. Let $\{C_1, C_2\}$ be a partition of V such that $\lceil \frac{n-1}{3} \rceil \leq |C_1| \leq \lfloor \frac{n}{2} \rfloor$. Obviously, any vertex of degree 2 with no neighbors out of its own part fulfils the proportion condition. Moreover any vertex of degree 2 in C_1 with only one out-neighbor satisfies $\frac{1}{|C_1|-1} \geq \frac{1}{|C_2|}$ since $|C_1| \leq |C_2|$.

If the partition is balanced, then $\frac{1}{|C_1|-1} \geq \frac{1}{|C_2|}$ and $\frac{1}{|C_2|-1} \geq \frac{1}{|C_1|}$, and hence the vertices of degree 2 from both parts with exactly one out-neighbor satisfy the proportion condition. □

The lemmas 4.7 and 4.11 can be sum up into the following theorem which gives a good overview of the possibilities for each vertex belonging to a 2-community structure.

Theorem 4.12. *Let $G = (V, E)$ be a graph of maximum degree 3 of size n and $\{C_1, C_2\}$ be a partition of V such that $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$.*

If the partition has one of the properties (i)-(iii) where only specified vertices may have out-neighbors (the other ones have only in-neighbors), then $\{C_1, C_2\}$ is a 2-community structure on G:

(i) The vertices of degree 2 from the smaller part and all the vertices of degree 3 have at most one out-neighbor.

(ii) The vertices of degree 2 and 3 have at most one out-neighbor and the partition is balanced.

(iii) The vertices of degree 2 from the smaller part have at most one out-neighbor, the vertices of degree 3 in C_i, for some $i \in \{1, 2\}$, have at most two out-neighbors and $|C_i| = \lceil \frac{n-1}{3} \rceil$ (or also $|C_i| = \lceil \frac{n-1}{3} \rceil + 1$ if $n \equiv 1 \mod 3$) and the vertices of degree 3 in C_{3-i} have at most one out-neighbor.

Proof. In each case (i), (ii), or (iii), all the vertices of the graph G satisfy the proportion condition due to Lemmas 4.7 and 4.11. Thus, $\{C_1, C_2\}$ is a 2-community structure on G. □

The general algorithm which find a 2-community structure in polynomial time in graphs of maximum degree 3 consists in starting with an initial partition with the property that no vertex has all of its neighbors out of its part (see Lemma 4.13). Then, the algorithm consists in moving vertices from one part to the other until the remaining partition is a 2-community structure (see Theorem 4.14). Convergence properties ensure that the algorithm always finishes and output the solution in polynomial time.

Lemma 4.13. *Every connected graph of maximum degree 3 on at least 6 vertices has a partition $\{C_1, C_2\}$ such that no vertex in the graph has all of its neighbors out of its own part, $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$. Moreover, such a partition can be found in polynomial time.*

Proof. Let $G = (V, E)$ be a connected graph of maximum degree 3 on n vertices, $n \geq 6$.

Let $u, v \in V$ be such that $uv \in E$ and initially, put into C_1 the vertices u, v including its pendant vertices. If there is a vertex z of degree 2 with u and v as neighbors, update $C_1 := C_1 \cup \{z\}$.

Now we repeat the following steps (S1) and (S2) until $|C_1| \geq \lceil \frac{n-1}{3} \rceil$:

(S1) Let w be a neighbor of u (or v) which is not in C_1, put $C_1 := C_1 \cup \{w\}$, $C_2 := C_2 \setminus \{w\}$, and relabel $u := w$ (or $v := w$). If there exist vertices in C_2 which have all of their neighbors in C_1, transfer them into C_1. Notice, that these must be neighbors of w, hence at most two additional vertices are transferred to C_1.

(S2) If there is no such vertex w then u and v have all its neighbors in C_1. In such a case let u be any vertex in C_1 with at least one out-neighbor in C_2 (such a vertex must always exist due to connectivity of G).

Notice that each time we apply (S1), by (S2) we ensure that there is no vertex in C_2 with only out-neighbors. The algorithm finishes with a set C_1, $\lceil \frac{n-1}{3} \rceil \leq |C_1| \leq \lceil \frac{n-1}{3} \rceil + 2$ where $|C_1| \leq n - \lceil \frac{n-1}{3} \rceil$, due to $n \geq 6$.

Clearly, the algorithm runs in a polynomial time and results in a partition $\{C_1, C_2\}$, $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$ such that no vertex has all its neighbors out of its own part. □

Now we give the general algorithm which output a 2-community structure from a connected graph of maximum degree 3.

Theorem 4.14. *Let G be a connected graph of maximum degree 3 on n vertices, $n \geq 4$, except S_4. Then G has a 2-community structure which can be found in a polynomial time.*

Proof. Let $n = 4$. Since G is not isomorphic to S_4, any partition with an edge in each part has obviously a 2-community structure. If $n = 5$, then either G has a partition consisting of a cycle of length 3 and an edge (which has clearly a 2-community structure), or G must be isomorphic to one of the cases in Figure 4.7.

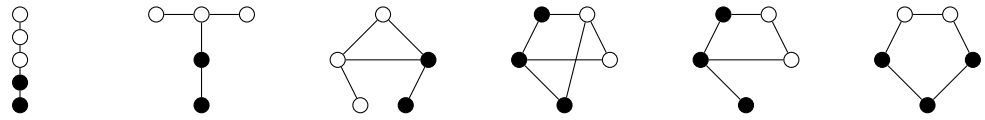


Figure 4.7: A 2-community structure $\{C_1, C_2\}$ (C_1 in white, C_2 in black) for other graphs on 5 vertices.

Let $n \geq 6$. Consider any partition $\{C_1, C_2\}$ of V, $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$, such that no vertex in G has all of its neighbors out of its part (such a partition can be found in polynomial time due to Lemma 4.13). In such partition there may exist vertices not satisfying the proportion condition which can be split into two categories:

(A) if there exists $i \in \{1, 2\}$, $|C_i| > \lceil \frac{n-1}{3} \rceil$ and $n \not\equiv 1 \mod 3$ or $|C_i| > \lceil \frac{n}{3} \rceil$ and $n \equiv 1 \mod 3$, the vertices of degree 3 in C_i with two out-neighbors,

(B) the vertices of degree 2 in the larger part with one out-neighbor, if the partition is not balanced.

Our algorithm transfers the vertices between C_1 and C_2 in several steps until all vertices satisfy the proportion condition. In each step of the algorithm the size of the cut between C_1 and C_2 decreases and we insure that no vertex from the partition has only out-neighbors.

The algorithm applies the improvement procedure (consisting of three stages) as many time as there is a vertex transferred between the parts. In the initial partition $\{C_1, C_2\}$, there are only 3 type of vertices that do not satisfy the proportion condition: vertices of degree 3 in C_1 with two out-neighbors (if $|C_1| > \lceil \frac{n-1}{3} \rceil$), vertices of degree 3 in C_2 with two out-neighbors (if (if $|C_2| > \lceil \frac{n-1}{3} \rceil$) and vertices of degree 2 with one out-neighbor, respectively handled by the three stages. Each stage consists in moving such vertices in the other part so they satisfy the proportion condition (See Figure 4.8 as an illustration), taking care of not compromising the proportion condition for any other vertex.

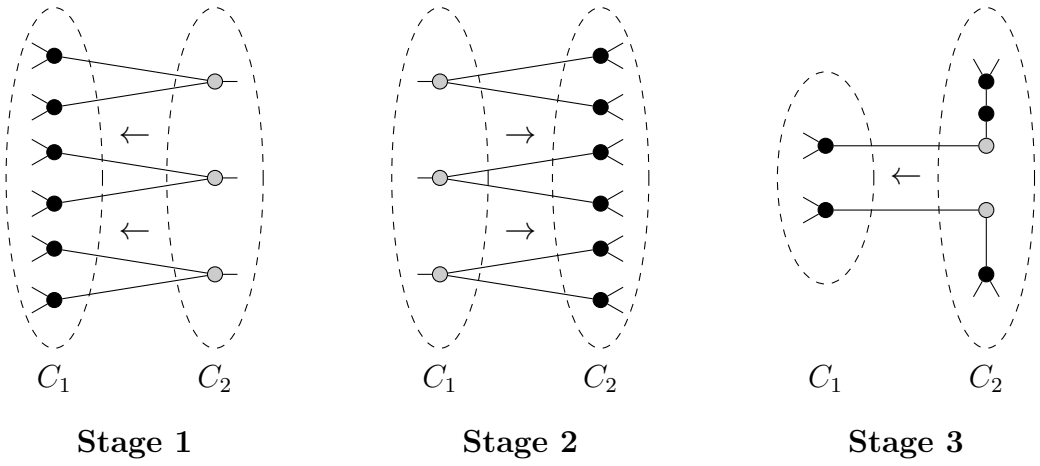


Figure 4.8: Sketch of the three stages of the algorithm to compute a 2-community structure in a graph of maximum degree 3. Candidates vertices to be moved are in gray.

Improvement Procedure: Stage 1

In this stage we primarily handle vertices of degree 3 in C_2 which have two out-neighbors by transferring them into C_1, keeping the size of C_1 under $n - \lceil \frac{n-1}{3} \rceil$ and the property that there is no vertex in the partition with only out-neighbors.

While $|C_1| < n - \lceil \frac{n-1}{3} \rceil$ and there is a vertex $u \in C_2$ which has two out-neighbors, repeat the following two steps:

(a) Update $C_1 := C_1 \cup \{u\}$, $C_2 := C_2 \setminus \{u\}$.

(b) If there exists a vertex v in C_2 with only out-neighbors, then update $C_1 := C_1 \cup \{v\}$ and $C_2 := C_2 \setminus \{v\}$.

We emphasize that at the end of each iteration of the while loop in Stage 1, no vertex in the partition has only out-neighbors. By assumption such vertex doesn't exist before entering the first iteration of the while loop. In each iteration of the while loop when (a) is executed, it may create at most one vertex with only out-neighbors (which must be

the neighbor of u in C_2), and (b) transfers it into C_1. In the part (b) a vertex with only out-neighbors cannot be created since the transferred vertex v has no neighbor in C_2.

Moreover, notice that each iteration of the while loop in Stage 1 decreases the size of the cut by at least one since both parts (a) and (b) decrease the size of the cut by at least one each time they are executed. After the last iteration of while loop necessarily $|C_1| \leq n - \lceil \frac{n-1}{3} \rceil + 1$ since each iteration of the while loop may increase the size of C_1 by at most 2.

If $|C_1| = n - \lceil \frac{n-1}{3} \rceil + 1$, then the last iteration of the while loop had to decrease the size of the cut by at least two by applying both steps (a) and (b). In such a case we make the following Size Update.

Size Update in case $|C_1| = n - \lceil \frac{n-1}{3} \rceil + 1$:

(a^*) Let $v \in C_1$ be a vertex with at least one neighbor in C_2, update $C_1 := C_1 \setminus \{v\}$, $C_2 := C_2 \cup \{v\}$.

(b^*) If there exists vertices in C_1 with only out-neighbors, then transfer all such vertices into C_2.

Since G is connected, such vertex v in part (a^*) always exists. Such transfer may create at most 2 vertices in C_1 with only out-neighbors (they must be neighbours of v in C_1), so we also transfer them into C_2 in part (b^*). After the Size Update, there is no vertex with only out-neighbours and $\lceil \frac{n-1}{3} \rceil \leq |C_1| \leq n - \lceil \frac{n-1}{3} \rceil$ since $n \geq 6$.

Notice that the Size Update may increase the size of the cut by at most one, but the last iteration of the while loop in Stage 1 decreases it by at least two, hence together the size of the cut decreases by at least one.

Improvement Procedure: Stage 2

In this stage we primarily handle vertices of degree 3 in C_1 which have two out-neighbors by transferring them into C_2, keeping the size of C_2 under $(n - \lceil \frac{n-1}{3} \rceil)$ and the property that there is no vertex in the partition with only out-neighbors.

Since Stage 2 is symmetrical to Stage 1, swap the roles of C_1 and C_2 by relabelling $C_1 := C_2$ and $C_2 := V \setminus C_1$ and apply Stage 1. Notice then all previous conclusions hold.

Improvement Procedure: Stage 3

If the partition is not balanced, the vertices of degree 2 with one out-neighbor must be transferred from the larger part to the smaller part, keeping the property that there is no vertex in the partition with only out-neighbors.

If $|C_1| > |C_2|$, relabel $C_1 := C_2$ and $C_2 := V \setminus C_1$
While $|C_1| < \lfloor \frac{n}{2} \rfloor$ and there exists a vertex u of degree 2 in C_2 with one neighbor in C_1:

(a) Update $C_1 := C_1 \cup \{u\}$, $C_2 := C_2 \setminus \{u\}$.

(b) If there exists a vertex v *in* C_2 *with only out-neighbors, then update* $C_2 := C_2 \cup \{v\}$ *and* $C_1 := C_1 \setminus \{v\}$.

In each iteration of the while loop in Stage 3 when (a) is executed, part (b) ensures that there is no vertex in the partition with only out-neighbors at the end of the iteration.

Each iteration of the while loop in Stage 3 doesn't increase the size of the cut. In the end of Stage 3 if the final partition doesn't have a 2-community structure then a vertex of the category (A) or (B) must exist in the partition. In the first case Stage 1 or 2 must be executed before entering Stage 3 again, hence the cut-set is decreased by at least 1. The second case is only possible if $\lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$ and two vertices were added in the last iteration of the while loop. It means part (b) must be executed and the cut-size is decreased by at least 2. Notice that Stage 3 may again create vertices of the category (A) even if they didn't exist before entering Stage 3.

It is easy to see that the algorithm always finishes. After each iteration of the while loop in Stage 1 (resp. Stage 2), the size of the cut decreases by at least one. In Stage 3 each iteration of the while loop increases the size of the smaller parts by at least one vertex and stops before/when the partition is balanced or the smaller part becomes the larger part, but in that case the cut-size is decreased by at least 2. The algorithm clearly runs in a polynomial time.

Let's discuss the correctness of the algorithm. It always finishes with the sizes $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$. The algorithm starts with the partition in which every vertex has at least one neighbor in its own part. The key feature of the algorithm is to keep this property valid during the all stages of the algorithm which was discussed separately for each stage. Hence all vertices of degree 1 in the final partition clearly satisfy the proportion condition.

- If the final partition is balanced then all vertices of degree 2 and 3 may have at most one out-neighbor (otherwise Improvement Procedure could be applied again), hence it has a 2-community structure due to Theorem 4.12(ii).
- If the final partition is not balanced, then the partition must have the properties described in Theorem 4.12(i) or (iii). Otherwise, one of Stages 1-3 could be applied again. Hence, the final partition has a 2-community structure.

□

The latter algorithm gives a 2-community structure in polynomial time without any restriction about its connectivity. It is interesting that this algorithm can be modified in order to insure the connectivity of the 2-community structure given in output. The key

difference from the previous algorithm is to ensure that the initial partition is connected (instead of just requiring that any vertex has at least one neighbor in its part). This can be found in polynomial time and allows us to design a new algorithm which ensure the connectedness of the 2-community structure.

Lemma 4.15. *Every connected graph of maximum degree* 3 *on* n *vertices,* $n \geq 4$*, has a connected partition* $\{C_1, C_2\}$ *such that* $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$*,* $i = 1, 2$*. Moreover, such a partition can be found in polynomial time.*

Proof. Let $G = (V, E)$ be a graph with the given properties.

Initially, put into C_1 any vertex v such that $G[V \setminus \{v\}]$ is connected. The existence of such vertex is ensured by the following procedure. Let x be any vertex of G. While $G[V \setminus \{x\}]$ is disconnected, label the vertex x and consider any other vertex x' in a connected component of $G[V \setminus \{x\}]$ that does not contains any labeled vertex, and let $x := x'$. Let v be the last considered vertex x. This procedure terminates in less than $|V|$ steps.

The algorithm keeps connectivity of $G[C_1]$ and $G[C_2]$ and extends C_1 either by transferring vertices from C_2 to C_1 or relabelling a suitable connected part of the graph until $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$.

The algorithm starts with the initial set C_1 and repeats the Update Procedure until $|C_1| \geq \lceil \frac{n-1}{3} \rceil$. In each run of the procedure only one of the options 1 or 2 is executed.

The Update procedure:
Let w be a vertex in C_2 which has a neighbor in C_1 (such a vertex must exist since G is connected).

Option 1: If the subgraph induced by $C_2 \setminus \{w\}$ is connected, put:

$$C_1 := C_1 \cup \{w\}, C_2 := C_2 \setminus \{w\}.$$

Option 2: If the subgraph induced by $C_2 \setminus \{w\}$ is disconnected (w must be of degree 3), then denote by A, B the vertex-sets of two connected induced subgraphs of G on $C_2 \setminus \{w\}$. Depending on the size of A, the following update is executed.

- If $|A| \leq n - 2\lceil \frac{n-1}{3} \rceil$, put:

  $$C_1 := C_1 \cup A \cup \{w\}, C_2 := B.$$

  Notice that $|C_1| \leq n - \lceil \frac{n-1}{3} \rceil$, $\{C_1, C_2\}$ is a connected partition and the size of C_1 strictly increased.

- If $n - 2\lceil \frac{n-1}{3} \rceil + 1 \leq |A| \leq n - \lceil \frac{n-1}{3} \rceil$, then notice that $|A| \geq \lceil \frac{n-1}{3} \rceil$ and put:

  $$C_1 := A,\ C_2 := V \setminus A.$$

  Obviously, $\{C_1, C_2\}$ is a connected partition with $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$, hence the Update Procedure halts.

- If $|A| > n - \lceil \frac{n-1}{3} \rceil$, put:

$$C_1 := C_1 \cup B \cup \{w\},\ C_2 := A.$$

  Notice that $|C_1| < \lceil \frac{n-1}{3} \rceil$, $\{C_1, C_2\}$ is a connected partition and the size of C_1 strictly increased.

If $|C_1| \geq \lceil \frac{n-1}{3} \rceil$ after the execution of the option 1 or 2, then the UPDATE PROCEDURE halts, otherwise the UPDATE PROCEDURE is repeated again.

By our construction, the partition $\{C_1, C_2\}$ remains connected during each run of the UPDATE PROCEDURE. Each time the UPDATE PROCEDURE is executed, the size of C_1 strictly increases, hence the algorithm always terminates. At the end of the algorithm $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$ and the algorithm clearly runs in a polynomial time. □

Theorem 4.16. *Every connected graph of maximum degree* 3 *with at least* 4 *vertices (except a star) has a connected* 2*-community structure which can be found in polynomial time.*

Proof. Let $G = (V, E)$ be a connected graph of maximum degree 3 on n vertices, $n \geq 4$, not isomorphic to a star. Due to Lemma 4.15, a connected partition $\{C_1, C_2\}$ of V such that $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$, can be found in polynomial time. Let $\{C_1, C_2\}$ be such a partition and notice that the vertices that do not satisfy the proportion condition can be split into two categories:

(A) if there exists $i \in \{1, 2\}$ such that $|C_i| > \lceil \frac{n-1}{3} \rceil$ in case $n \not\equiv 1 \mod 3$ or $|C_i| > \lceil \frac{n-1}{3} \rceil + 1$ in case $n \equiv 1 \mod 3$, then all the vertices of degree 3 in C_i with two out-neighbors,

(B) if the partition is not balanced, then all the vertices of degree 2 in the larger part with one out-neighbor.

The algorithm starts with the initial partition $\{C_1, C_2\}$ and then the IMPROVEMENT PROCEDURE (consisting in three stages) can be applied several times. The procedure transfers step-by-step all the vertices of degree at least 2 (with exactly one neighbor in its own part) between C_1 and C_2 or relabel the sets, until all the vertices satisfy the proportion condition. Since the initial partition is connected, transferring vertices with such a property never disconnects any part of the partition.

The IMPROVEMENT PROCEDURE: STAGE 1 (Category (A) vertices)
In this stage we handle vertices in C_2 of degree 3 with two out-neighbors by transferring them into C_1, keeping the size of C_1 smaller than $n - \lceil \frac{n-1}{3} \rceil$ and ensuring connectivity of the partition $\{C_1, C_2\}$.

While $|C_1| < n - \lceil \frac{n-1}{3} \rceil$ *and there is a vertex* $u \in C_2$ *with two out-neighbors, update:*

$$C_1 := C_1 \cup \{u\}, \quad C_2 := C_2 \setminus \{u\}.$$

Notice that each iteration of Stage 1 decreases the size of the cut by at least one.

The Improvement Procedure: Stage 2 (Category (A) vertices)
Similarly to Stage 1, in Stage 2 we handle vertices in C_1 of degree 3 with two out-neighbors by transferring them into C_2, keeping the size of C_2 smaller than $n - \lceil \frac{n-1}{3} \rceil$ and ensuring connectivity of the partition $\{C_1, C_2\}$.

While $|C_2| < n - \lceil \frac{n-1}{3} \rceil$ *and there is a vertex* $u \in C_1$ *with two out-neighbors, update:*

$$C_2 := C_2 \cup \{u\}, \quad C_1 := C_1 \setminus \{u\}.$$

Notice that each iteration of Stage 2 decreases the cut-size by at least one.

The Improvement Procedure: Stage 3 (Category (B) vertices)
If the partition is not balanced, the vertices of degree 2 with one out-neighbor must be transferred from the larger part to the smaller part.

If $|C_1| > |C_2|$, relabel $C_1 := C_2$ and $C_2 := V \setminus C_1$.

While $|C_1| < \lfloor \frac{n}{2} \rfloor$ *and there exists a vertex* u *of degree* 2 *in* C_2 *with one neighbor in* C_1*, update:*

$$C_1 := C_1 \cup \{u\}, \quad C_2 := C_2 \setminus \{u\}.$$

Each iteration of the while loop in Stage 3 doesn't increase the size of the cut. In the end of Stage 3 if the final partition doesn't have a 2-community structure then a vertex of the category (A) must exist in the partition. In that case, Stage 1 or 2 must be executed before entering Stage 3 again, hence the cut-size is decreased by at least one. Notice that Stage 3 may again create vertices of the category (A) even if they didn't exist before entering Stage 3.

It is easy to see that the algorithm always terminates. Each iteration of the while loop in Stage 1 (resp. Stage 2) decreases the cut-size by at least one. In Stage 3 each iteration of the while loop increases the size of the smaller part by at least one and halts before or when the partition is balanced. Following the construction, if the Improvement Procedure needs to be run again, it must first run through Stage 1 or 2 which decreases the cut-size by at least one. Moreover, the algorithm clearly runs in polynomial time.

Let's discuss the correctness of the algorithm. Suppose the algorithm terminates with the final partition $\{C_1, C_2\}$. Due to the conditions inside the algorithm, $\lceil \frac{n-1}{3} \rceil \leq |C_i| \leq n - \lceil \frac{n-1}{3} \rceil$, $i = 1, 2$. Initially, the partition is connected and remains so after each stage, hence the final partition is connected too. Moreover, each vertex of degree 1 necessarily satisfies the proportion condition since it must be in the same part as its neighbor. Now there are two options:

- If the final partition is balanced then all vertices of degree 2 and 3 may have at most one out-neighbor (otherwise the IMPROVEMENT PROCEDURE could be applied again), hence the final partition $\{C_1, C_2\}$ is a 2-community structure due to Lemma 4.12(ii).
- If the final partition is not balanced, then the partition must have the properties described in Lemma 4.12(i) or (iii) (otherwise, one of Stages 1-2 could be applied again). Hence the final partition $\{C_1, C_2\}$ is a 2-community structure.

□

4.3.3 Dense graphs

Now we investigate the problem of the existence and finding of a connected 2-community structure in dense graphs. We prove that any graph $G = (V, E)$ of minimum degree $|V| - 3$ has a connected 2-community structure which can be found in polynomial time.

Lemma 4.17. *If the complement of the graph G is 2-colorable (using each color for at least 2 vertices), then G has a connected 2-community structure which can be found in polynomial time.*

Proof. Let $G = (V, E)$ be a graph such that its complement $\overline{G}$ is 2-colorable. Fix a 2-coloring of $\overline{G}$ (with at least 2 vertices for each color) and define $\{C_1, C_2\}$ as a partition of V, where each part corresponds to one color in $\overline{G}$. Obviously, $|C_1|, |C_2| \geq 2$. Notice that the induced subgraph on the vertex set C_1 (resp. C_2) is a clique. Therefore, any vertex $v \in V$ satisfies the proportion condition and the partition $\{C_1, C_2\}$ is a 2-community structure. Since a 2-coloring can be found in polynomial time, the 2-community structure $\{C_1, C_2\}$ too. Obviously, the partition is connected. □

This result directly implies the following theorem:

Theorem 4.18. *The complement of any bipartite graph (with at least two vertices in each part) has a connected 2-community structure which can be found in polynomial time.*

Lemma 4.17 can also be used for graphs of minimum degree $(n - 3)$:

Theorem 4.19. *Any graph (except a star) of minimum degree $(n - 3)$, $n \geq 4$, where n is the number of vertices of the graph, has a connected 2-community structure which can be found in polynomial time.*

Proof. Let G be a graph of size n and of minimum degree $(n-3)$ (except a star), $n \geq 4$, and $\overline{G}$ be the complement of G. Notice that $\overline{G}$ is of degree at most 2. If $\overline{G}$ does not contain an odd cycle, then there exists a 2-coloring of $\overline{G}$ with at least 2 vertices for each color. In such case, a connected 2-community structure can be found in polynomial time due to Lemma 4.17.

Now let A be the union of all vertices belonging to an odd cycle in $\overline{G}$ and denote by $B := V \setminus A$. $\overline{G}[A]$ is the union of p odd induced cycles with the vertex sets $O_1, \ldots, O_p$, $p \geq 1$. For each i, $1 \leq i \leq p$, let v_i be any vertex of O_i and fix a 2-coloring of $\overline{G}[O_i \setminus \{v_i\}]$. Let $O_{i,1}$, $O_{i,2}$ be the set of vertices corresponding to each color, obviously $|O_{i,1}| = |O_{i,2}|$. If $|B| \geq 2$, take a 2-coloring of B and define a partition $\{B_1, B_2\}$ of B (each part corresponding to a color) such that $|B_1| \geq |B_2| \geq 1$, otherwise $B_1 := B$, $B_2 := \emptyset$. Define

$$C_1 := \cup_{i=1}^{p}(O_{i,1} \cup \{v_i\}) \cup B_1, \quad C_2 := \cup_{i=1}^{p} O_{i,2} \cup B_2.$$

Observe that $|C_1|, |C_2| \geq 2$ ($|C_2| \leq 1$ is only possible for a star or a graph with 3 vertices). Obviously, every such 2-colouring can be found in polynomial time. Finally we show that the partition $\{C_1, C_2\}$ is a connected 2-community structure.

All vertices of C_2 satisfy the proportion condition in G since $G[C_2]$ is a clique. For each i, $1 \leq i \leq p$, all neighbors of v_i in $G[C_1]$ satisfy the proportion condition in G since they have all vertices of C_1 as neighbors. Moreover, the non-neighbor of v_i in $G[C_1]$ and v_i itself satisfy the proportion condition in G since $|C_1| > |C_2|$ implies that $\frac{|C_1|-2}{|C_1|-1} \geq \frac{|C_2|-1}{|C_2|}$.

Observe that the partition $\{C_1, C_2\}$ is connected. Obviously, $G[C_2]$ is connected since $G[C_2]$ is a clique. Moreover, any two vertices in C_1 are neighbors except v_i and its neighbour in $\overline{G}[O_{i,1}]$ for all i, $1 \leq i \leq p$. If $B_1 \neq \emptyset$, such two vertices must have a common neighbor in B_1. If $B_1 = \emptyset$, then either $|O_{1,1}| \geq 3$ or $p \geq 2$ (due to assumptions on G), and such two vertices have a common neighbor either in $O_{1,1}$ or $O_{j,1}$, $j \neq i$. Hence, $G[C_1]$ is also connected. □

Theorem 4.20. *Let $G = (V, E)$ be a graph with minimum degree $\lceil \frac{(c-1).|V|}{c} \rceil$ where c is the size of an inclusion-wise maximal clique in G, i.e. such a clique is not a subgraph of another clique. Then, G has a connected 2-community structure which can be found in polynomial time.*

Proof. If $c \geq |V| - 1$, then for any vertex $u \in V$, $d(u) \geq \lceil \frac{(|V|-2).|V|}{|V|-1} \rceil \geq |V| - 3$ and the rest follows from Theorem 4.19.

If $c \leq |V| - 2$, let C be the inclusion-wise maximal clique in G and take $\{C, V \setminus C\}$ as a partition. Obviously, the size of both parts is at least 2. C is a clique, hence the proportion condition is trivially satisfied for all vertices in C. If a vertex $u \in V \setminus C$ has a neighbor in C, then:

$$\frac{d_{in}(u)}{|V| - c - 1} \geq \frac{\frac{(c-1).|V|}{c} - (c-1)}{|V| - c - 1} \geq \frac{c-1}{c} \geq \frac{d_{out}(u)}{c},$$

hence the proportion condition is satisfied for all vertices $u \in V \setminus C$ with a neighbor in C. The rest of vertices in $V \setminus C$ trivially satisfy the proportion condition since they do not have a neighbor in C.

Now we prove that the partition $\{C, V \setminus C\}$ is connected, which is obviously true for $G[C]$. Let suppose that $G[V \setminus C]$ be disconnected and A be the smallest connected component of $G[V \setminus C]$. Notice that $|A| \leq \frac{|V|-c}{2}$ and let $u \in A$. Then $\frac{(c-1)\cdot|V|}{c} \leq d(u) \leq \frac{|V|-c}{2} + c - 2$ and hence $|V| \leq \frac{c(c-4)}{c-2} < c$, which is impossible. Therefore, $G[V \setminus C]$ is a connected subgraph. □

4.4 Balanced 2-community structures

In this section we study complexity of the problems related to balanced 2-community structures. First, we discuss the hardness of the problem in general graphs. We prove that the Balanced Weak 2-community and Balanced 2-community problems are NP-complete. The latter result is contained as the main result in [58], an alternative shorter proof is presented in this section. Both NP-completeness results are extended to a connected balanced 2-community structure. Then, we investigate the problem in graphs of low edge density. We prove that every graph of maximum degree 3 has a balanced weak 2-community structure that can be found in polynomial time. The structural properties of low-degree graphs are crucial to obtain such a result. Finally, we prove that the problem is polynomial-time solvable for graphs with bounded treewidth.

4.4.1 General graphs

We focus on the problem of Balanced 2-community in general graphs. In [40] it has been proved that finding a connected balanced partition without any additional constraint is an NP-complete problem in general graphs. We prove similar results for Balanced Weak 2-community and Balanced 2-community and their connected variants. To show that Balanced Weak 2-community is NP-complete, we use a reduction from the Balanced Co-Satisfactory Partition problem, proved to be NP-complete in [21].

The problems is defined as follow:

Balanced Co-Satisfactory Partition
Input : A graph $G = (V, E)$ on an even number of vertices.
Question : Is there a balanced partition $\{C_1, C_2\}$ of V such that for every $v \in V$, $d_{in}(v) \leq d_{out}(v)$?

Theorem 4.21. Balanced Weak 2-community *is NP-complete.*

Proof. The problem is clearly in NP. In the following we define a polynomial-time reduction from Balanced Co-Satisfactory Partition to Balanced Weak 2-community. Let G be a graph on an even number n of vertices as an instance of Balanced Co-Satisfactory Partition, and let $\overline{G}$, the complement of G, be an instance of Balanced Weak 2-community. If G admits a balanced co-satisfactory partition $\{C_1, C_2\}$ then $\{C_1, C_2\}$ is also a weak 2-community. Suppose $d_{in}(v) \leq d_{out}(v)$ for every vertex $v \in V$ (in

the graph G). Let $\bar{d}_{in}(v)$ (resp. $\bar{d}_{out}(v)$) be the number of in-neighbors (resp. out-neighbors) of v in $\overline{G}$. Then, the following holds $\bar{d}_{in}(v)+1 = \frac{n}{2} - d_{in}(v) \geq \frac{n}{2} - d_{out}(v) = \bar{d}_{out}(v)$, which is the weak proportion condition for a balanced partition. Conversely, any balanced weak 2-community in $\overline{G}$ is a balanced co-satisfactory partition in G. □

The proof of the NP-completeness of Balanced Co-Satisfactory Partition in [21] is based on the graphs $G = (V, E)$, where $V = F \cup T \cup V_0$ with some additional properties: F and T are independent sets, there are no edges between T and V_0, and there is a vertex $f \in F$ that is not adjacent to any vertex of V_0. Any balanced co-satisfactory partition $\{C_1, C_2\}$ of V must have the following structure: $C_1 = F \cup S$ and $C_2 = T \cup (V_0 \setminus S)$ where $S \subseteq V_0$. If $\overline{G}$ is an instance of Balanced Weak 2-community (constructed following the proof of Theorem 4.21), one can see that C_1 is connected since f is adjacent to all vertices in $F \cup S$ and C_2 is connected since T is a clique and every vertex of T is adjacent to every vertex of $V_0 \setminus S$. Hence we can conclude that even the connected version of Balanced Weak 2-community is NP-complete:

Theorem 4.22. Connected Balanced Weak 2-community *is NP-complete.*

Estivill-Castro *et al.* [58] have shown that Balanced 2-community is NP-complete by constructing a reduction from a variant of the Clique problem. We propose a shorter alternative proof which is also valid for the Connected Balanced 2-community problem. The proof is based on the NP-complete problem Balanced Satisfactory Partition which was introduced by Bazgan *et al.* [20] as follows:

Balanced Satisfactory Partition
Input : A graph $G = (V, E)$ on an even number of vertices.
Question : Is there a balanced partition $\{C_1, C_2\}$ of V such that for every $v \in V$, $d_{in}(v) \geq \frac{d(v)}{2}$?

It can be proved that these two problems are in fact equivalent when the number of vertices is even.

Lemma 4.23. *Let $G = (V, E)$ be a graph with n vertices. Consider a partition $\{C_1, C_2\}$ of V and $v \in C_1$. Then the following assertions are equivalent:*

1. $\frac{d_{in}(v)}{|C_1|-1} \geq \frac{d(v)}{n-1}$
2. $\frac{d_{out}(v)}{|C_2|} \leq \frac{d(v)}{n-1}$
3. $\frac{d_{in}(v)}{|C_1|-1} \geq \frac{d_{out}(v)}{|C_2|}$

Proof. (1) $\Leftrightarrow$ (2) : $\frac{d_{in}(v)}{d(v)} \geq \frac{|C_1|-1}{n-1} \Leftrightarrow 1 - \frac{d_{out}(v)}{d(v)} \geq \frac{n-|C_2|-1}{n-1} \Leftrightarrow 1 - \frac{n-|C_2|-1}{n-1} \geq \frac{d_{out}(v)}{d(v)} \Leftrightarrow \frac{d_{out}(v)}{d(v)} \leq \frac{|C_2|}{n-1}$

(3) $\Leftrightarrow$ (1) : $\frac{d_{in}(v)}{|C_1|-1} \geq \frac{d_{out}(v)}{|C_2|} \Leftrightarrow \frac{d_{in}(v)}{|C_1|-1} \geq \frac{d(v)-d_{in}(v)}{n-|C_1|} \Leftrightarrow d_{in}(v)[\frac{1}{|C_1|-1} + \frac{1}{n-|C_1|}] \geq \frac{d(v)}{n-|C_1|} \Leftrightarrow \frac{d_{in}(v)}{d(v)} \geq \frac{|C_1|-1}{n-1}$ □

Remark 4.24. *Notice that the third assertion in Lemma 4.23 is the proportion condition of a 2-community structure.*

Lemma 4.25. *Let $G = (V, E)$ be a graph with an even number n of vertices and $\{C_1, C_2\}$ be a balanced partition of V. Then for any vertex $v \in V$, $d_{in}(v) = \frac{n/2-1}{n-1}d(v)$ if and only if $d(v) = n - 1$.*

Proof. If $d(v) = n - 1$, then clearly $d_{in}(v) = \frac{n}{2} - 1$. Suppose now that $d_{in}(v) = \frac{n/2-1}{n-1}d(v)$. Notice that $(-2)(\frac{n}{2} - 1) + 1(n - 1) = 1$ from which it can be easily shown that $\frac{n}{2} - 1$ and $n - 1$ do not have common divisors. This implies that $d(v)$ is a multiple of $n - 1$. Thus, $d(v) = n - 1$. □

Remark 4.26. *Let $\{C_1, C_2\}$ be a balanced partition of G and $v \in C_1$ be a vertex of degree $n-1$. Since v has $\frac{n}{2}-1$ neighbors in its own part and $\frac{n}{2}$ in other part, v does not satisfy the condition of* Balanced Satisfactory Partition. *However, v satisfies the* Balanced 2-Community *condition since $\frac{d_{in}(v)}{|C_1|-1} = 1$.*

Proposition 4.27. *For any graph with n vertices and maximum degree $(n-2)$ the problems* Balanced Satisfactory Partition *and* Balanced 2-Community *are equivalent.*

Proof. Suppose that $G = (V, E)$ is a yes-instance of Balanced Satisfactory Partition. Hence there exists a balanced partition $\{C_1, C_2\}$ of V such that any vertex $v \in V$ satisfies the condition $d_{in}(v) \geq \frac{1}{2}d(v)$, which implies that $d_{in}(v) \geq \frac{|C_1|-1}{2|C_1|-1}d(v) = \frac{|C_1|-1}{n-1}d(v)$. Thus, G is a yes-instance of Balanced 2-Community.

Suppose now that G is a yes-instance of Balanced 2-Community. Hence there exists a balanced partition $\{C_1, C_2\}$ of V such that any vertex $v \in V$ satisfies the condition $d_{in}(v) \geq \frac{|C_1|-1}{|C_2|}d_{out}(v)$ that is equivalent to $d_{in}(v) \geq \frac{|C_1|-1}{n-1}d(v)$ using Lemma 4.23. According to Lemma 4.25, there is no vertex v such that $d_{in}(v) = \frac{|C_1|-1}{n-1}d(v)$.

Now we need to show that for every vertex $v \in V$, $d_{in}(v) \geq \frac{1}{2}d(v)$. Suppose by contradiction that there exists a vertex $v \in V$ that does not satisfy the inequality that is:

$$\frac{|C_1| - 1}{n - 1}d(v) < d_{in}(v) < \frac{1}{2}d(v)$$

First, notice that $\frac{1}{2}d(v) - \frac{|C_1|-1}{n-1}d(v) = \frac{1}{2(n-1)}d(v) < 1$, which means that there is at most one integer number between $\frac{|C_1|-1}{n-1}d(v)$ and $\frac{1}{2}d(v)$.

Moreover, $d(v)$ cannot be even, since otherwise $\frac{d(v)}{2}$ would be a whole number and thus $d_{in}(v)$ could not be an integer number. Then $d(v)$ is odd and let $d(v) = 2p + 1$ for some integer p. We arrive to a contradiction by showing that $p < d_{in}(v) < p + \frac{1}{2}$. Notice that $d(v) < n - 1 \Rightarrow \frac{d(v)-1}{2} < \frac{|C_1|-1}{n-1}d(v)$ that implies $p < \frac{|C_1|-1}{n-1}d(v) < d_{in}(v)$. Then necessarily $d_{in}(v) \geq \frac{1}{2}d(v)$ for every vertex $v \in V$, that is G is a yes-instance of Balanced Satisfactory Partition. □

Balanced Satisfactory Partition has already been proved NP-complete in [20], even if both parts are required to be connected. Moreover, the reduction used in [20] does not construct a graph with vertices of degree $n-1$.

Thus we obtain a similar result as in [58] (the authors have mentioned in the proof that used technique works also in a connected case).

Theorem 4.28. Connected Balanced 2-Community *is NP-complete.*

Finally, it is interesting to notice that there exist graphs in which every 2-community structure is balanced (see Figure 4.9).

Figure 4.9: An example of a graph in which all 2-community structures are balanced

4.4.2 Balanced 2-community structures in graphs with low density

Remark 4.29. *Due to Theorem 4.16, every graph of maximum degree 3 has a 2-community structure, but it is not true for a balanced 2-community structure, see Figure 4.10. The graph is obtained by linking three 'cross gadgets'. First notice that if a balanced 2-community exists for the graph, then all vertices of each cross gadget must be in the same part. Indeed, each vertex of such community structure must have two neighbors in its own part. But on the other hand, this graph is impossible to split into two balanced parts without splitting a cross gadget.*

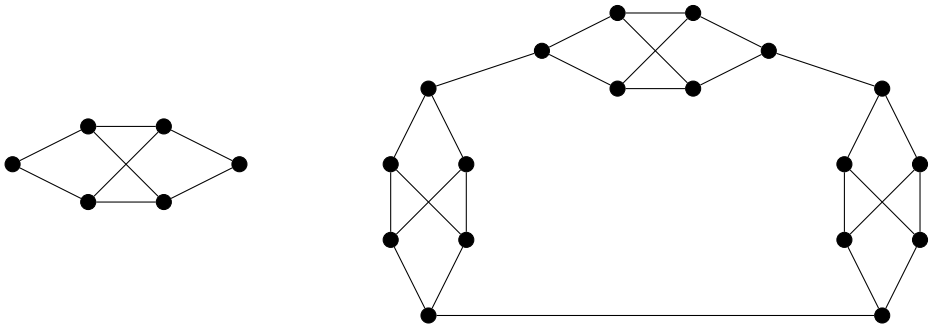


Figure 4.10: A cross gadget and a graph of maximum degree 3 without balanced 2-community structure.

Nevertheless, if we focus on a weak community, a balanced weak 2-community always exists in graphs of maximum degree 3, as it is shown in the following theorem.

Theorem 4.30. *Any graph of maximum degree* 3 *with at least* 4 *vertices has a balanced weak* 2*-community structure. Moreover, such a community structure can be found in polynomial time.*

Proof. Let $G = (V, E)$ be a connected graph of maximum degree 3. First notice that in any balanced partition of V, each vertex of degree 1 fulfils the weak proportion condition (even if its neighbor is not in its own part), and each vertex of degree 2 or 3, which has at least one neighbor in its own part, satisfies the weak proportion condition.

Therefore, the only vertices which may not satisfy the weak proportion condition are vertices of degree 2 or 3 which have no neighbor in their own part.

Consider any balanced partition $\{C_1, C_2\}$ of G and repeat the following steps (S1)-(S2) until it is possible:

(S1) If both parts contain a vertex of degree 2 or 3 that has no neighbor in its own part (say $v_1 \in C_1$, $v_2 \in C_2$), then update: $C_1 := C_1 \cup \{v_2\}\backslash\{v_1\}, C_2 := C_2 \cup \{v_1\}\backslash\{v_2\}$.

(S2) If there is only one partition that contains a vertex v of degree 2 or 3 that has no neighbor in its own part (without loss of generality suppose $v \in C_1$), then choose a vertex $w \in C_2$ such that w has at least one neighbor in C_1 and update: $C_1 := C_1 \cup \{w\}\backslash\{v\}, C_2 := C_2 \cup \{v\}\backslash\{w\}$.

First notice that if case (S2) occurs, such a vertex w always exists since the graph is connected. Moreover, the partition remains balanced after each step (S1) or (S2). Besides, the cut size between the partitions C_1 and C_2 always decreases (by at least 2 in case (S1), by at least 1 in case (S2)) so after a finite number of iterations (bounded trivially by $O(|V|^2)$, every vertex of degree 2 or 3 has at least one neighbor in its own part. Hence, the algorithm returns a balanced weak 2-community structure. □

Remark 4.31. *Notice that Theorem 4.30 cannot be extended to a connected case. There exist graphs of maximum degree* 3 *in which every balanced weak 2-community structures is disconnected, see Figure 4.11 as an example.*

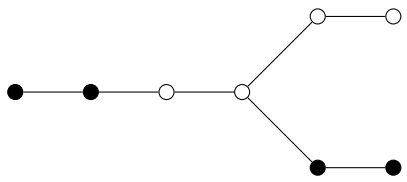


Figure 4.11: A tree of maximum degree 3 in which any balanced 2-community structure (or even balanced weak 2-community structure) is disconnected (an example of a balanced 2-community structure is presented by the black and white colors)

.

Theorem 4.32. BALANCED 2-COMMUNITY *(hence also* BALANCED WEAK 2-COMMUNITY*) is polynomially solvable for graphs with bounded treewidth.*

Proof. Such result follows from [19] where the t-DECOMPOSITION problem closely related to communities was studied. The input to the t-DECOMPOSITION problem is a graph $G = (V, E)$, an integer-valued function $t = t(n)$ such that $0 \leq t(n) \leq n$ for every $n \in \mathbb{N}$, and two functions $a, b : V \to \mathbb{N}$ such that $a(v), b(v) \leq d(v)$, for all $v \in V$. The problem consists of deciding if there is a partition $\{V_1, V_2\}$ of V with $|V_1| = t(|V|)$ such that $d_{G[V_1]}(v) \geq a(v)$ for every $v \in V_1$ and $d_{G[V_2]}(v) \geq b(v)$ for every $v \in V_2$.

In order for $\{V_1, V_2\}$ to be a balanced 2-community structure with $|V_1| \geq |V_2|$, every $v \in V_1$ must satisfy the condition $\frac{d_{G[V_1]}(v)}{\lceil n/2 \rceil - 1} \geq \frac{d(v) - d_{G[V_1]}(v)}{\lfloor n/2 \rfloor}$ and analogously for every $v \in V_2$ must hold $\frac{d_{G[V_2]}(v)}{\lfloor n/2 \rfloor - 1} \geq \frac{d(v) - d_{G[V_2]}(v)}{\lceil n/2 \rceil}$. Thus, BALANCED 2-COMMUNITY can be condidered as the t-DECOMPOSITION problem for selected values of the functions t, a, b. The conditions for BALANCED 2-COMMUNITY can be transformed to the conditions of the t-DECOMPOSITION problem where $t(n) = \lceil \frac{n}{2} \rceil$, $a(v) = b(v) = \lceil \frac{n/2-1}{n-1} d(v) \rceil$ for n even and $a(v) = \lceil d(v)/2 \rceil$, $b(v) = \lceil \frac{(n-1)/2-1}{n-1} d(v) \rceil$ for n odd. Since the t-DECOMPOSITION problem was proved to be polynomial-time solvable for bounded treewidth in [19], we can conclude the same result for the BALANCED 2-COMMUNITY problem. □

Notice that the result cannot be extended to a connected case for all graphs, see a tree on Figure 4.11 as a counterexample.

4.5 About graphs without 2-community structures

We know that any graph that is a star does not contain any community structure. We investigate if there are other graphs without any 2-community structure. By enumerating, using the computer, all graphs with n vertices starting with $n = 4$, we found that the minimum integer n for which there are graphs (not isomorphic to stars) that do not contain any 2-community structures is $n = 10$. The planar graph showed in Figure 4.12 has 10 vertices and has no 2-community structure.

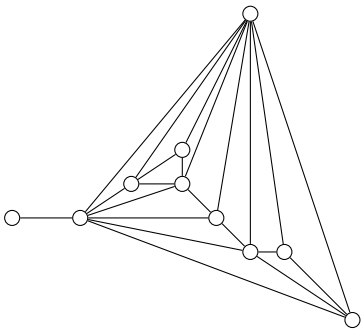


Figure 4.12: A graph with 10 vertices that does not contain any 2-community structure

We were able to generalize those counter-example in a infinite class of graphs in which there is no 2-community structure.

Definition 4.33. *Let $\mathcal{G}$ be a class of the graphs such that for $G = (V, E) \in \mathcal{G}$:*

- *$V = W_1 \cup W_2 \cup \{w, x, y, z\}$, where W_1, W_2 are cliques of the same size k, $k \geq 3$, and $\{x, y, w\}$ is a clique of size 3;*
- *w is adjacent to all vertices in $W_1 \cup W_2$,*
- *z is a pendant vertex adjacent to the vertex y,*
- *$1 \leq d_{W_1}(x) = d_{W_2}(x) \leq k - 1$ and $2 \leq d_{W_1}(y) = d_{W_2}(y) \leq k - 1$,*
- *$|W_i \cap (N(x) \cup N(y))| > \frac{3k}{k+3}$ for each $i \in \{1, 2\}$, and furthermore there exist vertices $\alpha, \beta \in W_1 \cup W_2$ such that $\alpha \in N(y) \setminus N(x)$, and $\beta \in N(x) \cap N(y)$,*
- *there is no edge between the vertex sets W_1 and W_2.*

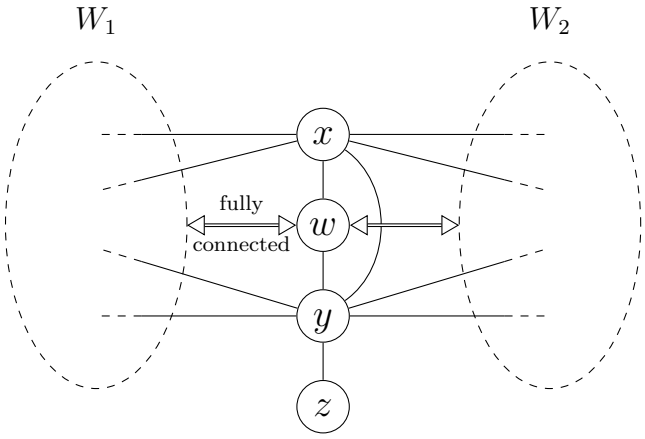


Figure 4.13: A schematic representation of a graph in $\mathcal{G}$.

Theorem 4.34. *Let $G = (V, E) \in \mathcal{G}$. Then G does not have a 2-community structure.*

Proof. Firstly, notice that there is no 2-community structure $\{A, B\}$ in G such that $|A| = 1$ or $|B| = 1$. Wlog, suppose by contradiction that $A = \{v\}$ for some vertex v, and notice that the neighbor of v in B must be a universal vertex in order to be satisfied. Since G does not contain a universal vertex, there is no 2-community structure $\{A, B\}$ in G with $|A| = 1$ or $|B| = 1$. Hence, assume that $|A|$, $|B| \geq 2$.

Observe that the vertex z is satisfied if and only if it belongs to the same community as the vertex y, hence without loss of generality we assume that $y, z \in B$. In addition, the vertex w has degree $|V| - 2$ and is not connected to $z \in B$. Hence, necessarily $w \in A$.

Now we prove that for any partition $\{A, B\}$ of V, where $w \in A$ and $y, z \in B$, there is at least one vertex which is not satisfied, hence there is no 2-community structure in G.

For any partition $\{A, B\}$ of V, we denote by A_i and B_i the sets $A \cap W_i$ and $B \cap W_i$, respectively, for $i \in \{1, 2\}$. In the first case, we suppose that B_1 or B_2 is empty. In the second case, we assume that B_1 and B_2 are not empty.

Case 1: $B_1 = \emptyset$ or $B_2 = \emptyset$

Suppose first that $B_1 = \emptyset$ and $B \subseteq \{x, y, z\} \cup W_2$.

- If $B_2 = \emptyset$, we have two possibilities:
  - if $x \in B$, then $B = \{x, y, z\}$ and $\beta \in A$ is not satisfied since $\frac{d_A(\beta)}{|A|-1} = \frac{k}{2k} < \frac{2}{3} = \frac{d_B(\beta)}{|B|}$;
  - if $x \in A$, then $B = \{y, z\}$ and $\alpha \in A$ is not satisfied since $\frac{d_A(\alpha)}{|A|-1} = \frac{k}{2k+1} < \frac{1}{2} = \frac{d_B(\alpha)}{|B|}$.
- If $B_2 \neq \emptyset$ and $B_2 \neq W_2$,
  - Case $x \in B$.
    - If there exists $u \in A_2$ such that $u \in N(x) \cup N(y)$, then if u is satisfied then:

$$\frac{|A_2|}{k + |A_2|} = \frac{d_A(u)}{|A| - 1} \geq \frac{d(u)}{|V| - 1} \geq \frac{k+1}{2k+3} \Rightarrow |A_2| \geq \frac{k^2 + k}{k+2} > k - 1\,,$$

      which is not possible since $|A_2| \leq k - 1$.
    - Otherwise, for all $u \in A_2$, $u \notin N(x) \cup N(y)$. Hence, for any $u \in A_2$, if u is satisfied then:

$$\frac{|A_2|}{k + |A_2|} = \frac{d_A(u)}{|A| - 1} \geq \frac{d(u)}{|V| - 1} = \frac{k}{2k+3} \Rightarrow |A_2| \geq \frac{k^2}{k+3}\,.$$

      Due to our assumptions about the graph, $|W_2 \cap (N(x) \cup N(y))| > \frac{3k}{k+3}$. Thus, $k - \frac{3k}{k+3} > |W_2 \setminus (N(x) \cup N(y))| \geq |A_2| \geq \frac{k^2}{k+3}$ which implies $k > k$, a contradiction.
  - Case $x \in A$. Let $u \in A_2$.
    - If $u \in N(y) \cap N(x)$, then if u is satisfied we have:

$$\frac{|A_2| + 1}{k + |A_2| + 1} = \frac{d_A(u)}{|A| - 1} \geq \frac{d(u)}{|V| - 1} = \frac{k+2}{2k+3}\,,$$

      which implies $|A_2| \geq k - \frac{1}{k+1}$, and then $|A_2| \geq k$, a contradiction since $B_2 \neq \emptyset$.
    - If $u \in N(y) \setminus N(x)$, then both $d_A(u)$ and $d(u)$ decrease by one. Similarly to the previous case, we obtain that $|A_2| \geq k + \frac{1}{k+2}$ and then $|A_2| > k$, a contradiction.

* If $u \in N(x) \setminus N(y)$, then:

$$\frac{|A_2|+1}{k+|A_2|+1} = \frac{d_A(u)}{|A|-1} \geq \frac{d(u)}{|V|-1} = \frac{k+1}{2k+3} \Rightarrow |A_2| \geq \frac{k^2-2}{k+2} > k-2\,.$$

Since assuming that there is a vertex in $A_2 \cap N(y)$ leads to a contradiction (see previous cases), we can assume that $A_2 \cap N(y) = \emptyset$. Then, since $d_{W_2}(y) \geq 2$, then $|W_2 \setminus N(y)| \leq k-2$. Thus $k-2 \geq |A_2| > k-2$, a contradiction.

* If $u \notin N(x) \cup N(y)$, then both $d_A(u)$ and $d(u)$ decrease by one. Similarly to the previous case, we obtain $|A_2| \geq k$, a contradiction since $|B_2| \neq \emptyset$.

- If $B_2 = W_2$, then either $B = \{x, y, z\} \cup W_2$, and we have $|A| + 2 = |B|$ but $d_A(x) = d_B(x)$ thus x is not satisfied, or $B = \{y, z\} \cup W_2$, and since $|A| = |B|$ we have: $\frac{d_B(y)}{|B|-1} < \frac{d_B(y)+1}{|B|} = \frac{d_A(y)}{|B|} = \frac{d_A(y)}{|A|}$, thus y is not satisfied.

We conclude that if there is a 2-community structure in G, then $B_1 \neq \emptyset$. The case $B_2 = \emptyset$ is similar and we also conclude that if there is a 2-community structure in G, $B_2 \neq \emptyset$.

Case 2: $B_1, B_2 \neq \emptyset$.

Without loss of generality, we suppose $|B_1| \leq |B_2|$. Let $u \in B_1$ and suppose that u is satisfied in the partition $\{A, B\}$. We prove that in all cases, if u is satisfied then it implies a contradiction with $|B_1| \leq |B_2|$.

- If $x \in A$
  - If $u \in N(x) \cap N(y)$:

$$\frac{|B_1|}{|B_1|+|B_2|+1} = \frac{d_B(u)}{|B|-1} \geq \frac{d(u)}{|V|-1} = \frac{k+2}{2k+3} \Rightarrow |B_1| > |B_2|\,,$$

    a contradiction, hence u is not satisfied.

  - If $u \in N(x) \setminus N(y)$, we have $d_B(u) = |B_1| - 1$ and $d(u) = k+1$ and similarly we obtain $|B_1|\cdot(k+2) \geq |B_2|\cdot(k+1)+(k+4) \geq |B_2|\cdot(k+1)+(|B_2|+4) > |B_2|\cdot(k+2)$, a contradiction since $|B_1| \leq |B_2|$.
  - If $u \in N(y) \setminus N(x)$, we have $d_B(u) = |B_1|$ and $d(u) = k+1$ and similarly we obtain $|B_1|\cdot(k+2) \geq |B_2|\cdot(k+1)+(k+1) \geq |B_2|\cdot(k+1)+(|B_2|+1) > |B_2|\cdot(k+2)$, a contradiction since $|B_1| \leq |B_2|$.

– If $u \notin N(x) \cup N(y)$, we have $d_B(u) = |B_1| - 1$ and $d(u) = k$ and similarly we obtain $|B_1| \cdot (k+3) \geq |B_2| \cdot k + 3(k+1) \geq |B_2| \cdot k + 3(|B_2|+1) > |B_2| \cdot (k+3)$, a contradiction since $|B_1| \leq |B_2|$.

- If $x \in B$
  – If $u \in N(x) \cap N(y)$:
  $$\frac{|B_1|+1}{|B_1|+|B_2|+2} = \frac{d_B(u)}{|B|-1} \geq \frac{d(u)}{|V|-1} = \frac{k+2}{2k+3} \Rightarrow |B_1| > |B_2|\,,$$
  a contradiction, hence u is not satisfied.

  – If $u \in N(x) \setminus N(y)$ or $u \in N(y) \setminus N(x)$, we have $d_B(u) = |B_1|$ and $d(u) = k+1$ and similarly we obtain $|B_1| \cdot (k+2) \geq |B_2| \cdot (k+1) + 2(k+1) \geq |B_2| \cdot (k+1) + 2(|B_2|+1) > |B_2| \cdot (k+3)$, a contradiction since $|B_1| \leq |B_2|$.
  – If $u \notin N(x) \cup N(y)$, we have $d_B(u) = |B_1| - 1$ and $d(u) = k$ and similarly we obtain $|B_1| \cdot (k+3) \geq |B_2| \cdot k + 4k + 3 \geq |B_2| \cdot \frac{k}{k+3} + 4 \cdot |B_2| + 3 > |B_2| \cdot (k+4)$, a contradiction since $|B_1| \leq |B_2|$.

□

On the other hand, any graph in $\mathcal{G}$ has $\{W_1 \cup \{x, w\}, W_2 \cup \{y, z\}\}$ as a balanced weak 2-community structure. Thus, the existence of graphs in which there is no weak 2-community structure remains open for all graphs.

4.6 Conclusions

The following overview summarises the results achieved in this chapter. All considered graphs are of size at least 4 and are not stars. The problem of finding a 2-community structure in a graph has been studied in the following graph classes.

(i) *trees:*
– a connected 2-community structure exists and can be found in linear time (Theorem 4.6),
– there are trees with a balanced 2-community structure, but without a connected balanced weak 2-community structure (Remark 4.31),

(ii) *graphs of maximum degree* 3*:*
– a connected 2-community structure exists and can be found in polynomial time (Theorem 4.16),
– a balanced weak 2-community structure exists and can be found in polynomial time (Theorem 4.30),
– there are graphs without a balanced 2-community structure (Remark 4.29),
– there are graphs with a balanced 2-community structure, but without a connected balanced weak 2-community structure (Remark 4.31)

(iii) *graphs of minimum degree* $(|V| - 3)$*, complements of bipartite graphs, graphs with minimum degree* $\lceil \frac{(c-1)\cdot|V|}{c} \rceil$ *where* c *is the size of an inclusion-wise maximal clique in the graph:*
– a connected 2-community structure exists and can be found in polynomial time (Theorems 4.18, 4.19, 4.20)

(iv) *graphs of bounded tree-width:*
– there are graphs without a balanced 2-community structure (Remark 4.29), but to decide whether such a structure exists and if it exists, find it, can be done in polynomial time (Remark 4.32)

Estivill-Castro *et al.* [58] proved that the problem of finding a balanced 2-community structure is NP-complete in general graphs. In Section 4.4 it has been showed that the same result also holds for a weak community, even with additional constraint of connectivity for both parts. It also has been presented a shorter proof of the known NP-complete result for a balanced 2-community structure in general graphs based on an alternative definition of community structure [20], which also implies NP-completeness for a connected balanced 2-community structure.

In case of Balanced 2-Community the situation is different. Any graph of maximum degree 3 has a balanced weak 2-community structure, while there are graphs without a balanced 2-community structure within the same class. Computationally speaking, finding a balanced weak 2-community structure can be done in polynomial time in graphs of maximum degree 3 while the Balanced 2-Community problem is NP-complete in general graphs just as its weak version. The results are similar for connected 2-community structures.

Finally, we found a family of graphs that do not contain any 2-community structure. Finding other families of graphs that satisfy this property could help to distinguish graphs that contain 2-community structures and the other ones. On the other hand, our family of graphs always have a balanced weak 2-community structure.

In addition, in order to get a better understanding of community structures, there are other interesting problems left open, as to extend 2-community results to other graph classes, to characterize graph classes where the existential and complexity results for 2-community and weak 2-community problems and their connected versions are different or to generalize the results to k-communities for a fixed k, $k \geq 3$.

5 Max Community

Contents

The content of this chapter is based on the following article in preparation:

- C. Bazgan, J. Chlebíková, C. Dallard and T. Pontoizeau, *Community of maximum size: complexity and approximation*, in preparation.

5.1 Introduction

As we saw in Chapter 4, there exist graphs in which there is no 2-community structure. In this way, it is interesting to consider a relaxation of the definition by accepting that one of the two parts may not fulfill the proportion condition. In this way, we investigated finding a community of maximum size.

In this chapter, we use the definition of a community of Olsen [133] *i.e.* a community is defined as a subset C of vertices such that each vertex of C has a greater proportion of neighbors in C than outside of C. We study the problem of, given a graph, finding a community of maximum size. Section 5.2 gives notations that we use in this paper. Section 5.3 discuss the NP-hardness of the problem. Section 5.4 gives several results about the approximation of this problem. We investigate the co-NP-hardness of deciding if a community is inclusion-wise maximal in Section 5.6. We discuss the linear time solvability of the problem in some graph classes, and in particular in Hamiltonian cubic graphs, in Section 5.5. Conclusions and open problems are given in Section 5.7.

5.2 Preliminaries

In this chapter all considered graphs are simple, undirected and connected. Let $G = (V, E)$ be a graph, $C \subset V$ a subset of vertices and $v \in V$ a vertex.

For a given graph $G = (V, E)$, Olsen ([133]) defined a 2-community structure as a partition $\{C_1, C_2\}$ of V such that each part has at least two vertices and each vertex has a greater proportion of neighbor in its part than in the other part, *i.e.* for each vertex in C_i, $i \in \{1, 2\}$, $\frac{d_{C_i}(v)}{|C_i|-1} \geq \frac{d_{\overline{C_i}}(v)}{|\overline{C_i}|}$. In this chapter, we investigate a variant consisting in finding only one community (in the sense that the previous condition only need to be fulfilled for vertices of one part of the partition) of maximum size. Such condition will still be called *proportion condition* throughout this chapter.

Definition 5.1. *Given a graph $G = (V, E)$, a community is a subset of vertives $C \subset V$, $C \neq V$, such that $|C| \geq 2$ and for each vertex $u \in C$,*

$$\frac{d_C(u)}{|C| - 1} \geq \frac{d_{\bar{C}}(u)}{|\bar{C}|}. \tag{5.1}$$

If a vertex $u \in C$ respects the latter proportion condition, we say u is satisfied (in C).

Notice that since we are looking for a community of maximum size, a community of size less than 2 is not worth of interest. Thus, we can derive that the proportion condition is equivalent to $|\bar{C}| \cdot d_C(u) \geq (|C| - 1) \cdot d_{\bar{C}}(u)$, that we also use in the chapter.

Observe that the induced subgraph of a community may either be connected or not. We study both cases and talk about *connected community* when the subgraph induced by the community is connected.

We define our main problem Max Community as follows:

Max Community
Input: A graph $G = (V, E)$.
Output: A community in G of maximum size.

5.3 Hardness results

In this section we prove two major results regarding the complexity of Max Community. We show that it is NP-hard and APX-hard, even for split graphs.

5.3.1 NP-hardness

We establish a reduction from the decision version associated to Max Independent Set which was proved NP-hard in [104]:

Max Independent Set **Input:** A graph $G = (V, E)$. **Output:** A subset S of vertices of minimum size such that any edge in E is adjacent to a vertex in S.

In order to do so, we first introduce the transformation involved in the reduction and some lemmas that will be necessary for the main theorem.

Definition 5.2. *Let $G = (V, E)$ be a connected graph. We define the construction Γ transforming the graph G into $\Gamma(G) = G'(V', E')$ as it follows (see Figure 5.1 for an example):*

- *$V' := M \cup N \cup \{z_1, z_2\}$, where $M := \{e : e \in E\}$, $N := V$ and z_1, z_2 are two new vertices;*
- *for all $e \in M$ and all $u \in N$, the edge $(e, u) \in E'$ if and only if $u \notin e$;*
- *the subgraph $G'[\{z_1, z_2\} \cup M]$ is complete.*

Notice that the construction from Definition 5.2 can be done in polynomial time. Moreover, $\Gamma(G)$ yields a connected graph if and only if G is not isomorphic to a star.

Lemma 5.3. *Let $G = (V, E)$ be a connected graph not isomorphic to a star and let $\Gamma(G) = G' = (V', E')$ be a graph defined following Definition 5.2. Let $C \subset V'$ be a set of vertices such that $M \cup \{z_1, z_2\} \subseteq C$. A vertex $w \in C$ is satisfied in C if and only if $d_C(w) \geq |C| - 2$.*

Proof. Obviously, $\bar{C} \subset N$. Let $u \in N \cup \{z_1, z_2\}$ then $d_{\bar{C}}(u) = 0$. Therefore, any vertex in $C \cap (N \cup \{z_1, z_2\})$ is satisfied in C. A vertex $e \in M$ has a degree $d(e) = |V'| - 3$. Hence if $d_C(e) < |C| - 2$, then $d_{\bar{C}}(e) = |\bar{C}|$ and e is not satisfied in C as it does not respect the proportion condition. However, if $d_C(e) \geq |C| - 2$, then $d_{\bar{C}}(e) < |\bar{C}|$ and since $|C| > |\bar{C}|$ we have

$$|\bar{C}| \cdot d_C(e) \geq |\bar{C}| \cdot (|C| - 2) > (|C| - 1) \cdot (|\bar{C}| - 1) \geq (|C| - 1) \cdot d_{\bar{C}}(e)\,,$$

and thus e is satisfied in C. □

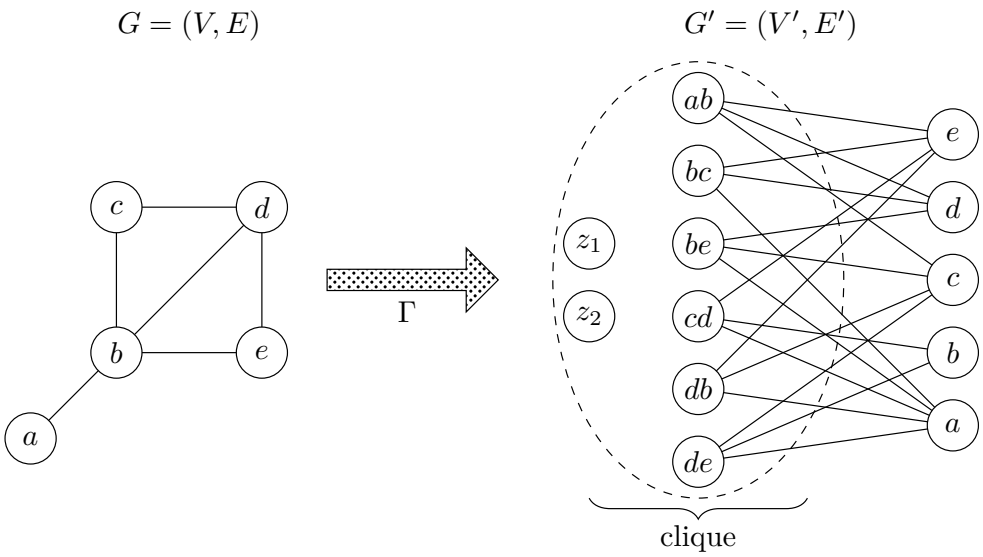


Figure 5.1: Example of the transformation Γ.

Lemma 5.4. *Let $G = (V, E)$ be a connected graph not isomorphic to a star and let $G' = (V', E') = \Gamma(G)$. Let $C_1 \subset V'$ be a community in G'. Then there exists a community C_2 in G' such that $|C_2| \geq |C_1|$ and $M \cup \{z_1, z_2\} \subseteq C_2$. Moreover, C_2 can be found in polynomial time.*

Proof. Firstly, we claim that N is not included in C_1. We denote by $\overline{C}_1 := V' \setminus C_1$ the set of vertices which are not in the community C_1. To prove a contradiction, consider the two following cases:

- if $C_1 = N$, since $G'[N]$ is an independent set, then any vertex $u \in C_1$ has $d_{C_1}(u) = 0$ and $d_{\overline{C}_1}(u) > 0$; hence u does not satisfies the proportion condition and C_1 is not a community;
- if $N \subset C_1$, then $\overline{C_1}$ is a subset of the clique $M \cup \{z_1, z_2\}$; it means any vertex $u \in M \cup \{z_1, z_2\} \cap C_1$ has $d_{\overline{C_1}}(u) = |\overline{C_1}|$ and $d_{C_1}(u) < |C_1| - 2$, thus

$$|\overline{C}_1| \cdot d_{C_1}(u) < |\overline{C}_1| \cdot (|C_1| - 2) < (|C_1| - 1) \cdot |\overline{C}_1| = (|C_1| - 1) \cdot d_{\overline{C}_1}(u)\,,$$

  hence v does not satisfies the proportion condition and C_1 is not a community.

Thus, we can consider that N is not included in C_1. Now, let $C_2 := C_1 \cup M \cup \{z_1, z_2\}$ and $\overline{C_2} := V' \setminus C_2$. Notice that $\overline{C_2} \subseteq N$ and that a vertex $u \in C_1 \cap N$ is satisfied in C_2, since $d_{\overline{C_2}}(u) = 0$. Obviously, z_1 and z_2 are satisfied in C_2.

Let $e \in M$ be a vertex not satisfied in C_2. Hence, for all $e \in M$ which is not satisfied in C_2, transfer a vertex $u \in N$, non adjacent to e, from C_2 to $\overline{C_2}$. Notice that for any $f \in M \cap C_1$, $d_{C_2}(f) \geq d_{C_1}(f)$ and $d_{\overline{C_2}}(f) = d_{\overline{C_1}}(f)$, hence f is satisfied in C_2. Since e

is unsatisfied, then $e \in M \setminus C_1$ and at most $|M \setminus C_1|$ transfers are needed to satisfy all the vertices in C_2. Thus $|C_2| \geq |C_1|$ holds true. Furthermore, observe that for any vertex $w \in C_1$, the condition $d_{C_2}(w) \geq |C_2| - 2$ stays true after the transfer of u, hence according to Lemma 5.3 w remains satisfied. Also, since C_2 is not included in N and C_2 is not included in N then $C_2 \neq V$.

Therefore, all vertices of C_2 are satisfied, and $|C_2| \geq |C_1|$. Obviously, C_2 can be found in polynomial time. □

We can now prove the main theorem:

Theorem 5.5. Max Community *is NP-hard, even on split graphs.*

Proof. We consider the decision variants of the problems Max Community and Max Independent Set. Let $G = (V, E)$ be a connected graph not isomorphic to a star and let $G' = (V', E') := \Gamma(G)$ and $k \in \{1, \ldots, |V| - 1\}$. We claim that there is an independent set of size at least k in G if and only if there is a community of size at least $|M| + 2 + k$ in G'.

Let R be a independent set of G of size at least k. In G', we define $C := M \cup \{z_1, z_2\} \cup R$ and $\overline{C} := V' \setminus C$. First, note that $R \subseteq N$ thus $\overline{C} = N \setminus R$. The vertices in $C \cap N \cup \{z_2, z_2\}$ are obviously satisfied in C as they only have neighbors in C. Hence, the only possible unsatisfied vertices are from the set M. Consider a vertex $e \in M$. Since R is an independent set of G, then for each edge $e = (u, v) \in E$ at most one of the vertices u and v belongs to R. Hence, the vertex $e \in M$ is not adjacent to at most one vertex in C and thus $d_C(e) \geq |C| - 2$. According to Lemma 5.3, e is satisfied. Thus C is a community of size at least $|M| + 2 + k$.

Let C be a community in G' of size at least $|M| + 2 + k$. According to Lemma 5.4, there exists a community C' of G' such that $|C'| \geq |C|$ and $\{z_1, z_2\} \cup M \subseteq C'$. We claim that $R' := C' \cap N$ is an independent set of G of size at least k. Obviously $|R'| \geq k$. Moreover, Lemma 5.3 states that for all satisfied vertices $w \in C'$, $d_{C'}(e) \geq |C'| - 2$. Hence for each vertex $e \in M$ there is at most one vertex $u \in C$ that is not adjacent to e. Since a non-edge between vertices $e \in M$ and $u \in N$ in G' implies $u \in e$ in G, then the edge $e \in E$ has at most one endpoint $u \in R'$ in the graph G. Thus, R' is an independent set of size at least k. □

Notice that there exists graphs on which a community of maximum size is not connected, even if the graph is a cubic graph (Figure 5.2) or a caterpillar (Figure 5.3).

5.3.2 Non-approximability

The previous reduction from Theorem 5.5 is actually an L-reduction which allow us to establish its APX-hardness.

Proposition 5.6. Max Community *is APX-hard, even on split graphs.*

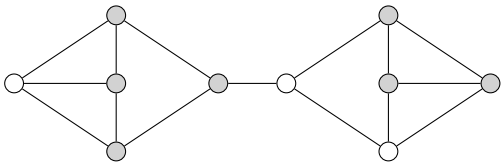


Figure 5.2: A cubic graph where the community of maximum size (in gray) is disconnected.

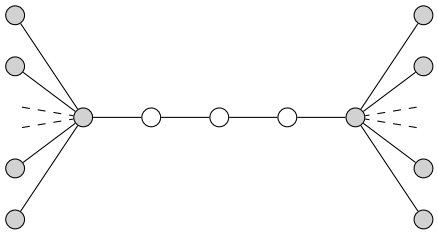


Figure 5.3: A caterpillar $T = (V, E)$ constructed by two stars S_d for some integer d such that their center are joined by a path of length 2. The community in gray is a disconnected community of maximum size, whereas a connected community of maximum size have no more than $\frac{|V|}{2}$ vertices.

Proof. We prove that the reduction from Theorem 5.5 is an L-reduction when we reduce Max Independent Set on cubic graphs to Max Community. Let I be an instance of Max Independent Set on the cubic graph $G = (V, E)$ and we construct an instance I' of Max Community defined on the graph $G' = (V', E') = \Gamma(G)$.

We recall that since a cubic graph is 4-colorable, $opt(I) \geq \frac{|V|}{4}$. Since a cubic graph contains exactly $\frac{3|V|}{2}$ edges we have $opt(I') = 2 + |E| + opt(I) = 2 + \frac{3|V|}{2} + opt(I) \leq 2 + \frac{3|V|}{2} + |V| \leq 2 + \frac{5|V|}{2} \leq 2 + 10 \cdot opt(I) \leq 11 \cdot opt(I)$.

Moreover, for any community S' in G' we can construct an independent set S in G of size $|S| = |S'| - |E| - 2$. Since $opt(I') \geq |S'| = |S| + |E| + 2$, in particular, when S is an optimum independent set, $opt(I') \geq opt(I) + |E| + 2$. In addition, $opt(I) \geq |S| = |S'| - |E| - 2$, in particular, when S' is an optimum community, $opt(I) \geq opt(I') - |E| - 2$. Thus we obtain $opt(I) = opt(I') - |E| - 2$ and $opt(I) - |S| = opt(I') - |S'|$.

Since Max Independent Set is APX-hard on cubic graphs [7], we conclude that Max Community is APX-hard on split graphs. □

Corollary 5.7. *There is no ptas for* Max Community *unless P=NP.*

In the next section, we discuss some positive results for approximation.

5.4 Positive results for approximation

We show that MAX COMMUNITY is polynomial-time 2-approximable, which establishes its APX-completeness. We also give a polynomial-time $\frac{2\cdot(\Delta-1)+1}{\Delta}$ - approximation algorithm, where Δ is the maximum degree of the graph, using an upper-bound of the size of a community.

We design a polynomial-time algorithm that generates, given a graph $G = (V, E)$, a community of size at least $\lceil \frac{|V|}{2} \rceil$.

Lemma 5.8. *Let $G = (V, E)$ be a graph with n vertices. Let $C \subset V$ be a set of vertices of size $\frac{n}{2}$ or $\frac{n}{2} + 1$ for n even, and $\frac{n+1}{2}$ for n odd. If C is not a community in G, then there exists a vertex $u \in C$ such that $d_C(u) < d_{\bar{C}}(u)$ if $|C| \leq \frac{n+1}{2}$, and $d_C(u) \leq d_{\bar{C}}(u)$ otherwise.*

Proof. Let $C \subset V$ such that C is not a community. Hence, there exists a vertex $u \in C$ such that the proportion condition is not satisfied and thus $|\bar{C}| \cdot d_C(u) < (|C| - 1) \cdot d_{\bar{C}}(u)$ (*).

- If $|C| = \frac{n}{2} + 1$ (n even), assume by contradiction that $d_C(v) > d_{\bar{C}}(v)$, for each vertex $v \in C$. In particular inequality (*) becomes $(\frac{n}{2} - 1) \cdot (d_{\bar{C}}(u) + 1) < \frac{n}{2} \cdot d_{\bar{C}}(u)$ which is true if and only if $d_{\bar{C}}(u) \geq \frac{n}{2}$. Thus $d(u) = d_C(u) + d_{\bar{C}}(u) > n$, which is not possible.

- If $|C| = \lceil \frac{n}{2} \rceil$, assume by contradiction that $d_C(v) \geq d_{\bar{C}}(v)$, for each vertex $v \in C$. In particular inequality (*) becomes $\lfloor \frac{n}{2} \rfloor \cdot d_C(u) < (\lceil \frac{n}{2} \rceil - 1) \cdot d_{\bar{C}}(u) \leq \lfloor \frac{n}{2} \rfloor \cdot d_{\bar{C}}(u)$, which contradicts our assumption.

□

Theorem 5.9. *Given a connected graph on n vertices, a community of size at least $\lceil \frac{n}{2} \rceil$ vertices can be constructed in linear time.*

Proof. We apply Algorithm 1 on G and prove that the algorithm terminates.

Algorithm 1: Find a community of size at least $\frac{n}{2}$.

Input: $G = (V, E)$ a graph with n vertices.
Output: $C \subset V$ a community in G.
Let $C \subset V$ with $|C| = \lceil \frac{n}{2} \rceil$;
while C *is not a community* **do**
    Let $u \in C$ such that $d_{\bar{C}}(u) - d_C(u)$ is maximum;
    $C := \bar{C} \cup \{u\}$;
return C;

- **Case 1:** n **is odd.** Notice that at the end of each loop, the set C is modified without changing its size $|C| = \frac{n+1}{2}$. If C is not a community, then according to Lemma 5.8 there is a vertex $v \in C$ for which $d_C(v) < d_{\overline{C}}(v)$. Therefore, the vertex u chosen within the loop has $d_{\overline{C}}(u) - d_C(u) > 0$. Thus the size of the cut between C and $\overline{C}$ decreases after each loop and the algorithm terminates.

- **Case 2 :** n **is even.** Notice that Algorithm 1 starts with $|C| = \frac{n}{2}$. If C is not a community in G, then due to Lemma 5.8, it exists a vertex $v \in C$ such that $d_C(v) < d_{\overline{C}}(v)$. The selection of the vertex $u \in C$ inside the loop ensures that the size of the cut between C and $\overline{C}$ strictly decreases at the end of the loop. Now observe that after the first loop, $|C| = \frac{n}{2} + 1$. If C is not a community, according to Lemma 5.8, there exists a vertex $v \in C$ such that $d_C(v) \leq d_{\overline{C}}(v)$. Therefore the vertex u inside the loop has $d_C(u) \leq d_{\overline{C}}(u)$. Obviously, after the second loop, $|C| = \frac{n}{2}$. Since after each loop $|C|$ alternates between $\frac{n}{2}$ and $\frac{n}{2}+1$, the cut between C and $\overline{C}$ strictly decreases every two loops, and the algorithm terminates. Thus the algorithm enters while-loop at most $O(|E|)$ times and terminates when C is a community.

□

Algorithm 1 implies several consequences. Firstly, it gives a 2-approximation algorithm since any community has size at most $|V| - 1$. Besides, it shows that the decision version associated to Max Community is in FPT when parameterized by the natural parameter k (*i.e.* the size of the solution). Indeed, if the parameter $k \leq \lceil \frac{|V|}{2} \rceil$, then a community of size greater than k can be found in polynomial time using Algorithm 1. On the other hand, if $k > \lceil \frac{|V|}{2} \rceil$, then we have $|V| < 2k$ and an exhaustive research can be done in $O(2^{2k})$ operations.

We show in the following how the calculation of the approximation ratio can be improved with regard to the maximum degree of the graph.

Lemma 5.10. *Let $G = (V, E)$ be a connected graph and $C \subset V$ be a community in G. Then $|C| \leq \lfloor \frac{|V| \cdot (\Delta(G)-1)+1}{\Delta(G)} \rfloor$.*

Proof. Let v be a vertex of C with at least one neighbor in $\overline{C} = V \setminus C$ (such vertex exists since G is connected). Since C is a community, v fulfills the proportion condition, that is $\frac{\Delta(G)-1}{|C|-1} \geq \frac{d_C(v)}{|C|-1} \geq \frac{d_{\overline{C}}(v)}{|\overline{C}|} \geq \frac{1}{|V|-|C|}$ which implies that $|C| \leq \frac{|V| \cdot (\Delta(G)-1)+1}{\Delta(G)}$. Since $|C|$ is an integer, we obtain $|C| \leq \lfloor \frac{|V| \cdot (\Delta(G)-1)+1}{\Delta(G)} \rfloor$. □

Proposition 5.11. Max Community *is polynomial-time $\frac{2 \cdot (\Delta(G)-1)+1}{\Delta(G)}$-approximable.*

Proof. Let $G = (V, E)$ be a graph, C be a solution given by Algorithm 1 and $opt(G)$ denote the size of a community of maximum size in G. According to Lemma 5.10 we have $opt(G) \leq \frac{|V| \cdot (\Delta(G)-1)+1}{\Delta(G)}$. Thus we obtain $|C| \geq \frac{|V|}{2} \geq \frac{\Delta(G)}{2 \cdot (\Delta(G)-1)+1} \cdot \frac{|V| \cdot (\Delta(G)-1)+1}{\Delta(G)} \geq \frac{\Delta(G)}{2 \cdot (\Delta(G)-1)+1} \cdot opt(G)$. □

As mentioned above, regardless of the maximum degree of the graph, Algorithm 1 gives a 2-approximation for MAX COMMUNITY since $|C| \geq \frac{1}{2} \cdot |V| \geq \frac{1}{2} \cdot opt(G)$.

Theorem 5.12. MAX COMMUNITY *is APX-complete.*

5.5 Polynomial-time solvability in some graph classes

5.5.1 Some easy graph classes

First notice that the problem is easy to solve in graphs of maximum degree 2. Indeed, given a connected graph $G = (V, E)$, any community C cannot have a size greater than $\lfloor \frac{|V|+1}{2} \rfloor$ since of Lemma 5.10. Then, Algorithm 1 gives a community of maximum size.

Even if MAX COMMUNITY is NP-complete on split graphs, we show that MAX COMMUNITY can be solved easily on some particular split graphs, namely threshold graphs, by noticing that a community of any size can be found in linear time.

Proposition 5.13. *Any threshold graph $G = (V, E)$ contains a community of size exactly k for any integer $k \in \{1, \ldots, |V| - 1\}$.*

Proof. Since G is a threshold graph, we can order the vertices of the independent set as $v_1, \ldots, v_p$ such that $N(v_1) \subseteq N(v_2) \subseteq \ldots \subseteq N(v_p)$.

If $k \leq |V| - p$, then any k vertices from the clique of the graph is trivially a community. Now suppose that $k > |V| - p$, then we claim that the subset $C := V \setminus \{v_1, v_2, \ldots, v_{|V|-k}\}$ is a community of size exactly k. Indeed, any vertex from C in the independent set has no neighbor in $V \setminus C$, so they trivially satisfy the proportion condition. Moreover, any vertex from C in the clique is linked to any other vertex of C by the choice of the vertices we put outside of C, and thus trivially satisfies the proportion condition. Thus C is a community and has size k. □

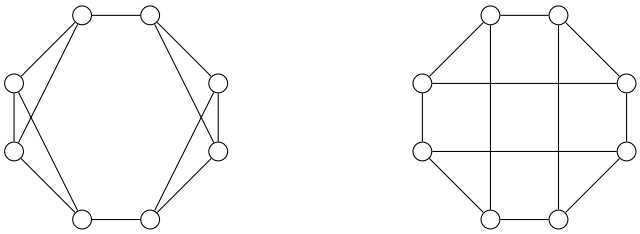
Corollary 5.14. MAX COMMUNITY *can be solved in linear time on threshold graphs.*

Proof. Let $G = (V, E)$ be a threshold graph. According to Proposition 5.13, a community of size $|V| - 1$ can be found using the structure of threshold graphs. Since threshold graphs can be recognized in linear time [96], the result follows. □

5.5.2 Hamiltonian cubic graphs

Now we prove that any Hamiltonian cubic graph $G = (V, E)$ (except two graphs, see Figure 5.4) has a community of the maximum possible size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$ (see Lemma 5.10 for upper bounds on a community size). Furthermore, we can find such a community in linear time if a Hamiltonian cycle is given.

A Hamiltonian cubic graph $G = (V, E)$ can be described as a set of vertices $V = \{0, 1, \ldots, |V| - 1\}$, a Hamiltonian cycle and a set of edges between non-successive vertices



H_1 H_2

Figure 5.4: Two Hamiltonian cubic graphs H_1 and H_2 with 8 vertices in which there is no community of size $\lfloor \frac{2 \cdot 8+1}{3} \rfloor = 5$.

in the cycle. Without loss of generality, we assume that such graphs have a Hamiltonian cycle $(0, 1, ..., |V| - 1, 0)$. To avoid tedious notations, we may use $i \in \mathbb{Z}$ to refer to the vertex $(i \pmod n)$.

We introduce two essential notions that will be used throughout the proof.

Definition 5.15. *Let $G = (V, E)$ be a Hamiltonian cubic graph and $k := \lceil \frac{|V|-1}{3} \rceil$, then:*

- *a good shift denotes a set $P := \{u, u+1, u+2, \ldots, u-k-1\}$ for some $u \in V$, such that $d_P(u) = d_P(u-k-1) = 2$.*
- *an almost good shift denotes a set $P := \{u, u+1, u+2, \ldots, u-k\}$ for some $u \in V$, such that $d_P(u) = d_P(u-k) = 2$.*

Notice that a good shift P is a community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$ since any vertex of P has at least two neighbors in P due to the structure of the graph. An almost good shift P is of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor + 1$, and thus is not a community due to Lemma 5.10, but has the property that any vertex of P has at least two neighbors in P.

Now, we prove that if G does not contain a good shift, then G contains an almost good shift P and, in such case, it is always possible to find a vertex $v \in P$ such that $P \setminus \{v\}$ is a community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$.

Due to the structure of the graph, we can highlight specific sets of vertices of a Hamiltonian cubic graph:

Definition 5.16. *Let $G = (V, E)$ be a Hamiltonian cubic graph, and $k := \lceil \frac{|V|-1}{3} \rceil$, then:*

- *for each vertex $u \in V$, $c(u)$ denotes the vertex $v \in V$ such that $uv \in E$ and $|u-v| > 1$;*
- $L := \{u \in V : c(u) \in \{u-k, u-k+1, , \ldots, u-2\}\}$;
- $R := \{u \in V : c(u) \in \{u+2, u+3, \ldots, u+k\}\}$;

We derive some observations from these definitions. For a given Hamiltonian cubic graph $G = (V, E)$ and $u \in V$, notice that $u \in L$ if and only if $c(u) \in R$, and symmetrically. This particularly implies that $|L| = |R| \leq \frac{|V|}{2}$. Moreover, notice that for a vertex $u \in L$, the set $P := \{u, u+1, \cdots, u-k-1\}$ cannot be a good shift, since $d_P(u) = 1$. Symmetrically, if $u \in R$, the set $P := \{u+k+1, u+k+2, \ldots, u-1, u\}$ cannot be a good shift, since $d_P(u) = 1$. These observations are summed up in the following lemma.

Lemma 5.17. *Let $G = (V, E)$ be a Hamiltonian cubic graph, $k := \lceil \frac{|V|-1}{3} \rceil$ and $u \in V$. If $u \notin L$ and $(u-(k+1)) \notin R$, then the set $\{u, u+1, \ldots, u-(k+1)-1, u-(k+1)\}$ is a good shift. Symmetrically, if $u \notin R$ and $(u+k+1) \notin L$, then the set $\{u+k+1, u+k+2, \ldots, u-1, u\}$ is a good shift.*

Proof. The proof is straightforward. Since $u \notin L$ and $(u-(k+1)) \notin R$, we have $d_P(u) = d_P(u-(k+1)) = 2$, where $P := \{u, u+1, \cdots, u-(k+1)\}$. The other case is symmetrical. □

A natural consequence of Lemma 5.17 is that if a Hamiltonian cubic graph G has no good shift, then we can define a whole set of vertices that must be either in L or in R. In that way, we define the following:

Definition 5.18. *Given a Hamiltonian cubic graph $G = (V, E)$ and a vertex $u \in V$, we define $<u>:= \{u + \delta \cdot (k+1) : \delta \geq 1, \delta \in \mathbb{N}\}$ with $k := \lceil \frac{|V|-1}{3} \rceil$.*

Applying Lemma 5.17 recursively on a graph without any good shift, we obtain Corollary 5.19.

Corollary 5.19. *Let $G = (V, E)$ be a Hamiltonian cubic graph without a good shift and $u \in V$. Therefore:*

- *if $u \notin R$ then $<u> \subseteq L$ and also $u \in L$,*
- *if $u \notin L$, then $<u> \subseteq R$ and also $u \in R$,*
- $|L| = |R| = \frac{|V|}{2}$.

Proof. First notice that since $u + |V| \cdot (k+1) = u$, we have $u \in <u>$. If $u \notin R$, we can apply Lemma 5.17 recursively and derive that $<u> = \{u + \delta \cdot (k+1) : \delta \geq 1, \delta \in \mathbb{N}\} \subseteq L$, and since $u \in <u>$, $u \in L$.

Symmetrically, if $u \notin L$, then $\{u - \delta \cdot (k+1) : \delta \geq 1, \delta \in \mathbb{N}\} \subseteq R$. Since $u - \delta \cdot (k+1) = u - \delta \cdot (k+1) + |V| \cdot (k+1) = u + \delta \cdot (|V|-1) \cdot (k+1)$, we have $\{u - \delta \cdot (k+1) : \delta \geq 1, \delta \in \mathbb{N}\} = <u>$. Thus, $<u> \subseteq R$, and since $u \in <u>$, $u \in R$.

This implies that for any vertex $u \in V$, $u \in L$ or $u \in R$. Finally, since $u \in L$ if and only if $c(u) \in R$ and $u \in R$ if and only if $c(u) \in L$, then it is obvious that $|L| = |R| = \frac{|V|}{2}$. □

Now, given a Hamiltonian cubic graph $G = (V, E)$ that does not contain any good shift, we show that V can be partitioned into sets $< i >$ with $i \in \{0, 1, \cdots, gcd(|V|, k+1) - 1\}$, with $gcd(|V|, k+1)$ the greatest common divisor of $|V|$ and $k+1$. This partition will be

useful to display an *almost good shift* P and a vertex to remove from P in order to obtain a community in G. This result comes from a basic property of the cyclic group $\mathbb{Z}/n\mathbb{Z}$ that we recall on Lemma 5.20.

Lemma 5.20. *Let $n \geq 4$ be some integer, $k := \lceil \frac{n-1}{3} \rceil$ and $d := \gcd(k+1, n)$. If all integers are considered mod n, then $\{0, 1, \cdots, n-1\} = \cup_{i \in \{0,1,\cdots,d-1\}} < i >$, and for any $i, j \in \{0, 1, \cdots, d-1\}$ with $i \neq j$, $< i > \cap < j >= \emptyset$.*

Proof. First we prove that for any $u \geq d$, $u \in < i >$ for some $i \in \{0, 1, \cdots, d-1\}$. Let $u \geq d$. Then there exist two integers a, b with $b \leq d-1$, such that $u = a \cdot d + b$. Moreover there exist two integers c, f such that $c \cdot (k+1) + f \cdot n = d$ since $d = \gcd(k+1, n)$. Then, $u = a \cdot c \cdot (k+1) + a \cdot f \cdot n + b = b + a \cdot c \cdot (k+1)$. Thus $u \in < b >$ with $b \leq d-1$. This proves that any integer is in a set $< i >$ for some $i \leq d-1$, *i.e.* $\{0, 1, \cdots, n-1\} = \cup_{i \in \{0,1,\cdots,d-1\}} < i >$.

In order to prove the second part, we first show that $n = | < u > | \cdot d$ for any $u \in \{0, 1, \cdots, n-1\}$. Let $u \in \{0, 1, \cdots, n-1\}$ and $p \geq 1$ be the smallest integer such that $u + p(k+1) = u$. Notice that $| < u > | = p$ and let us show that $n = p \cdot d$. Let n', k' be two integers such that $n = n' \cdot d$, $k+1 = k' \cdot d$ and $\gcd(n', k') = 1$. We prove that $n' = p$ by verifying that n' divides p and p divides n'. First, notice that $u + n' \cdot (k+1) = u + n' \cdot k' \cdot d = u + n \cdot k' = u$. Thus, p divides n'. On the other hand, notice that $u = u + p(k+1) = u + p \cdot k' \cdot d$, then $p \cdot k' \cdot d = 0$. This implies that n divides $p \cdot k' \cdot d$ and thus n' divides $p \cdot k'$. Since $\gcd(n', k') = 1$, n' divides p. Now notice that two sets $< i >$, $< j >$ for some integers i, j are either equal or disjoint. Since for any $u \in \{0, 1, \cdots, n-1\}$ we have $| < u > | = \frac{n}{d}$, then obviously all sets $< i >$, $i \in \{0, 1, \cdots, d-1\}$ are disjoints. □

Now, given a Hamiltonian cubic graph $G = (V, E)$ and $k := \lceil \frac{n-1}{3} \rceil$, we discuss the possible values of $gcd(|V|, k+1)$ and prove how one can find a community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$ in G. First, we show that if $|V| = 3k$, then there is always a good shift in G.

Lemma 5.21. *Let n be an even integer, $n \geq 4$. Then:*

- *if $n = 3k-1$, then $\gcd(n, k+1) \in \{2, 4\}$,*
- *if $n = 3k$, then $\gcd(n, k+1) \in \{1, 3\}$,*
- *if $n = 3k+1$, then $\gcd(n, k+1) = 2$.*

Proof. Consider the case $n = 3k-1$, then $d := \gcd(k+1, 3k-1) = \gcd(k+1, 3k-1-2(k+1)) = \gcd(k+1, k-3) = \gcd(4, k-3)$. As n is even, then k is odd and $d \in \{2, 4\}$. The other cases can be proved using the same reasoning. □

Corollary 5.22. *Let G be a Hamiltonian cubic graph with $3k$ vertices, $k \geq 2$. Then G has a good shift.*

Proof. Suppose by contradiction that there is no good shift in $G = (V, E)$. Notice that if $|V| = 3k$, then $k = \lceil \frac{|V|-1}{3} \rceil$. Let $d := \gcd(k+1, |V|)$. From Lemma 5.21 we get $d \in \{1, 3\}$. According to Corollary 5.19, $|L| = |R| = \frac{|V|}{2}$. If $d = 1$, then $V = < 0 >$ (Lemma 5.20),

hence $V = L$ or $V = R$, which is impossible. If $d = 3$, then $|V| = <0> \cup <1> \cup <2>$ (Lemma 5.20). According to Corollary 5.19, $<i> \subseteq L$ or $<i> \subseteq R$ for any $i \in \{0, 1, 2\}$, thus $|R| \neq |L|$, which is not possible. □

From Lemma 5.20 and Lemma 5.21, if a Hamiltonian cubic graph $G = (V, E)$ has no good shift, then V can be written as $V = <0> \cup <1> \cup <2> \cup <3>$ (we may have $<0> = <2>$ and $<1> = <3>$). Hence those graphs can be split into two categories:

- *type RLRL:* for any vertices $i, i+1$ with $i \in V$, we have $i \in L$ and $i+1 \in R$, or $i \in R$ and $i+1 \in L$. In this case, we always assume without loss of generality that $R = <0> \cup <2>$ and $L = <1> \cup <3>$.
- *type RRLL:* there exist two vertices $i, i+1$ with $i \in V$ such that $i, i+1 \in L$ or $i, i+1 \in R$. In this case, we always assume without loss of generality that $R = <0> \cup <1>$ and $L = <2> \cup <3>$.

We can finally show that, given a Hamiltonian cubic graph G, if G has no good shift, then there exist an almost good shift P in G (Lemma 5.23) and a vertex $v \in P$ such that $P \setminus \{v\}$ is a community (Proposition 5.24 and Theorem 5.25).

Lemma 5.23. *Let $G = (V, E)$ be a Hamiltonian cubic graph with no good shift and $k := \lceil \frac{|V|-1}{3} \rceil$. Then there exist an almost good shift $P := \{u, u+1, u+2, \cdots, u-k\}$, $u \in V$, of size $|V| - k + 1$ such that $d_P(v) \geq 2$ for all $v \in P$, and $u+1 \in L$ and $u-k-1 \in R$.*

Proof. Let $d := \gcd(k+1, |V|)$. Since G has no good shift, according to Lemma 5.21 and Corollary 5.22, $d \in \{2, 4\}$ and $|V| = 3k-1$ or $|V| = 3k+1$. From Corollary 5.19, we know that each vertex in V belong to either L or R.

- Case 1: *G is of type RLRL.* Since $|V|$ is even, then $|P|$ is even. Therefore, since two vertices $i, i+1 \in P$ do not both belong to L or R, then the vertex $-k$ belongs to L. Then the set $P := \{0, 1, \cdots, -k\}$ fulfills the requirements.
- Case 2: *G is of type RRLL.* Consider the set $P := \{1, 2, \cdots, -k+1\}$. According to Lemma 5.21, since $d = 4$, $|V| = 3k-1$. Hence, $-k+1 = 2-(k+1) \in <2>$. Thus $-k+1 \in L$ and P fulfills the requirements.

□

Recall that the graphs H_1 and H_2 with 8 vertices showed in Figure 5.4 have no community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor = 5$. We show in Theorem 5.25 that they are the only cubic Hamiltonian graphs in which there is no community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$.

Before proving the main theorem, we first deal with small graphs ($|V| < 20$) that are particular cases that need to be treated independently.

Proposition 5.24. *Let $G = (V, E)$ be a Hamiltonian cubic graph not isomorphic to H_1 or H_2 with $|V| < 20$. Then there is a community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$ in G.*

Proof. Let $k = \lceil \frac{|V|-1}{3} \rceil$. Since G is cubic, its number of vertices is even. From Lemma 5.21, $\gcd(k+1, |V|) \in \{1, 2, 3, 4\}$. If $\gcd(k+1, |V|) \in \{1, 3\}$, then there exists a good shift from Corollary 5.22. We suppose then that $\gcd(k+1, |V|) \in \{2, 4\}$. The following cases remain:

- If $|V| = 4$, then G is the complete graph K_4, and any set of 3 vertices is a community of size $\lfloor \frac{2 \cdot 4+1}{3} \rfloor$.
- If $|V| = 8$, we claim that G must have a good shift. By contradiction, suppose that G has no good shift. If G is of *type RRLL* then G is isomorphic to H_1, and if G is of *type RLRL* then G is isomorphic to H_2, which is impossible since we assumed that G is not isomorphic to H_1 or H_2.
- If $|V| = 10$, if G has no good shift, since $gcd(k+1, |V|) = 2$, G is necessarily of *type RLRL* and $c(0) = 3$, $c(1) = 8$, $c(2) = 5$, $c(4) = 7$, $c(6) = 9$. In this case, $V \setminus \{0, 6, 9\}$ is a community of size $\lfloor \frac{2 \cdot 10+1}{3} \rfloor$.
- If $|V| = 14$, if G has no good shift, since $gcd(k+1, |V|) = 2$, then G is necessarily of *type RLRL*. Following Lemma 5.23, let $P := \{0, 1, \cdots, 9\}$ be an almost good shift and:
  - If $c(3) \neq 0$, notice that $c(2), c(4) \in P$ (since $2, 4 \in R$) and $c(3) \in V \setminus P$. Thus $P \setminus \{3\}$ is a community of size $\lfloor \frac{2 \cdot 14+1}{3} \rfloor$.
  - If $c(6) \neq 9$, notice that $c(7), c(5) \in P$ (since $5, 7 \in L$) and $c(6) \in V \setminus P$. Thus $P \setminus \{6\}$ is a community of size $\lfloor \frac{2 \cdot 14+1}{3} \rfloor$.
  - If $c(3) = 0$ and $c(6) = 9$, notice that $c(3) \in P$, $c(5) \in P$ and $d_P(c(4)) = 3$ since $c(4) \neq 9$). Thus $P \setminus \{4\}$ is a community of size $\lfloor \frac{2 \cdot 14+1}{3} \rfloor$.
- If $|V| = 16$, if G has no good shift, since $gcd(k+1, |V|) = 2$, G is necessarily of *type RLRL*. Following Lemma 5.23, let $P := (0, 1, \cdots, -k)$ be an almost good shift. Since $0 \in R$, we have either $c(0) = 3$ or $c(0) = 5$. In each case, the graph is completely determined due to the constraints. In the first case, $P \setminus \{4\}$ is a community of size $\lfloor \frac{2 \cdot 16+1}{3} \rfloor$. In the second case, $P \setminus \{3\}$ is a community of size $\lfloor \frac{2 \cdot 16+1}{3} \rfloor$.

In each case, if G is not isomorphic to H_1 or H_2, then either G has a good shift which is a community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$, or we give a community of such size. □

Theorem 5.25. *Let $G = (V, E)$ be a Hamiltonian cubic graph not isomorphic to H_1 or H_2. Then there is a community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$ in G.*

Proof. If $|V| < 20$, then there is a community of size $\lfloor \frac{2 \cdot |V|+1}{3} \rfloor$ in G from Proposition 5.24. Now we suppose that $|V| \geq 20$, which implies that $k := \lceil \frac{|V|-1}{3} \rceil \geq 7$.

From Lemma 5.21, $\gcd(k+1, |V|) \in \{1, 2, 3, 4\}$. If $\gcd(k+1, |V|) \in \{1, 3\}$, then there exists a good shift (Corollary 5.22).

We suppose that $\gcd(k+1, |V|) \in \{2, 4\}$. If G contains a good shift, then the proof is done. Hence, we assume that G has no good shift. We prove that given an almost good shift P, there exists a vertex $u^* \in P$ such that $P \setminus \{u^*\}$ is a community. Observe that such vertex u^* exists if and only if $c(u^*-1), c(u^*+1) \in P$, and either $c(u^*) \in V \setminus P$ or $d_P(c(u^*)) = 3$.

- If G is of *type RLRL*, then $R = < 0 > \cup < 2 >$ and $L = < 1 > \cup < 3 >$. According to Lemma 5.23, the set $P := \{0, 1, 2, \cdots, -k\}$ is an almost good shift and $0 \in R, 1 \in L$. Since $2 \in R$ and $4 \in R$, then $c(2) \in P$ and $c(4) \in P$. If $c(3) \neq 0$, then $c(3) \in V \setminus P$ since $3 \in L$. Thus, $P \setminus \{3\}$ is a community of size $\lfloor \frac{2\cdot|V|+1}{3} \rfloor$. Symmetrically, if $c(-k-3) \neq -k$, then $c(-k-3) \in V \setminus P$ since $3 \in R$. Thus, $P \setminus \{-k-3\}$ is a community of size $\lfloor \frac{2\cdot|V|+1}{3} \rfloor$. On the other hand, if $c(3) = 0$ and $c(-k-3) = -k$, then $c(k-1) \neq -k$ and $c(k-1) \in P$. Moreover, since $k-3 \in R$ then $c(k-3) \in P$. Therefore, $c(k-2) \in V \setminus$ or $d_P(c(k-2)) = 3$ (since $k \geq 7$, $k-2 \neq 3$ and $c(k-2) \neq 0$). Thus, $P \setminus \{k-2\}$ is a community of size $\lfloor \frac{2\cdot|V|+1}{3} \rfloor$.

- If G is of *type RRLL*, then $R = <0> \cup <1>$ and $L = <2> \cup <3>$. According to Lemma 5.23, the set $P := \{1, 2, \cdots, -k+1\}$ is an almost good shift and $1 \in R, 2 \in L, -k \in R, -k+1 \in L$. Since $k+1 \in < 0 >$ and $k+2 \in < 1 >$, we necessarily have $k-1, k \in L$ and $k+1, k+2 \in R$. In this case, notice that since $k \geq 7$, $\{k-3, k-2, k-1\} \in P$. Moreover, $k-3, k-2 \in R$, which implies $c(k-3), c(k-2) \in P$. We show that either $c(k-1) \in P$ or $c(k) \in P$. Suppose that $c(k) \notin P$. Then since $k \in L$, we have $c(k) = 0$. Since $k-1 \in L$, we have $c(k-1) \in \{-1, 0, 1, \cdots, k-3\}$. Since $0 = c(k)$ and $-1 \in L$, then $c(k-1) \neq -1$ and $c(k-1) \neq 0$. Thus $c(k-1) \in \{1, 2, ..., k-3\} \subset P$. Thus either $c(k-1) \in P$ or $c(k) \in P$. Now, if $c(k-1) \in P$, then since $c(k-3) \in P$, the set $P \setminus \{k-2\}$ is a community of size $\lfloor \frac{2\cdot|V|+1}{3} \rfloor$. Else, $c(k) \in P$ and then since $c(k-2) \in P$, the set $P \setminus \{k-1\}$ is a community of size $\lfloor \frac{2\cdot|V|+1}{3} \rfloor$.

□

From Theorem 5.25 and using Lemma 5.10, we obtain the following corollary:

Corollary 5.26. *Given a cubic graph $G = (V, E)$ and a Hamiltonian cycle of G, a community of maximum size can be found in linear time.*

5.6 Extension of a vertex subset into a community

Olsen proved that deciding whether a given vertex subset can be extended into a community structure in a graph is NP-complete [133]. In this section we prove that the same result holds for a community, *i.e.* to answer whether a given subset of vertices of a graph can be extended into a community is NP-complete, even on bipartite graphs.

Community Extension
Input: A graph $G = (V, E)$, a set $S \subset V$ in G.
Question: Does it exist $C \subset V$ such that $S \subset C$ and C is a community in G?

Lemma 5.27. *Let m, n and k be positive integers such that $1 \leq k < n - 1 \leq m$ and $\ell := m \cdot (n - k - 1) - k + 1$. Then $\frac{\ell+k-1}{m+\ell+k-1} = \frac{n-k-1}{n-k}$.*

Proof. $(n-k)\cdot(\ell+k-1) = (n-k-1)\cdot(\ell+k-1)+\ell+k-1 = (n-k-1)\cdot(\ell+k+1)+m\cdot(n-k-1) = (n-k-1)\cdot(m+\ell+k+1)$. □

Definition 5.28. *Let $G = (V, E)$ be a graph not isomorphic to a star with n vertices and m edges and $k \geq 2$ be an integer. We define Γ' such that the graph $G' = (V', E') = \Gamma'(G)$ is as follows (see Figure 5.5 for an illustration):*

- *$V' := L \cup M \cup N$, where L contains $\ell := m\cdot(n-k-1)-k+1$ vertices, $M := \{e : e \in E\}$ and $N := V$.*
- *for all $e \in M$ and all $u \in N$, the edge $(e, u) \in E'$ if and only if $u \notin e$;*
- *for all $e \in M$ and all $v \in L$, the edge $(e, v) \in E'$;*

Notice that in Definition 5.28, G' is bipartite as there are edges only between M and $L \cup N$. Obviously, the construction can be done in polynomial time.

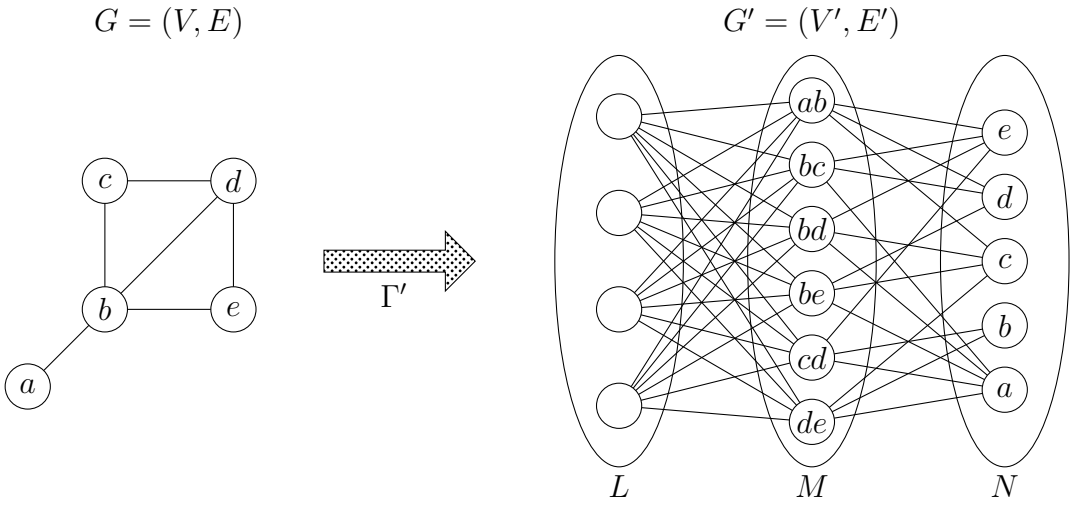


Figure 5.5: Example of the transformation Γ' with $k = 3$.

Theorem 5.29. Community Extension *is NP-complete even on bipartite graphs.*

Proof. Clearly, the problem is in NP. Let $G = (V, E)$ be a connected graph not isomorphic to a star with n vertices and m edges. Notice that if $k = |V| - 1$, as G is not isomorphic to a star, then there is no independent set of size $|V| - 1$ in G. For the given G and k, where $1 \leq k < |V| - 1$, let $G' = (V', E') = \Gamma'(G)$. We claim that G has an independent set of size at least k if and only if there exists a community C in G' such that $L \cup M \subset C$.

Let $R \subset V$ be an independent set of G of size $|R| = k' \geq k$. We claim that $C := L \cup M \cup R$ is a community in G'. Firstly, notice that since R is an independent set of G, then for any $e = (u, v) \in E$ at least one of u and v does not belong to R. Therefore, for any vertex $e \in M$, at least one $w \in N$ such that $(e, w) \notin E'$ does not belong to R, thus to C. Hence, we get that $\ell + k' - 1 \leq d_C(e) \leq \ell + k'$ and $d_{\overline{C}}(e) = d(e) - d_C(e) \leq n - k' - 1$. According to Lemma 5.27,

$$\frac{d_C(e)}{|C| - 1} \geq \frac{\ell + k' - 1}{m + \ell + k' - 1} \geq \frac{\ell + k - 1}{m + \ell + k - 1} = \frac{n - k - 1}{n - k} \geq \frac{n - k' - 1}{n - k'} \geq \frac{d_{\overline{C}}(e)}{|\overline{C}|},$$

we conclude that C is a community in G'.

Let C be a community in G' such that $L \cup M \subset C$, $L \cup M \neq C$. We claim that $R := C \cap N$ is an independent set of G of size at least k. Notice that since $L \cup M \subset C$ and $L \cup M \neq C$, then R is not empty. As it follows from Definition 5.28, a vertex $e \in M$ is not adjacent to exactly two vertices in N. Let u and v denote such two vertices. Since C is a community, the vertex e is satisfied if and only if at most one of u and v is in C. Thus at most one of the two endpoints u and v of the edge $e = (u, v) \in E$ is in R, hence R is an independent set. Now we prove that $|R| = k' \geq k$. Let $u \in R$ and $f \in M$ such that $u \in f$ in G, then $d_{\overline{C}}(f) \geq n - k' - 1$. If $d_{\overline{C}}(f) = n - k'$ then f is not satisfied in C. Therefore, $d_{\overline{C}}(f) = n - k' - 1$. Assume by contradiction that $k' < k$. Since C is a community and $|C| > 1$, we have

$$\frac{d_C(f)}{|C| - 1} \geq \frac{d_{\overline{C}}(f)}{|\overline{C}|} = \frac{n - k' - 1}{n - k'} > \frac{n - k - 1}{n - k}.$$

However, according to Lemma 5.27,

$$\frac{d_C(f)}{|C| - 1} = \frac{\ell + k' - 1}{m + \ell + k' - 1} < \frac{\ell + k - 1}{m + \ell + k - 1} = \frac{n - k - 1}{n - k}$$

which is a contradiction. Hence $k' \geq k$. □

In our reduction, the set $L \cup M$ can be a community or not, depending on the value of k and n. Indeed, $L \cup M$ is a community if and only if $\frac{\ell}{m+\ell-1} \geq \frac{n-2}{n}$ which directly implies $k \leq \frac{n}{2}$. Therefore, we stress that deciding if a community is inclusion-wise maximal is co-NP-complete.

Corollary 5.30. *Let $G = (V, E)$ be a connected graph and $C \subset V$ a community in G. Deciding if C is inclusion-wise maximal is co-NP-complete on bipartite graphs.*

In the following we show that the class of graphs from Figure 5.3 has a set of only 5 vertices that cannot be extended into a community (see Figure 5.6).

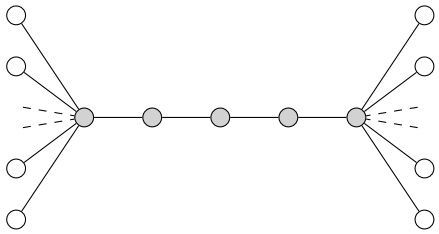


Figure 5.6: A caterpillar $T = (V, E)$ where the 5 vertices in gray cannot be extended into a community.

Proposition 5.31. *Let $G = (V, E)$ be a graph obtained by joining the center of two stars S_d for some integer d by a path of length 2. Let P denote the path on 5 vertices connecting the two centers of the stars. Then, there is no community $C \subset V$ in G such that $P \subseteq C$.*

Proof. We denote by z and z' the center vertices of the stars S and S' respectively. Firstly, notice that P is not a community since neither z nor z' is satisfied. Consider a set $C \subset V$ such that $P \subset C$. Let $\lambda := d_{\overline{C}}(z)$ and respectively $\lambda' := d_{\overline{C}}(z')$. Without loss of generality, assume that $\lambda \leq \lambda'$. We have $|C| = (d - \lambda + 1) + (d - \lambda' + 1) + 1 \geq 2 \cdot (d - \lambda' + 1) + 1$ and $|\overline{C}| = \lambda + \lambda' \leq 2\lambda'$. Suppose by contradiction that C is a community in G, then the vertex z' is satisfied and $2\lambda' \cdot (d - \lambda') \geq |\overline{C}| \cdot (d - \lambda') \geq (|C| - 1) \cdot \lambda' \geq 2\lambda' \cdot (d - \lambda' + 1)$, which is obviously impossible. Thus C is not a community. □

5.7 Conclusions

We proved that Max Community is NP-hard even on split graph, whether the community is required to be connected or not. In addition, the problem is APX-hard and 2-approximable in polynomial time. On the other hand, we proved that all Hamiltonian cubic graphs (except two) have a community with a size that reaches the theoretical bound $\lfloor \frac{|V| \cdot (\Delta(G) - 1) + 1}{\Delta(G)} \rfloor$, and such community can be found in polynomial time. Finally, we showed that determining if a community is inclusion-wise maximal is co-NP-complete, even on bipartite graphs.

Several questions remain open around this problem. It is not known in which graph classes a community that reaches the previous theoretical bound exists, and if so, if such community can be found in polynomial time. In particular, an extension of our theorem for Hamiltonian cubic graphs to general cubic graphs could be interesting to investigate. Furthermore, it could be interesting to investigate the problem in trees. From the approximation point of view for Max Community, we do not know if there is a better approximation ratio than 2, or a better inaproximation ratio than $(1 - \epsilon)$.

6 Clubs

Contents

The content of this chapter is based on the following article in preparation:

- C. Bazgan, P. Heggernes and T. Pontoizeau, *On the hardness of problems around s-clubs on split graphs*, in preparation.

6.1 Introduction

In this chapter, we investigate problems around s-clubs. In the context of community detection, the first intuitive way to define a cohesive group of people that constitutes a community is to look for a group of people where everybody knows each other, which corresponds to search for a clique in a graph.

However, as discussed in Chapter 3, considering communities as cliques is too restrictive: a subgraph with all possible internal edges except one would not be considered as a community under this assumption, even if it probably should be in real world social networks.

We can consider a less restrictive condition which still reflects cohesion. In this way, a community can be defined as a group of people such that every two members have a 'chain'

of relationship between them: the first person knows someone who knows someone ... who knows the second member, with a restricted length for this chain. Given a graph, Mokken introduced in [126] the notion of s-club which is a vertex set such that the subgraph induced by the vertex set has diameter at most s. Recent studies have been made around finding s-clubs in real networks [113, 127].

Problems around s-clubs have been well studied in the literature. A main one is, given a graph, to find an s-club of maximum size, which has been studied in [10, 38, 85, 93, 94, 147].

Another problem consists in finding a partition into k parts of vertices that are all s-clubs, that we discuss in this paper. This problem has been studied in [1, 37, 52, 138]. The problem has been showed linear time solvable in trees by Parley *et. al.* in [138]. In [52], Deogun *et. al.* proved that the problem is NP-hard for any $k \geq 3$ and $s \geq 2$ even in the case where the graph is both split and undirected path. Moreover, it is proved in [52] that for $s = 2$ the minimum number of parts is bounded by the domination number of a graph and is equal to the domination number on strongly chordal graphs. Abbas *et. al.* [1] proved that minimizing the number of parts in a partition of a graph into s-clubs is NP-hard for any $s \geq 2$ for bipartite graphs, NP-hard on chordal graphs, and also NP-hard for split graphs with $s = 2$ and any $k \geq 3$.

It is interesting to also study dynamic versions of community detection. In fact, real social network are constantly changing and links between members can either appear or disappear. In this way, we also study problems related to s-clubs around adding and removing edges. In [140], Plesnik studied the following problem: given a graph G, a cost function c on the edges and an integer B, finding a spanning subgraph G' of G with cost $\sum_{e \in G'} c(e) < B$ and with minimum diameter. The associated decision problem has been proved NP-complete. In [23], Biló *et al.* studied the two following problems. Given a graph $G = (V, E)$, and two positive integers D and B, find a minimum-cardinality set E' of edges to be added to G in such a way that the diameter of $G' = (V, E \cup E')$ is less than or equal to D. Given a graph $G = (V, E)$, find a set E' of B edges to be added to G in such a way that the diameter of $G' = (V, E \cup E')$ is minimized. Both are known to be NP-hard. Deleting at most t edges to a graph in order to obtain a graph of diameter at least s was proved NP-hard for $k = |E| - |V| + 1$ and $s = |V| - 1$ by Schoone *et al.* in [148].

The chapter is organized as follows. In Section 6.2, we introduce formally the problems around 2-clubs we studied. In Section 6.3, we discuss a wrong result from [37] and show that the problem of finding a partition into two 2-clubs is NP-hard, even in split graphs. In Section 6.4.1, we discuss the problem of adding a minimum number of edges in a graph in order to become of diameter at most 2. In Section 6.4.2, we study the problem of finding the minimum number of edges to keep in a graph while maintaining its diameter. Conclusion and open problems are given in Section 6.5.

6.2 Preliminaries

In this section we define the notions and the problems studied in this chapter. We are interested in the following decision problem :

In Section 6.3, we investigated the following problem:

k-Partition into s-clubs
Input : A graph $G = (V, E)$, two integers k, s.
Question : Is there a partition $\{P_1, P_2, \ldots, P_k\}$ of V such that P_i is an s-club, for each $i \in \{1, \ldots, k\}$?

In Section 6.4.1, we investigate the following problem:

s-Club Edges Adding
Input : A graph $G = (V, E)$, two integers s, t.
Question : Is there a set of edges E' of size at most t such that V is an s-club in the graph $G' = (V, E \cup E')$?

In Section 6.4.2, we investigated the following problem:

Spanning s-club
Input : A graph $G = (V, E)$, an integer k.
Question : Is there a set of edges $E' \subset E$ of size at most k such that the graph $G' = (V, E')$ is of diameter s ?

In order to prove some NP-hardness results, we use the following problem proved NP-hard in [77]:

Dominating Set
Input : A graph $G = (V, E)$, an integer t.
Question : Is there a set of vertices $S \subset V$ of size at most t such that S is a dominating set ?

Since we have several results in this class of graphs, we introduce some notation for an easier reading. For any split graph $G = (V = S \cup K, E)$, S always corresponds to the independent set and K to the clique. Moreover, for any dominating set D of G, we always consider that D is included in K. Indeed, if a vertex v of D belongs to S, we can take any neighbor v' of v from K in D instead of v from S without compromising the fact that D is a dominating set.

6.3 Partition into two 2-clubs

In this section, we first discuss that for bipartite graphs, partitioning a graph into two 2-clubs can be done in polynomial time whereas partitioning a graph into k 2-clubs, for any fixed $k \geq 3$, is NP-hard. Then, we show that partitioning a graph into two 2-clubs is NP-hard even on split graphs.

Since a 2-club in a bipartite graph is a biclique, then for bipartite graphs, partitioning a graph into k 2-clubs is equivalent to partition a graph into k bicliques. Since partitioning a bipartite graph into two bicliques is polynomial time solvable, [GT15] in [77], and

partitioning a bipartite graph into k bicliques for any fixed $k \geq 3$ is NP-hard [69], we can conclude that that for bipartite graphs, partitioning a graph into two 2-clubs can be done in polynomial time whereas partitioning a graph into k 2-clubs, for any fixed $k \geq 3$ is NP-hard.

Now notice that an interesting approach to detect s-clubs in graphs is to consider the power graph of the initial graph (see [37, 85]). In [37], Chang *et al.* claimed that the minimum number of parts in a partition into 2-clubs in a graph G equals the minimum number of cliques in a partition into cliques in G^2. Actually, there exist graphs (even split graphs) in which there is no partition into two 2-clubs but the squared graph contains a partition into two cliques. We can consider the graph in Fig. 6.1 as an example. The vertices a_1 and b_1 are at distance 3, then if such partition exists in this graph, a_1 and b_1 must be in different parts. We can apply this reasoning for the vertices a_1 and b_2, a_1 and b_3 and conclude that b_1, b_2, b_3 are in the same part. By symetry, a_1, a_2, a_3 are also in the same part. Since the only common neighbor of a_2, a_3 and b_1, b_2 is c, c needs to be in both parts in order to have a partition into 2-clubs, which is impossible. Thus, there is no partition into two 2-clubs.

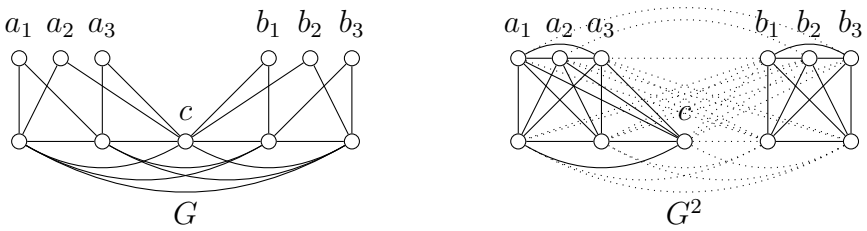


Figure 6.1: A split graph in which there is no partition into two 2-clubs but there is a partition into two cliques in the squared graph. Some edges are dotted in G^2 in order to highlight the partition into two cliques.

In order to prove that partitioning a graph into two 2-clubs is NP-hard on split graphs, we introduce Monotone 3-Sat that has been proved NP-hard in [77].

Monotone 3-Sat
Input : A set X of variables, a collection C of clauses over X which contains either only negated variables or only positive variables such that for each clause $c \in C$, $\|c\| = 3$.
Question : Is there a satisfying truth assignment for C ?

Theorem 6.1. 2-Partition into 2-clubs *is NP-hard even on split graphs.*

Proof. We reduce Monotone 3-Sat to 2-Partition into 2-clubs on split graphs. Let $I = (X, C)$ be an instance of Monotone 3-Sat with X the set of variables and C the set of clauses of size three, each clause being either positive or negative. We denote C_1 the

subset of C of positive clauses and C_0 the subset of C of negative ones. We define a split graph $G = (V = S \cup K, E)$ as an instance of 2-PARTITION INTO 2-CLUBS as follows (see Fig. 6.2).

For each positive clause c_i of C_1, introduce two vertices $c_{i,1}, c_{i,2}$ in a subset S_p. For each negative clause c_i of C_0, introduce two vertices $c_{i,1}, c_{i,2}$ in a subset S_n. For each variable $x_i \in X$ we introduce a vertex x_i in a subset K_x. Notice that $|K_x| = |X|$. All vertices $c_{i,1}, c_{i,2} \in S_p \cup S_n$ which correspond to a clause $c_i \in C$ are joined by an edge in E' to a vertex $x_i \in K_x$ that corresponds to a variable $x_i \in X$ either if it appears in a negative or a positive way in $c_i \in C$. Moreover, for each two vertices $c_{i,\ell}, c_{j,t}$ in S_p (resp. S_n) corresponding to two different positive clauses (resp. negative clauses), $i \neq j$ and $\ell, t \in \{1, 2\}$ such that both clauses do not contain any common variable, introduce a new vertex z in a subset K_p (resp. K_n) and introduce $c_{i,l}z, c_{j,t}z$ in E. Notice that $|K_p| = O(|C_1|^2)$ (resp. $|K_n| = O(|C_0|^2)$). Let S'_p (resp. S'_n) be a subset of $|K_p|$ (resp. $(|K_n|)$ vertices and introduce a perfect matching in E between K_p (resp. K_n) and S'_p (resp. S'_n). Moreover, for each couple $\{z_1, z_2\} \subset S_p \cup S'_p$ (resp. $S_n \cup S'_n$) such that they have no common neighbor in $K_p \cup K_x$ (resp. $K_n \cup K_x$), introduce an additional vertex w in a subset K'_p (resp. K'_n) and introduce $z_1w, z_2w \in E$. Notice that $|K'_p| = O(|C_1|^4)$ and $|K'_n| = O(|C_0|^4)$. Let $S = S_p \cup S_n \cup S'_p \cup S'_n$ and $K = K'_p \cup K_p \cup K_x \cup K_n \cup K'_n$ and join any two vertices of K by an edge in E so the subgraph induced by K becomes a clique, and we obtain a split graph $G = (S \cup K, E)$. Notice that $|S| = O(|C|^2)$ and $|K| = O(|C|^4 + |X|)$, then the size of the instance G is polynomial in $|X|$ and $|C|$.

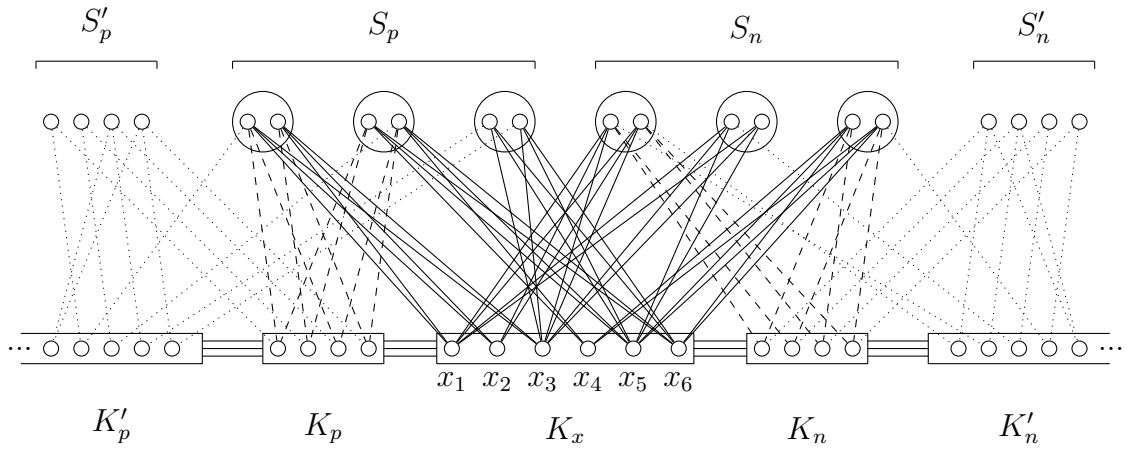


Figure 6.2: The split graph G defined from the instance $I = (X, C)$ with $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and $C = \{x_1 \vee x_2 \vee x_3, x_4 \vee x_5 \vee x_6, x_3 \vee x_5 \vee x_6, \overline{x_1} \vee \overline{x_2} \vee \overline{x_3}, \overline{x_1} \vee \overline{x_3} \vee \overline{x_5}, \overline{x_4} \vee \overline{x_5} \vee \overline{x_6}, \}$

Now we show that there is an assignment for the variables of X such that every clause of C is satisfied if and only if there is a partition of V into two 2-clubs.

Suppose that there is a assignment satisfying all clauses from C. Then, we define the

partition $\{V_1, V_0\}$ of V into two 2-clubs as follows. Let K_1 (resp. K_0) be the set of all vertices $x_i \in K_x$ such that the corresponding variables $x_i \in X$ have been assigned to true (resp. false). Define $V_1 = S'_p \cup S_p \cup K'_p \cup K_p \cup K_1$ (resp. $V_0 = S'_n \cup S_n \cup K'_n \cup K_n \cup K_0$). Now we show that for every pair of vertices $\{z_1, z_2\}$ in V_1, $d_{V_1}(z_1, z_2) \leq 2$. First, for any $z_1, z_2 \in V_1 \cap K$, $d_{V_1}(z_1, z_2) = 1$. Moreover, for any $z_1 \in V_1 \cap K$, $z_2 \in V'_1 \cap S$, $d_{V_1}(z_1, z_2) \leq 2$ since any vertex of S has a neighbor in K. Finally, for any $z_1, z_2 \in V_1 \cap S$, $d_{V_1}(z_1, z_2) = 2$ since z_1 and z_2 always have a common neighbor either in K'_p, K_p or K_1. Then, for any pair of vertices $\{z_1, z_2\}$ in V_1, $d_{V_1}(z_1, z_2) \leq 2$. The same reasoning can be done for V_0 by symmetry. Thus, $\{V_1, V_0\}$ is a partition of V' into two 2-clubs.

Suppose now that there is a partition $\{V_1, V_0\}$ of V such that V_1 and V_0 are 2-clubs. First notice that for any vertex $z_1 \in S'_p \cup S_p$ and any $z_2 \in S'_n$, $d(z_1, z_2) = 3$. Then we can assume without loss of generality that $(S'_p \cup S_p) \subset V_1$ and $S'_n \subset V_0$. This implies that $K'_p \cup K_p \subset V_1$ since any two vertices of $S'_p \cup S_p$ must have a common neighbor in V_1. By symmetry, we can also conclude that $S_n \cup S'_n \cup K'_n \cup K_n \subset V_0$. Then, we show that assigning all variables in X corresponding to the vertices in $K_x \cap V_1$ to true and all variables in X corresponding to the vertices in $K_x \cap V_0$ to false is an assignment satisfying all clauses in C. By contradiction, suppose that there is a clause in $c_i \in C$ which is false with respect to this assignment. Wlog we consider that the clause is a positive one. Then, all variables in c_i are assigned to false and the corresponding vertices in K_x are in C_0 whereas the two vertices $c_{i,1}, c_{i,2}$ in S_p corresponding to the clause c_i are in V_1. Then, by construction, $c_{i,1}$ and $c_{i,2}$ have no common neighbor in V_1 and thus V_1 is not a 2-club. Thus, the defined assignment for X satisfies all clauses in C. □

In [37], Chang *et al.* claimed that determining if the complement of the squared graph is bipartite allows to determine if there is a partition of a graph into two 2-clubs. Actually, this only allows to determine if there is a covering into two 2-clubs (*i.e.* a set $\{C_1, C_2\}$ of subsets of V such that $C_1 \cup C_2 = V$ and C_1, C_2 are 2-clubs). In fact, there is a covering into two 2-clubs if and only if the complement of the squared graph is bipartite. Indeed, if the complement of the squared graph is bipartite with the 2-partition (A, B), a covering into two 2-clubs can be determined by considering the partition (A, B) and for any two vertices which are not at distance 2 in the subgraph induced by one part, add a common neighbor of those two vertices into this part (without removing it from the original part). Notice that the existence of such bipartition ensures that such vertex always exists. In Figure 6.1, c would belong to both 2-clubs.

It can be observed that 2-Partition into 2-clubs is easy to solve on split graphs of diameter 2, since considering any vertex of the independent set in one part and the rest of the graph in the other part constitutes a 2-partition into two 2-clubs.

6.4 Edge editing

We now focus on problems in which edge adding or removal can ensure or maintain some distance between the vertices of the resulting subgraph.

6.4.1 Edge adding

The s-Club Edges Addition problem has been proved NP-hard for $s = 3$ by Schoone *et al.* in [148] and NP-hard for $s = 2$ by Li *et al.* in [118]. The case $s = 1$ is trivial since it corresponds to adding edges between every pair of nonadjacent vertices. Gao *et al.* [76] proved the W[2]-hardness of the problem for any $s \geq 2$ by establishing a reduction from Dominating Set.

We prove that 2-Club Edges Addition is W[2]-hard even on split graphs. Notice that even if a solution is not required to be a split graph, we show in Theorem 6.2 that it is always possible to obtain a split graph of diameter 2 with less edges than any solution.

Theorem 6.2. 2-Club Edges Adding *is W[2]-hard even on split graphs.*

Proof. We reduce Dominating Set on split graphs of diameter 2, which has been proved W[2]-hard by Lokshtanov *et. al.* in [119], to 2-Club Edges Addition on split graphs. Let $G = (V = S \cup K, E)$ be a split graph of diameter 2, instance of Dominating Set, where S corresponds to the independent set and K to the clique. We construct an instance $G' = (V', E')$ of 2-Clubs Edges Addition as follows. Consider a copy of G and add two new vertices s_0, k_0. The graph G' is a split graph with $V' = S' \cup K'$ where $S' = S \cup \{s_0\}$ and $K' = K \cup \{k_0\}$. Set E' is obtained from E by adding edges between k_0 and every vertex $v \in K$ and the edge $k_0 s_0$ (see Figure 6.3). We show now that there is a dominating set of size at most t in G if and only if we can add at most t edges to G' such that G' has diameter 2.

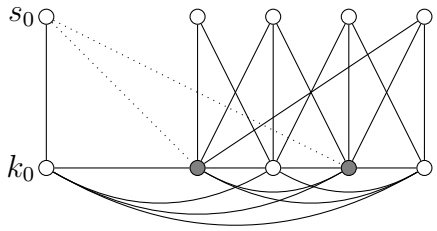


Figure 6.3: The graph G' and a set of edges (represented by dotted lines) of minimum size to add to make G' having diameter 2 (a minimum dominating set is given in gray).

Suppose that D is a dominating set of size t in G. We assume that $D \subseteq K$. The graph $(V', E' \cup D')$ where $D' = \{s_0 x : x \in D\}$ has diameter 2 since any vertex from S has a neighbor in D and then s_0 is at distance two from any vertex from S.

Suppose now that D' is a set of non edges of G' of size t such that $G'' = (V', E' \cup D')$ has diameter 2. We first show that we can assume that all edges in D' are between s_0 and K. Let $xy \in D'$. If $x = s_0$ and $y \in S$, let y' be a neighbor of y in K and the graph $G'' = (V', E' \cup D' \cup \{s_0 y'\} \setminus \{xy\})$ is still of diameter 2. If both x and y are different from s_0 and $x \in S, y \in K$, then the graph $G'' = (V', E' \cup D' \setminus \{xy\})$ is still of diameter 2 since G has diameter 2. If now both x and y are different from s_0 and $x \in S, y \in S$, then

the graph $G'' = (V', E' \cup D' \setminus \{xy\})$ is still of diameter 2 since G has diameter 2. Thus, we can assume that all edges in D' are between s_0 and K (by updating D' if necessary) and $|D'| \leq t$. Then, the set of vertices adjacent to s_0 by an edge in D' is necessarily a dominating set of G since s_0 and any vertex of S must be at distance 2. Thus, we obtain a dominating set in G of size at most t. □

6.4.2 Edge deletion

In this section, we discuss the hardness of Spanning s-club in split graphs and of Spanning s-club for any odd $s \geq 3$ in general graphs.

First, we observe that Spanning s-club is easy to solve for any $s \geq 4$ in split graphs. Indeed, for any split graph $G = (S \cup K, E)$, define the spanning tree $G' = (V', E')$ such that for any $v \in S$, choose a unique edge $e \in E$ adjacent to v in G and let e be in E'. Then choose any vertex $x \in K$ and for any $x' \in K$ such that $x \neq x'$, let xx' be in E'. The remaining graph G' has diameter 4 and the number of edges in E' is minimum since G' is a tree (see Figure 6.4).

Figure 6.4: A split graph and its spanning subtree of diameter 4

We believe that Spanning 3-club should be NP-hard in split graphs. A reduction could be done if the following conjecture is true:

Conjecture 6.3. *Let $G = (V = S \cup K, E)$ be a split graph and D be a minimum dominating set of G. Then, for any spanning subgraph $G' = (V, E')$ of G of diameter 3, we have $|E'| \geq \frac{|D|(|D|-1)}{2} + |V| - |D|$.*

Several observation makes us think that this conjecture is true. Let $G = (V = S \cup K, E)$ be a split graph and D be a minimum dominating set of G. Let $G' = (V, E')$ be a spanning subgraph of G of diameter 3.

First of all, consider any vertex $s \in S$. Since such vertex must be at distance 3 of any other vertex from S in the subgraph G', $\sum_{q \in N(s)} d(q) \geq |D|$. This shows that the number of edges in G' is strongly related to the size of the minimum dominating set.

Moreover, we can show that the conjecture is true if $|D| = |S|$. Indeed, if $|D| = |S|$ then any vertex d from D covers exactly one vertex in S. This implies that there is no s_1, s_2 in S which have a common neighbor in G since we would have $|D| < |S|$. Let $s_1, s_2, \ldots, s_{p+1}$ be the vertices of S. For each s_i, $1 \leq i \leq p+1$, we note S_i the set of

its neighbors in G'. Since G' has diameter 3, we know that any s_i, $1 \leq i \leq p+1$, must be at distance 3 from any other s_j, $j \neq i$, which implies that there must exist at least one edge in G' between any S_i and S_j, $i \neq j$. Thus, E' contains at least $\frac{|S|(|S|-1)}{2}$ edges between all S_i and S_j, $i \neq j$. Moreover, E' also contains all edges between $\{s_i\}$ and S_i for $1 \leq i \leq p+1$ by definition of S_i. Finally, any vertex from K which has no neighbor in S must have at least one adjacent edge in E' since G' must be connected. Thus, we obtain $|E'| \geq \frac{|S|(|S|-1)}{2} + \sum_{j=1}^{p+1} |S_j| + |K| - \sum_{j=1}^{p+1} |S_j| = \frac{|S|(|S|-1)}{2} + |K| = \frac{|D|(|D|-1)}{2} + |V| - |D|$, since $|D| = |S|$.

Showing this conjecture would allow us to prove the following:

Proposition 6.4. *If Conjecture 6.3 is true, then* Spanning 3-club *is* NP*-complete even on split graphs.*

Proof. Suppose that Conjecture 6.3 is true. We reduce Dominating Set on split graphs, which has been proved NP-complete in [22], to Spanning 3-club on split graphs. Notice that Spanning 3-club is obviously in NP. Let $G = (V = S \cup K, E)$ be a split graph as an instance of Dominating Set, where S corresponds to the independent set and K to the clique, and we also consider G as an instance of Spanning 3-club. We show that there is a dominating set of size at most b in G if and only if there is a spanning subgraph containing at most $\frac{b(b-1)}{2} + |V| - b$ in which V is a 3-club.

Let D be a dominating set of size b in G, $D \subseteq K$. Define the following spanning subgraph $G' = (V, E')$ with $E' \subset E$. For each pair of vertices $\{x, y\}$ in D, xy belongs to E'. Moreover, for each $v \notin D$, choose any edge vx with $x \in D$ in E'. Notice that $|E'| = \frac{b(b-1)}{2} + |V| - b$ and it is easy to see that G' is a spanning subgraph of diameter 3.

Now let $G' = (V, E')$ be a spanning subgraph of G of diameter 3 such that $|E'| \leq \frac{b(b-1)}{2} + |V| - b$ for some integer b and let D be a minimum dominating set of G. From Conjecture 6.3, $|E'| \geq \frac{|D|(|D|-1)}{2} + |V| - |D|$. Thus, $b(b-3) \geq |D|(|D|-3)$. Since $b, |D| \geq 1$, we obtain $b \geq |D|$. Thus, there exists a dominating set of size less than b in G. □

In order to go a little further, it is now interesting to notice that the latter conjecture would imply that Spanning $(2s+1)$-club is NP-complete in general graphs for any $s \geq 1$. In order to prove that, we introduce the concept of *graph almost split of length s*. For any $s \geq 1$, a graph G is said to be almost split of length s if we can construct G by considering a split graph and for each vertex of G we add a path of length $s-1$ linked to this vertex (see Figure 6.5). For a given graph almost split of length s, we note S_i the set of all vertices from the added paths which are at distance i from a vertex of the induced split graph. Notice that a split graph is a graph almost split of length 0.

For any $s \geq 1$, we reduce Spanning $(2s+1)$-club in graphs almost split of length $s-1$ to Spanning $(2s+3)$-club in graphs almost split of length s.

Proposition 6.5. *If Conjecture 6.3 is true, then* Spanning $(2s+1)$-club *is* NP*-complete on general graphs for any* $s \geq 1$.

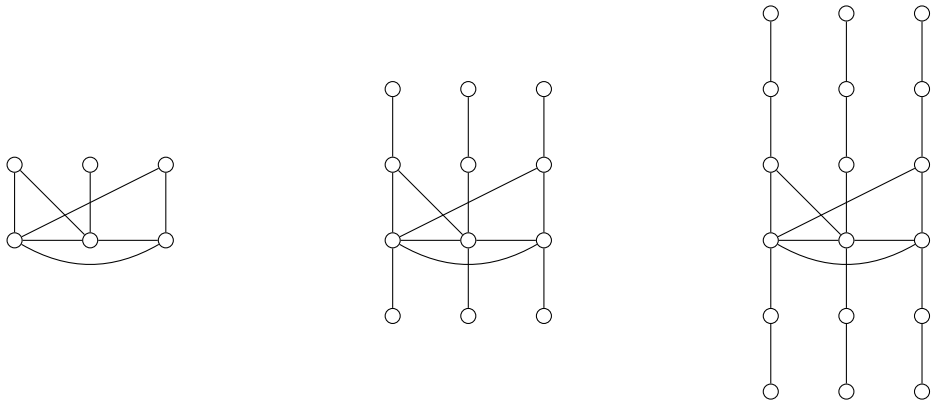


Figure 6.5: A split graph, a graph almost split of length 1 and a graph almost split of length 2

Proof. We suppose that Conjecture 6.4 is true. Let $s \geq 1$ be any integer. Since Spanning 3-club is supposed to be NP-complete from Conjecture 6.4, we reduce Spanning $(2s+1)$-club in graphs almost split of length $s-1$ to Spanning $(2s+3)$-club in graphs almost split of length s for any $s \geq 1$ and show the property by polynomial reduction transitivity.

Let $G = (V, E)$ be an instance of Spanning $(2s+1)$-club in graphs almost split of length $s-1$ where $V := S_0 \cup S_1 \cup \cdots \cup S_{s-1} \cup K$. We construct an instance $G' = (V', E' \cup E'')$ of Min spanning $(2s+3)$-club in graphs almost split of length s where $V' := S'_0 \cup S'_1 \cup \cdots \cup S'_s \cup K'$ as follows. For any $i \in \{1, \ldots, s-1\}$, let S'_i be a copy of S_i. Let K' be a copy of K. There is an edge in E between two vertices of $S_0 \cup S_1 \cup \cdots \cup S_{s-1} \cup K$ if and only if there is an edge in E' between the corresponding copies in $S'_0 \cup S'_1 \cup \cdots \cup S'_{s-1} \cup K'$. Finally, let S'_s be a set of $|V|$ vertices, and let E'' be a perfect matching between S'_s and S'_{s-1}.

We show now that there is a spanning subgraph with k edges in G of diameter $2s+1$ if and only if there is a spanning subgraph with $k + |S_0|$ edges in G' of diameter $2s+3$. Obviously, if G has a spanning subgraph of diameter $2s+1$ with k edges, considering the corresponding edges in G' and adding all edges from E'' between S'_{s-1} and S'_s, we obtain a spanning subgraph of diameter $2s+3$. Now suppose that G' has a spanning subgraph G'_0 of diameter $2s+3$ with $k+|S_0|$ edges. Since any vertex from S'_s must have an adjacent edge in G'_0 (otherwise the subgraph would not be connected), the spanning subgraph contains k edges between two vertices of $V' \setminus S'_s$. Consider the spanning subgraph G_0 of G containing those corresponding k edges. Since G'_0 has diameter $2s+3$, any two vertices from S'_{s-1} must be at distance $2s+1$, and then G_0 has diameter $2s+1$. □

We believe that Spanning 2-club must also be NP-complete. We give the following conjecture:

Conjecture 6.6. Spanning 2-club *is NP-complete even on split graphs.*

Using a similar reduction from Proposition 6.5, assuming that Conjecture 6.6, we can derive the following proposition:

Proposition 6.7. *If Conjecture 6.3 is true, then* Spanning $(2s)$-club *is* NP*-complete on general graphs for any* $s \geq 2$.

In that way, we obtain the following proposition:

Proposition 6.8. *If Conjectures 6.3 and 6.6 are true, then* Spanning s-club *is* NP*-complete on general graphs for any* $s \geq 2$.

6.5 Conclusions

In this chapter, we proved that partitioning a graph into two 2-clubs is NP-hard, even on split graphs. Moreover, we proved that 2-Club Edges Adding is W[2]-hard even on split graphs. We also discussed the possible NP-hardness of Spanning 2-club and Spanning 3-club on split graphs, and of Spanning s-club on general graphs for any integer $s \geq 1$.

On the other hand, in addition to our conjectures, some open questions remain open. In Section 6.3, we saw that partitioning a graph into two 2-clubs is harder than expected in [37]. However, we would expect that a graph of diameter 2 has always such a partition, but this problem remains open. On the other hand, investigating the complexity of 2-club Adding Edges and Spanning 2-club in bipartite graphs would give a better understanding of s-clubs in graphs.

7

Independent 2-cliques

Contents

The content of this chapter is based on the following papers [17, 18]:

- C. Bazgan, T. Pontoizeau, Z. Tuza. *On the Complexity of Finding a Potential Community*. The 10th International Conference on Algorithms and Complexity (CIAC 2017), LNCS 10236, pages 80-91, 2017.

- C. Bazgan, T. Pontoizeau, Z. Tuza. *Finding a potential community in networks*. Theoretical Computer Science, accepted.

7.1 Introduction

With the recent development of social networks and particularly online meet-up services like Couchsurfing or Meetup.com, it could be interesting to investigate the detection of some group of people who do not know each other, but are related by their other relationships. Such a group could be considered as a 'potential' community since it does not form a community in the first place, but could become one due to their proximity. This may find various applications in online dating and meet-up services in which members expect not to know the other members.

More precisely, considering a graph G, we want to define potential communities by looking at independent sets in which any two members are related within a specified distance in G. Contrary to a k-club, the distance between two vertices must be realized via vertices outside of the subgraph. We call such a subset of vertices an independent k-clique, where k is the largest distance between vertices of S in the original graph. In this paper, we study the problem of finding an independent 2-clique of maximum size.

We investigate the complexity of the problem in several graph classes. Since this problem is close to finding an independent set of maximum size, we also compare the hardness of the two problems. Figure 7.1 summarizes the results we prove in this chapter.

This chapter is structured as follows. In Section 7.2 we introduce formally some notation and definitions. In Section 7.3 we show that the complexity of Max Independent 2-Clique jumps from polynomial-time solvable to NP-hard when the input class is extended from planar graphs to apex graphs. In Section 7.4 we present polynomial algorithms to solve Max Independent 2-Clique in some graph classes. In Section 7.5 we show NP-hardness and non-approximability of Max Independent 2-Clique in some other graph classes. Conclusions and open problems are given in Section 7.6.

7.2 Preliminaries

An *independent 2-clique* is defined as a subset of vertices which is an independent set and a 2-clique at the same time. We recall from Chapter 3 that a 2-clique is a subset of vertices in which any two vertices are at distance at most 2 in the graph.

In this paper we are interested in the following optimization problem:

Max Independent 2-Clique
Input: A graph $G = (V, E)$.
Output: A subset $S \subset V$ which is an independent 2-clique of maximum size.

The Max Independent 2-Clique problem is closely related to the well known problem of finding an independent set of maximum size, named Max Independent Set.

Given a graph G, the standard notation for the maximum size of an independent set in G is $\alpha(G)$. The maximum number of vertices in an independent 2-clique of G will be denoted by $\alpha_{=2}(G)$. The subscript '=2' intends to express that the distance between any two vertices of the independent set is exactly 2.

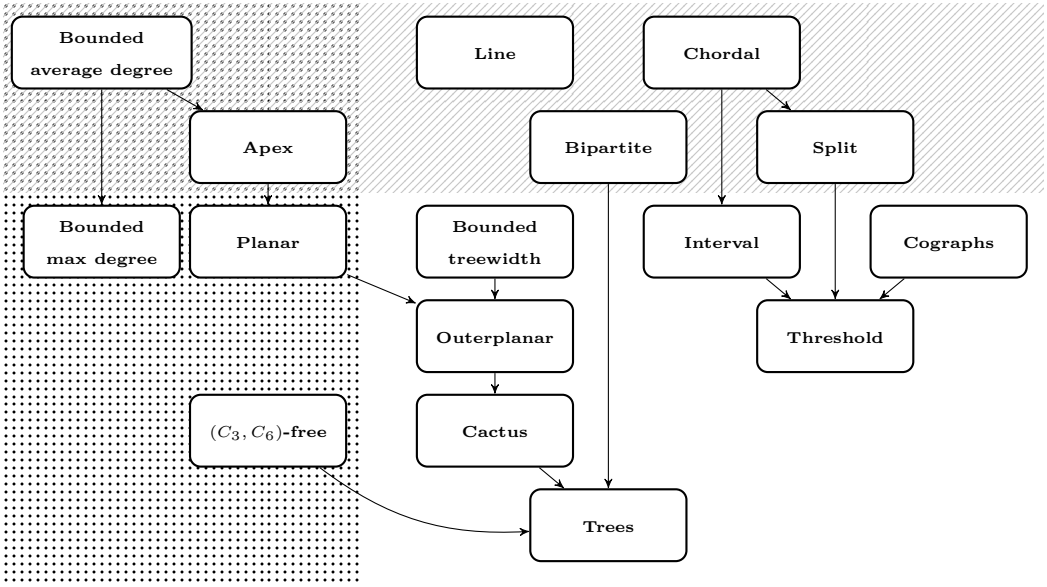


Figure 7.1: Relationship among some classes of (connected) graphs, where an arrow from a class to another indicates that the first class contains the second one. We compare the hardness of Max Independent 2-Clique and Max Independent Set in studied graph classes. Max Independent 2-Clique is NP-hard on graph classes at the top of the figure (hatched area) and is polynomial-time solvable on graph classes at the bottom (non-hatched area). Max Independent Set is NP-hard on graph classes on the left of the figure (dotted area) and is polynomial-time solvable on graph classes on the right (non-dotted area).

Note that $\alpha_{=2}(G) \geq 2$ whenever at least one connected component of G is not a complete graph. Indeed, any such component contains two vertices at distance exactly two, hence forming an independent 2-clique of size 2. Moreover, if G is disconnected and has components $G_1, \ldots, G_k$ then

$$\alpha_{=2}(G) = \max_{1 \leq i \leq k} \alpha_{=2}(G_i)$$

For these reasons we assume in this chapter that G is a non-complete, connected graph (although some of the algorithms also need to handle disconnected graphs temporarily).

Independent 2-cliques might have several forms. Indeed, the 2-clique property of an independent 2-clique S can be ensured either by only one vertex or a lot: if each pair of vertices have a common neighbor that is different for every pair, the number of vertices out of S that are useful to insure the 2-clique property can reach $\frac{|S| \cdot (|S|-1)}{2}$. See Figure 7.2 as an illustration.

From the parametrized complexity point of view, it is interesting to notice the following fact.

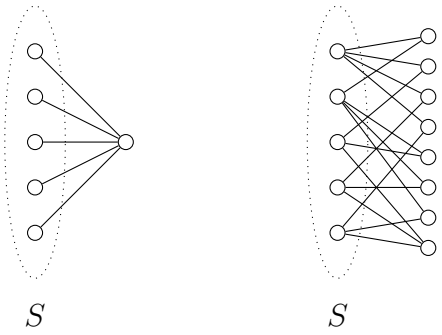


Figure 7.2: Two graphs in which S is an independent 2-clique.

Theorem 7.1. Max Independent 2-Clique *belongs to* $W[1]$ *in general graphs.*

Proof. We construct an FPT-reduction from Max Independent 2-Clique to Max Clique. Let $G = (V, E)$ be an instance of Max Independent 2-Clique. We construct an instance of Max Clique by considering the graph $G' = (V, E')$ in which $xy \in E'$ if and only if x and y are exactly at distance 2 in G. It is easy to see that there is an independent 2-clique of size k in G if and only if there is a clique of size k in G'. Since Max Clique belongs to $W[1]$ [53], Max Independent 2-Clique also belongs to $W[1]$. □

7.3 Complexity jump from planar graphs to apex graphs

According to [78], Max Independent Set is known to be NP-hard in planar graphs, and thus also in apex graphs. On the other hand, we prove that Max Independent 2-Clique is polynomial-time solvable on planar graphs but NP-hard on apex graphs. This shows that inserting or removing a single vertex in a graph may dramatically change the complexity of Max Independent 2-Clique.

Theorem 7.2. Max Independent 2-Clique *is NP-hard on apex graphs.*

Proof. We establish a polynomial reduction from Max Independent Set on cubic planar graphs, which is proved to be NP-hard in [78], to Max Independent 2-Clique on apex graphs. Let $G = (V, E)$ be a cubic planar graph, an instance of Max Independent Set. The instance $G' = (V', E')$ of Max Independent 2-Clique is defined by inserting an additional vertex z that is adjacent to every vertex of V. It is easy to see that $\{z\}$ itself is a one-element non-extendable independent 2-clique, while the independent 2-cliques of G' not containing z are precisely the independent sets of G. □

Theorem 7.2 implies another interesting result:

Corollary 7.3. Max Independent 2-Clique *is NP-hard on the class of graphs of average degree at most 5.*

Proof. Cubic graphs on n vertices have $3n/2$ edges, thus the graph constructed in the proof of Theorem 7.2 is of order $n+1$ and has $5n/2$ edges, yielding average degree less than 5. □

In order to prove that Max Independent 2-Clique is polynomial-time solvable on planar graphs, we use a famous theorem introduced by Courcelle in [46] which states that any problem expressible in Monadic Second-Order Logic is linear-time solvable for graphs of bounded treewidth. This allows to show first the following:

Theorem 7.4. Max Independent 2-Clique *is linear-time solvable on graphs with bounded treewidth.*

Proof. We observe that the problem is expressible in Monadic Second-Order Logic:

$$maxI2C(S) := max_S\{|S| : \forall x \forall y(Sx \wedge Sy) \rightarrow (\neg edg(x,y) \wedge (\exists z, edg(x,z) \wedge edg(y,z)))\}$$

Since any problem expressible in Monadic Second-Order Logic is linear-time solvable for graphs of bounded treewidth (see [46]), $\alpha_{=2}$ can be determined in linear time in graphs of bounded treewidth. □

Based on this result, we prove the following result.

Theorem 7.5. Max Independent 2-Clique *is polynomial-time solvable on planar graphs.*

Proof. Let $G = (V, E)$ be a planar graph and $v \in V$ any vertex. Then all the other vertices in an independent 2-clique S containing v are at distance exactly 2 apart from v. Further, the 2-clique property for $S \setminus \{v\}$ is ensured by vertices within distance at most 3 from v. Thus, the vertices relevant for S to be an independent 2-clique induce a subgraph G' in G such that G' belongs to the class of '4-outerplanar' graphs. Graphs which are 4-outerplanar have treewidth at most 11 (more generally, all k-outerplanar graphs have treewidth at most $3k-1$, due to [25]). Then, using Theorem 7.4, a polynomial-time algorithm for Max Independent 2-Clique in planar graphs consists in solving the problem for all subgraphs G' (which have treewidth at most 11) defined from each vertex v of G and choose a solution of maximum size. □

Concerning the parameterized complexity, we can show the following.

Theorem 7.6. *The parameterized problem associated with* Max Independent 2-Clique *with the natural parameter is in* FPT *on apex graphs.*

Proof. Let $G = (V, E)$ be an apex graph and $x \in V$ a vertex such that $G - x$ is planar. Since any planar graph is 4-colorable [106], the size of an independent set in $G[N(x)]$ is at least $\frac{|N(x)|}{4}$, and so is the size of an independent 2-clique in G. Thus, considering the parameter k, if $|N(x)| \geq 4k$, then the answer is yes.

If now $|N(x)| < 4k$, as discussed in the previous proof, considering any vertex v belonging to an independent 2-clique S in G, the 2-clique property for $S \setminus \{v\}$ is ensured

by vertices within distance at most 3 from v in $G - x$. Then, in G, considering any vertex v belonging to an independent 2-clique S, the 2-clique property is ensured by vertices within distance at most 3 from v in $V \setminus \{x\}$ and x and in its neighborhood $N(x)$. For this reason, for each vertex $v \in V$, we consider the subgraph induced by the set of all vertices at distance at most 3 from v and include $\{x\} \cup N(x)$. This subgraph has treewidth at most $12 + |N(x)| < 12 + 4k$. Since any problem parameterized by q expressible in Monadic Second-Order Logic is in FPT with respect to q on graphs of treewidth bounded by q [47], a polynomial-time algorithm can be designed by solving the problem for all such subgraphs defined from each vertex v of G, and answer yes if at least one such problem answers yes and answer no otherwise. □

7.4 Graph classes with polynomial-time algorithms

In the following we identify some graph classes on which Max Independent 2-Clique is computable in polynomial time, while Max Independent Set is not always polynomial-time solvable.

7.4.1 Graph classes related to the degree

First, it is interesting to notice that, according to the next propositions, Max Independent 2-Clique is polynomial-time solvable on graphs of bounded degree and also on complements of graphs of bounded degree, while Max Independent Set is NP-hard on graphs of bounded degree [78] but polynomial-time solvable on their complements (using exhaustive search in the non-neighborhood of each vertex, which can be done in linear time).

Proposition 7.7. Max Independent 2-Clique *is linear-time solvable on graphs with bounded maximum degree.*

Proof. The proof consists in computing, for each vertex v of a graph $G = (V, E)$, the largest size of an independent 2-clique v can belong to. Since the maximum degree is bounded, also the number of vertices at distance 2 from v is bounded, thus the largest independent 2-clique among them can be determined in constant time. Performing this for all vertices of the graph can be done in $O(|V|)$ steps. □

Proposition 7.8. Max Independent 2-Clique *is linear-time solvable on graphs of minimum degree at least* $(n - d)$*, where* d *is constant.*

Proof. Since every vertex is non-adjacent with fewer than d vertices, the size of a solution cannot exceed d. Then using an exhaustive search in the non-neighborhood of each vertex, we can find an optimal solution in linear time. □

7.4.2 Finding an independent 2-clique in the neighborhood of a vertex

Now, notice that a natural way to find an independent 2-clique is to take an independent set included in the neighborhood of one vertex. First, this principle can be applied easily on trees.

Proposition 7.9. *Every tree T satisfies $\alpha_{=2}(T) = \Delta(T)$. Thus,* Max Independent 2-Clique *is linear-time solvable on trees without using Monadic Logic.*

Proof. Any two vertices v, w of an independent 2-clique S share a neighbor, say u, which is unique in any tree. Non-neighbors of u cannot belong to S because they are at distance at least 3 from v or w (or both). On the other hand, all neighbors of u have mutual distance 2, so that $|S|$ is largest if S is the neighborhood of a vertex of maximum degree. □

In this way, it is interesting to investigate the properties of a graph in which an independent 2-clique is not included in the neighborhood of one vertex. We show in Lemma 7.10 that such a graph necessarily contains a cycle of length 3 or 6, and cannot be a cactus if such an independent 2-clique has a certain size. Such properties allow us to get an easy polynomial-time algorithm for Max Independent 2-Clique on (C_3, C_6)-free graphs, while Max Independent Set is NP-hard[1] on this class of graphs (see [5]). From Theorem 7.4 we already know that Max Independent 2-Clique is linear-time solvable on cactus graphs, but the property of Lemma 7.10 allows us to give a simpler algorithm for this class of graphs.

Lemma 7.10. *Let $G = (V, E)$ be a graph. Suppose that there exists an independent 2-clique S not contained in the neighborhood of a single vertex. Then G contains an induced cycle of length 3 or 6. Moreover, if $|S| \geq 4$, G is not a cactus.*

Proof. Let S be an independent 2-clique in G such that all vertices of S do not have a common neighbor. Let u be a vertex in $V \setminus S$ which has the maximum number of neighbors in S, and N_u be the neighborhood of u in S. Then there exists a vertex z in S which is not a neighbor of u. Let v be any vertex of N_u, and w be a common neighbor of z and v. Let v' be a vertex in N_u non-adjacent to w (it exists by the choice of u). Since S is a 2-clique, v' and z have a common neighbor, say w' (notice that w' can be neither u nor w). Thus, $C := (u, v, w, z, w', v', u)$ is a cycle in G (see Figure 7.3).

If C has no chord, then it is an induced 6-cycle of G; and otherwise any chord of C lies inside $\{u, w, w'\}$ and thus it creates a 3-cycle in G. This proves the first assertion.

Suppose now that $|S| \geq 4$. Then there are three options:

- u has only two neighbors in S. Then any two vertices of S must have a different common neighbor in $V \setminus S$ (by the choice of u), moreover there exists z' in $S \setminus \{N_u, z\}$.

[1] It is proved in [5] that for a finite set H of connected graphs, Max Independent Set is NP-hard on the class of H-free graphs if no member of H is either a path or a tree with one vertex of degree 3 and the other vertices of degree at most 2.

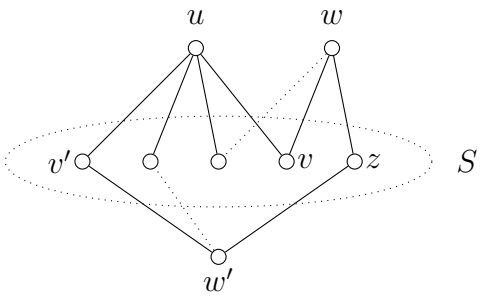


Figure 7.3: The independent 2-clique S and its (partial) neighborhood selected in the proof of Lemma 7.10. Dotted lines are possible edges so z can be at distance 2 from other vertices in S but those are unimportant for the proof.

In this situation v, z, z' with their three pairwise neighbors create a 6-cycle sharing the edge wz with C and thus G is not a cactus.

- u has at least 3 neighbors and w has only v as a neighbor in N_u. Let z' be a vertex of $N_u \setminus \{v', v\}$. Then z and z' must have a common neighbor x (which cannot be u or w but could be w'). Then wz is a common edge of C and the 6-cycle (u, z', x, z, w, v, u) and thus G is not a cactus.
- u has at least 3 neighbors and w has at least 2 neighbors in N_u, say v and z'. Then vw is a common edge of C and the 4-cycle (u, v, w, z', u) and thus G is not a cactus.

□

This lemma implies the following theorem:

Theorem 7.11. *Any (C_3, C_6)-free graph G satisfies $\alpha_{=2}(G) = \Delta(G)$ and* Max Independent 2-Clique *is linear-time solvable on it.*

Proof. By Lemma 7.10, in (C_3, C_6)-free graphs any independent 2-clique is the neighborhood of some vertex. Then, an independent 2-clique of maximum size is given in linear time by taking the neighborhood of a vertex of maximum degree since two vertices in the neighborhood of any vertex are not adjacent in C_3-free graphs. □

Finally, Lemma 7.10 allows to give a polynomial-time algorithm for Max Independent 2-Clique on cactus.

Proposition 7.12. Max Independent 2-Clique *is linear-time solvable on cactus graphs.*

Proof. Since all cactus graphs have a bounded treewidth, an implicit algorithm running in linear time follows from the proof of Theorem 7.4.

Being more constructive, let $G = (V, E)$ be a cactus which is not isomorphic to C_6. Let $\{v_1, \ldots, v_n\}$ be the set of vertices in G and for any $i \in \{1, \ldots, n\}$, note N_i the neighborhood

of v_i in G and construct S_i as an independent set of maximal[2] size in the subgraph induced by N_i in linear time. Notice that S_i is actually an independent set of maximum size in the subgraph induced by N_i since any vertex of N_i may only have at most one neighbor in N_i since G is a cactus. Let S be one of the sets S_i, $i = 1, \ldots, n$, of maximum size. By construction, S is an independent 2-clique of maximum size among all independent 2-cliques in G in which every vertices have a common neighbor in V. Moreover, S can be found in linear time.

Now we prove that S is an independent 2-clique of maximum size in G. If $|S| \leq 3$, by Lemma 7.10 we know that $\alpha_{=2}(G) \leq 3$. Indeed, if $\alpha_{=2}(G) \geq 4$ then G would not be a cactus since S is an independent 2-clique of maximum size among all independent 2-cliques in G in which every vertices have a common neighbor. Then, if $|S| = 3$, S is already an independent 2-clique of maximum size. Now suppose that $|S| = 2$. Then $\alpha_{=2}(G) = 2$. Indeed, if $\alpha_{=2}(G) = 3$, since $|S| = 2$ any independent 2-clique of size 3 would be included in an induced cycle C_6 (as in Figure 7.3) without chord (since G is a cactus) and we note such a cycle c. Since G is not isomorphic to a cycle of length 6, one of the vertices (say v) of the cycle c has a neighbor out of c. Then since G is a cactus, the two neighbors of v in c and one neighbor of v in $V \setminus c$ is an independent 2-clique of size 3, which is a contradiction since $|S| = 2$. Thus, S is an independent 2-clique of maximum size. If $|S| \geq 4$, by Lemma 7.10, any independent 2-clique in G is included in the neighborhood of a vertex in V, then S is an independent 2-clique of maximum size in G. □

7.4.3 Other graph classes in which both problems are polynomial-time solvable

We focus now on classes of graphs on which both Max Independent 2-Clique and Max Independent Set are polynomial-time solvable. We first investigate a subclass of split graphs, namely threshold graphs. It follows from the definitions that a threshold graph $G = (V, E)$ is a split graph with the following property: the vertices of the independent set S can be ordered as $v_1, \ldots, v_p$ such that $N_G(v_1) \subseteq N_G(v_2) \subseteq \ldots \subseteq N_G(v_p)$. We denote by $u_1, \ldots, u_q$ the vertices of the clique K, and we suppose that $d_G(u_1) \leq d_G(u_2) \leq \ldots \leq d_G(u_q)$. Without loss of generality, we assume that there is no isolated vertex in G. Note that a threshold graph can be recognized in linear time (see [96]).

Proposition 7.13. Max Independent 2-Clique *is linear-time solvable on threshold graphs. Moreover, in every threshold graph G without isolated vertices we have $\alpha_{=2}(G) = \alpha(G)$.*

Proof. Let $G = (V, E)$ be a threshold graph with the previous decomposition into S and K. Let $N_G(v_p) = \{u_r, u_{r+1}, \ldots, u_q\}$, for some $r \geq 1$. Then a maximum independent 2-clique in G is S if $K \setminus N_G(v_p) = \emptyset$, and otherwise it is $S \cup \{z\}$ with any $z \in K \setminus N_G(v_p)$, since in both cases the common neighbor of all these vertices is u_q. Since Max Independent

[2]Vertices of N_i are added to S_i one by one until it is not possible to add a vertex from $N_i \setminus S_i$ to S_i without compromising the fact that S_i is an independent 2-clique.

SET can be solved in linear time in threshold graphs [73], MAX INDEPENDENT 2-CLIQUE can be solved in linear time. □

The previous result can be extended in two directions, for interval graphs and for cographs.

Using the results of Booth and Lueker [26] it can be tested in linear time whether a graph G is an interval graph; and if it is, then an interval representation $I_1, \ldots, I_n$ of G can also be generated.

Proposition 7.14. MAX INDEPENDENT 2-CLIQUE *is polynomial-time solvable on interval graphs.*

Proof. Consider any $G = (V, E)$ and let $I_1, \ldots, I_n$ be an interval representation of G. In order to determine $\alpha_{=2}(G)$, first notice that all vertices of an independent 2-clique S of G must have a common neighbor. Indeed, if I and I' are the leftmost and the rightmost intervals of S then any of their common neighbors intersects all intervals located between them, and therefore is a common neighbor of all members of S. Then, for every vertex I, we compute a maximum independent set in the subgraph induced by the neighborhood of I. An optimal solution is such an independent set with maximum size. Since MAX INDEPENDENT SET is polynomial-time solvable on interval graphs [89], the result follows. □

We consider now the class of cographs, that contains all threshold graphs, and we show that MAX INDEPENDENT 2-CLIQUE is linear-time solvable on this class. In [48], Courcelle *et. al.* proved that any problem expressible in Monadic Second-Order Logic is linear-time solvable for graphs of bounded clique-width. Since cographs are exactly the graphs with clique-width at most 2 [49], MAX INDEPENDENT 2-CLIQUE is linear-time solvable on cographs. We give an alternative proof which is more constructive.

To each cograph G with n vertices, we can associate a rooted tree T, called the *cotree* of G. Leaves of T correspond to vertices of the graph G, and internal nodes of T are labeled with either '$\cup$' (union-node) or '$\times$' (join-node). A subtree rooted at node '$\cup$' corresponds to the vertex-disjoint union of the subgraphs defined by the children of that node, and a subtree rooted at node '$\times$' corresponds to the complete join of the subgraphs defined by the children of that node; that is, we add an edge between every two vertices corresponding to leaves in different subtrees under the join-node in question. Cographs can be recognized in linear time and the cotree representation can be obtained efficiently [45, 90]. Moreover, any cotree can easily be transformed in linear time to a binary cotree with $O(n)$ nodes.

Proposition 7.15. MAX INDEPENDENT 2-CLIQUE *is linear-time solvable on cographs.*

Proof. Consider a cograph G with n vertices $v_1, \ldots, v_n$. Given a binary cotree representation T of G with $O(n)$ nodes, we show in the following how to solve MAX INDEPENDENT 2-CLIQUE recursively.

Let $x_1, \ldots, x_t$ be the nodes of T where x_r is its root and t is in $O(n)$. For $i = 1, \ldots, t$, denote by T_i the subtree rooted at x_i, G_i the subgraph induced by the vertices corresponding to the leaves of T_i, and V_i the set of these vertices.

For each i, we compute $\alpha_{=2}(G_i)$ 'bottom-up' in the cotree. We start by computing values of leaves, and after that the value of an internal node if the values of its two children are already computed. Together with $\alpha_{=2}(G_i)$ we also determine the independence number $\alpha(G_i)$, which is well known to admit an easy recursion (which follows immediately by the constructive definition of cographs).

Given a node x_i of the cotree, the corresponding values are obtained as follows:

- If x_i is a leaf then $\alpha_{=2}(G_i) = |V_i| = 1$. Also, $\alpha(G_i) = 1$.
- If x_i is a union-node with two children x_ℓ and x_r, we have no edges between G_ℓ and G_r. Then any maximum independent 2-clique of G_i is entirely contained either in G_ℓ or in G_r. So, $\alpha_{=2}(G_i) = \max\{\alpha_{=2}(G_\ell), \alpha_{=2}(G_r)\}$. On the other hand, clearly, $\alpha(G_i) = \alpha(G_\ell) + \alpha(G_r)$.
- If x_i is a join-node with two children x_ℓ and x_r, every vertex in V_ℓ is adjacent to every vertex in V_r. Then a maximum independent 2-clique in G_i is a maximum independent set entirely contained either in G_ℓ or in G_r. So, $\alpha_{=2}(G_i) = \alpha(G_i) = \max\{\alpha(G_\ell), \alpha(G_r)\}$.

Since each step can be performed in constant time, moreover postorder traversal requires linear time, the algorithm runs proportionally to the size of the cotree, which is $O(n)$. □

Notice that since Max Independent Set is linear-time solvable on chordal graphs [73], it is also linear-time solvable on interval graphs and threshold graphs. Moreover, Max Independent Set is also linear-time solvable on cographs by bottom-up tree computation [43].

7.5 NP-hardness and non-approximability

We investigate graph classes in which Max Independent 2-Clique is NP-hard and, in some case, non approximable in polynomial time. Using the reduction from the proof of Theorem 7.2, we can conclude:

- Max Independent 2-Clique is NP-hard on dense (resp. everywhere dense) graphs, since Max Independent Set is NP-hard on dense (resp. everywhere dense) graphs. Moreover, Max Independent 2-Clique is not $n^{1-\varepsilon}$-approximable for any $\varepsilon > 0$, if $\mathsf{P} \neq \mathsf{NP}$, on everywhere dense graphs (and respectively dense graphs) since the same result holds for Max Independent Set on everywhere dense graphs (and respectively dense graphs). In order to get this last result, we use the same inaproximability result for Max Independent Set on general graphs [161] and a reduction preserving approximation from general graphs to everywhere dense graphs

(that consists of adding a clique of the same size as the size of the graph and joining every vertex from the original graph to all vertices in this clique).

- Max Independent 2-Clique is NP-hard on K_4-free graphs, since Max Independent Set is NP-hard on K_3-free graphs [5].

We now investigate graph classes in which Max Independent 2-Clique is NP-hard while Max Independent Set is polynomial-time solvable.

7.5.1 Split graphs

We first consider a graph class containing threshold graphs, namely the class of split graphs, for which Max Independent 2-Clique becomes NP-hard (and even not $n^{1-\varepsilon}$-approximable). Since Max Independent Set is polynomial-time solvable on chordal graphs [73], it is also polynomial-time solvable on split graphs.

Proposition 7.16. Max Independent 2-Clique *is NP-hard on split graphs.*

Proof. We reduce Max Clique on general graphs to Max Independent 2-Clique on split graphs. Let $G = (V, E)$ be an instance of Max Clique. We define an instance $G' = (V', E')$ of Max Independent 2-Clique on split graphs as follows: for every vertex $v_i \in V$ we consider a vertex $v'_i \in V'$ and for every edge $e \in E$ we consider a vertex e' in V'. We also add an additional vertex z in V'. Moreover, for any edge $e = v_1v_2 \in E$ we associate two edges in E', the edges v'_1e' and v'_2e'. Finally, the subgraph induced by vertices $e' \in V'$ and z is defined to be a clique. Now it is easy to see that C is a clique of size at least k in G if and only if $C' = \{v' : v \in C\} \cup \{z\}$ is an independent 2-clique of size at least $k+1$ in G'. On the other hand, given an independent 2-clique S, if $z \notin S$ and $e' \in S$ holds for some (only one) $e \in E$, then we can modify S to an independent 2-clique of the same size by replacing e' with z. Hence, the maximum can always be attained by involving z. □

Theorem 7.17. Max Independent 2-Clique *is W[1]-complete on split graphs.*

Proof. From Theorem 7.1, we know that Max Independent 2-Clique belongs to W[1]. On the other hand, the reduction in Proposition 7.16 is an FPT-reduction. Since Max Clique is W[1]-hard on general graphs [53], then Max Independent 2-Clique is also W[1]-hard on split graphs. □

Theorem 7.18. Max Independent 2-Clique *is not $n^{1-\varepsilon}$-approximable in polynomial time on split graphs unless P=NP.*

Proof. We construct an E-reduction from Max Clique. Let $I = (V, E)$ be an instance of Max Clique and let $I' = (V', E')$ be the corresponding instance of Max Independent 2-Clique, considering the same reduction as in Proposition 7.16. First, notice that $opt(I) = opt(I') - 1$, thus we have $opt(I') \leq 2 \cdot opt(I)$. Now let S' be an independent

2-clique of I' of size at least 2 and let S be the set of all copies of vertices from V in S'. Since $opt(I) = opt(I') - 1$ and $|S| = |S'| - 1$, we obtain $opt(I) - |S| = opt(I') - |S'|$. Since it has been proved in [161] that MAX CLIQUE is not $n^{1-\varepsilon}$-approximable in polynomial time unless P = NP, MAX INDEPENDENT 2-CLIQUE is not $n^{1-\varepsilon}$-approximable in polynomial time on split graphs unless P = NP. □

7.5.2 Bipartite graphs

We prove now that MAX INDEPENDENT 2-CLIQUE is NP-hard (and even not $n^{1/2-\epsilon}$-approximable, unless P = NP) on bipartite graphs while MAX INDEPENDENT SET is polynomial-time solvable since the number of vertices in a maximum independent set equals the number of edges in a minimum edge covering.

Proposition 7.19. MAX INDEPENDENT 2-CLIQUE *is NP-hard on bipartite graphs.*

Proof. MAX INDEPENDENT SET is known to be NP-hard on 3-regular graphs [78], so MAX CLIQUE is also NP-hard on $(n-4)$-regular graphs (where n is the number of vertices), by considering its complement. We reduce MAX CLIQUE on $(n-4)$-regular graphs to MAX INDEPENDENT 2-CLIQUE on bipartite graphs. Let $G = (V, E)$ be an $(n-4)$-regular graph. We construct an instance of $G' = (V', E')$ of MAX INDEPENDENT 2-CLIQUE on bipartite graphs as follows (see Figure 7.4).

Let V_1, V_2, V_3, V_4 be four copies of V. Let E_1 be a set of $|E|$ vertices corresponding to the edges in E, and define $V' := V_1 \cup V_2 \cup V_3 \cup V_4 \cup E_1$. Let there exist an edge in E' between a vertex v in V_i, $i \in \{1, 2, 3, 4\}$ and a vertex e in E_1 if and only if the corresponding vertex v in V is incident with the corresponding edge e in E.

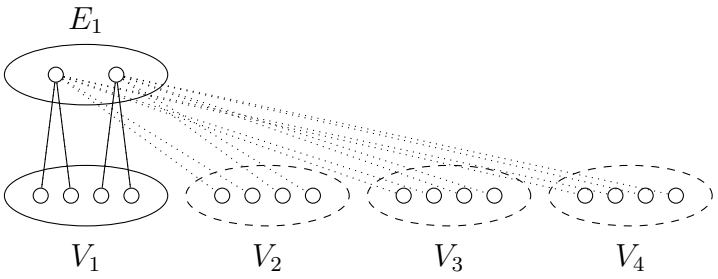


Figure 7.4: The bipartite graph G', an instance of MAX INDEPENDENT 2-CLIQUE

Now we show that G contains a clique of size at least k if and only if G' contains an independent 2-clique of size at least $4k$.

Given a clique $C \subseteq V$ of size at least k in G, the union of the four copies of C in G' is an independent 2-clique of size at least $4k$.

For the other direction, notice first that the value of a maximum independent set in a 3-regular graph is at least $\lceil \frac{n}{4} \rceil$. Then, the value of a maximum clique in an $(n-4)$-regular

graph is also at least $\lceil \frac{n}{4} \rceil$. Thus the size of a maximum independent 2-clique in G' is at least n.

We consider now a solution C' of MAX INDEPENDENT 2-CLIQUE in G' with at least $4k \geq n$ vertices (this restriction is always possible because of the previous comment). Notice that C' cannot contain both a vertex from E_1 and a vertex from $V' \setminus E_1$ since the distance between any two vertices of C' must be 2. A solution which is a subset of E_1 would mean pairwise intersecting edges in G, hence would have size at most $\max(3, n-4) < n$. Therefore C' must be a subset of $V' \setminus E_1$. Notice that for any $i \in \{1,2,3,4\}$, $C' \cap V_i$ must be a copy of a clique in G. Then C' is a union of copies of four cliques in G, and $|C'| \geq 4k$. Let C_0 be the copy of largest size, which thus has $|C_0| \geq k$. Then C_0 is the copy of a clique C of G of size at least k. □

Theorem 7.20. MAX INDEPENDENT 2-CLIQUE *is not* $n^{1/2-\varepsilon}$*-approximable in polynomial time on bipartite graphs, unless* $\mathsf{P} = \mathsf{NP}$.

Proof. We construct an E-reduction from MAX CLIQUE. Let $I = (V, E)$ be an instance of MAX CLIQUE. Consider a reduction similar to the one in the proof of Proposition 7.19, except that we now consider $\ell = |V|$ copies $V_1, \ldots, V_\ell$ instead of four copies of V; adjacencies are defined in the same way as before. We denote by $I' = (V', E')$ the corresponding instance of MAX INDEPENDENT 2-CLIQUE from the reduction. As in Proposition 7.19, starting with a clique of size $opt(I)$, we can construct an independent 2-clique of size $\ell \cdot opt(I)$ in G' and thus $opt(I') \geq \ell \cdot opt(I)$. Let S' be any independent 2-clique in I' of size at least ℓ (it always exists, take e.g. the ℓ copies of the same vertex, one copy in each V_i). As before S' cannot contain both a vertex of E_1 and a vertex from $V \setminus E_1$ since two vertices of S' must have distance 2 in G', and S' cannot contain only vertices from E_1 since any independent 2-clique included in E_1 is of size at most $\max(3, \Delta(G)) \leq \ell - 1$. Moreover, each subset $V_i \cap S'$ corresponds to a clique in G. Let S be the subset $V_i \cap S'$ of largest size. We have $|S| \geq \frac{|S'|}{\ell}$ and then $opt(I) \geq |S| \geq \frac{|S'|}{\ell} = \frac{opt(I')}{\ell}$ when S' is an optimal solution. Using that $opt(I') \geq \ell \cdot opt(I)$ we get $opt(I') = \ell \cdot opt(I)$ and we obtain:

$$\epsilon(I, S) = \frac{opt(I)}{|S|} - 1 \leq \frac{\ell \cdot opt(I')}{\ell \cdot |S'|} - 1 = \epsilon(I', S')$$

Since we clearly have $opt(I') \leq p(|I|) \cdot opt(I)$ with a polynomial p, the reduction is an E-reduction. Then, since MAX CLIQUE is not $\ell^{1-\varepsilon}$-approximable unless P=NP [161], the same property holds for MAX INDEPENDENT 2-CLIQUE. Thus MAX INDEPENDENT 2-CLIQUE is not $n^{1/2-\varepsilon}$ approximable where $n = |V'|$ since $n = \ell^2 + |E|$. □

7.5.3 Line graphs

Finally we prove that MAX INDEPENDENT 2-CLIQUE is NP-hard (and even APX-hard) on line graphs, while MAX INDEPENDENT SET is polynomial-time solvable since it consists in a maximum matching in the original graph.

Proposition 7.21. MAX INDEPENDENT 2-CLIQUE *is* NP*-hard on line graphs.*

Proof. We establish a reduction from the MAX CLIQUE problem on general graphs. Consider an instance $G = (V, E)$ of MAX CLIQUE with $|V| = n$. We construct a graph $G' = (V', E')$ (see Figure 7.5) as follows. Let $G_0 = (V_0, E_0)$ be a copy of G. Let V' be $V_0 \cup A \cup B \cup C$ where A, B, C are three sets of n vertices. Then, let $E' = E_0 \cup E_1 \cup E_2 \cup E_3 \cup E_4$ such that E_1 is a perfect matching between V_0 and A, E_2 is the set of all possible edges (i.e., a complete bipartite graph) between the vertices of A and the vertices of B, E_3 is a perfect matching between B and C, and E_4 is the set of all possible edges between any two vertices of C (a complete subgraph). The line graph of G', denoted by $L(G')$, is an instance of MAX INDEPENDENT 2-CLIQUE. Notice that an independent 2-clique in $L(G')$ corresponds to a set of edges in G' such that, for each pair of edges $\{e_1, e_2\}$ in the set, e_1 and e_2 are not adjacent but are joined by an edge. We show that G contains a clique of size at least k if and only if $L(G')$ contains an independent 2-clique of size at least $k + n$.

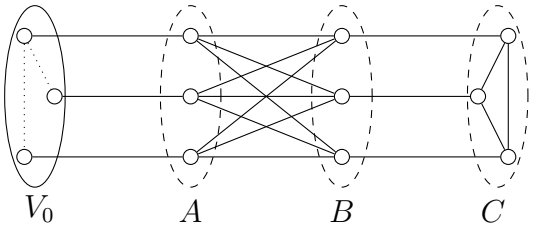


Figure 7.5: The graph G' for which the corresponding line graph $L(G')$ is an instance of MAX INDEPENDENT 2-CLIQUE

Consider a clique S of size k in G, and let S_0 be its copy in G'. We define a set of edges S' of size at least $k + n$ in G' as follows. For any vertex $v \in S_0$, add in S' its adjacent edge in E_1. Moreover add the entire E_3 to S'. We show now that any pair of edges in S' have an adjacent edge in common. Two edges of $S' \cap E_1$ have a common adjacent edge in E_0 since the subgraph induced by S_0 is a clique. Similarly, two edges of E_3 have a common adjacent edge in E_4. Moreover, an edge of $S' \cap E_1$ and an edge of E_3 have a common adjacent edge in E_2 since the subgraph induced by $A \cup B$ is $K_{n,n}$. Then, the corresponding set of vertices in $L(G')$ is an independent 2-clique of size $k + n$.

In the other direction, consider an independent 2-clique in $L(G')$ of size $k + n$. Notice that it is always possible to take the set of vertices in $L(G')$ corresponding to E_3 in G' and two edges in E_1 whose vertices in V_0 are neighbors in G', hence we can suppose that $k \geq 2$. Let S' be the set of all corresponding edges in G'. Suppose first that there is exactly one edge from E_0 in S'. Then, there are at most $n - 2$ edges from E_1 in S', and there are at most 2 edges from E_2 in S', due to the constraints of an independent 2-clique. There cannot be edges from $E_3 \cup E_4$ in S' since they would not be joined to the edge of $E_0 \cap S'$ by any edge. Then, S' contains at most $n + 1$ edges in S', which contradicts $k \geq 2$. Suppose now that there are at least $g \geq 2$ edges from E_0 in S'. Name two of them $e_{0,1}$ and $e_{0,2}$. Then, there are at most $n - 2g$ edges from E_1 in S' but there is no edge from E_2 in S'. Indeed, an edge e_2 from E_2 in S' can be joined by an edge to at most one of $e_{0,1}$ and $e_{0,2}$. Then the size of S' does not exceed n, which contradicts $k \geq 2$. Thus, we can assume that

there is no edge from E_0 in S'. Similarly, there is no edge from E_4 in S'. Now, notice that $|S' \cap (E_2 \cup E_3)| \leq n$ since if $S' \cap (E_2 \cup E_3)$ contained $n+1$ edges then at least two of these edges would have a common endpoint. Consequently, $|S' \cap E_1| \geq k$. Moreover, any two edges from $S' \cap E_1$ must have a common adjacent edge in E_0 since they cannot have a common adjacent edge in E_2. Then, the subgraph of G induced by the set of vertices in V_0 which are the endpoints of the edges in $S' \cap E_1$ must be a clique whose size is at least k. □

Theorem 7.22. Max Independent 2-Clique *is* APX*-hard on line graphs.*

Proof. We construct now an L-reduction from Max Clique to Max Independent 2-Clique on line graphs. Let I be an instance of Max Clique on graphs of degree at least $n-4$ and I' the corresponding instance of Max Independent 2-Clique on line graphs from the previous reduction. We prove that this reduction is an L-reduction. We proved in Proposition 7.21 that any independent 2-clique in I' has a size at most $2n$. Then $opt(I') \leq 2n = 8 \cdot \frac{n}{4} \leq 8 \cdot opt(I)$ follows since $opt(I) \geq \frac{n}{4}$ in graphs of degree at least $n-4$. Moreover, starting with a clique of size $opt(I)$, we can construct an independent 2-clique of size $opt(I)+n$ and therefore $opt(I') \geq n + opt(I)$. Let S' be an independent 2-clique in I' of size at least $n+2$ (we proved in Proposition 7.21 that it always exists and that such a set must be included in $E_1 \cup E_2 \cup E_3$). Let S be the set of vertices in V_0 which are incident with edges in $E_1 \cap S'$. We have $|S'| - |S| \leq n$ which implies $n + |S| \geq |S'|$. Then we obtain $opt(I) - |S| \leq opt(I') - n - |S| = opt(I') - (n + |S|) \leq opt(I') - |S'|$. Since Max Independent Set is APX-hard on the class of graphs of maximum degree 3 [6], Max Clique is also APX-hard on the class of graphs of minimum degree at least $n-4$. Thus, Max Independent 2-Clique is APX-hard on line graphs. □

7.6 Conclusions

Despite that Max Independent 2-Clique and Max Independent Set are similar problems, their complexity can be very different depending on the graph class we try to solve the problem in. We mainly showed that Max Independent 2-Clique is NP-hard on apex, dense and everywhere dense, K_4-free, split, bipartite and line graphs while it is polynomial-time solvable on bounded treewidth, planar, bounded degree (and complement of bounded degree), (C_3, C_6)-free, interval graphs and cographs. Many further types of graphs may be of interest, concerning separation of graph classes in which the problem is NP-hard from the ones where the problem is solvable in polynomial time.

8 Conclusions

The development of social networks and online meet-up services have made the study of community detection a major recent stake. In this thesis, we studied four particular definitions which are relevant for different reasons. Since the definitions for a community can be various and have different interests depending on the aspect of the cohesion we want to capture, it might be interesting to investigate other definitions.

On the other hand, we suggest some research directions for future work. For instance, in Chapter 4, we investigate the notion of community structure, in which each member has a greater proportion of neighbors in its part than in any other part. We saw that there exist graphs in which there is no 2-community structure, and the existence of k-community structure in a graph remains open for general values of k. An interesting relaxation of the problem would be to consider that each member has only to have a larger proportion of neighbors in its part than outside of its part. This definition is equivalent to the notion of k-community structure for $k = 2$, but becomes less restrictive for higher values of k. Furthermore, the definition of a community structure can take some robustness into account by asking, given a partition, for each vertex to satisfy the proportion condition even if we remove a certain constant q of their neighbors from the graph.

In the context of meet-up services, we investigate in Chapter 7 the notion of independent 2-clique in which any two members of such potential community has a common acquaintance. The vertices that ensure this property between two vertices in an independent 2-clique is the heart of the problem which makes the difference with the problem of just finding an independent set. Given an independent 2-clique S, we call the set of vertices that have at least two neighbors in S the *support* of S. Then, an interesting problem that we can study is, given a graph and some integer k, to find an independent set of size k such that the size of its support is maximized. This problem makes sense since the size of the support gives a good natural quality function of the cohesion of such potential community.

In further research around meet-up services, it might be interesting to define even other aspects of potential communities. In particular, signed graphs, in which labels "+" and "-" represent good and bad relationships, could be used in order to capture good potential communities. For instance, an induced path (x, y, z) of length 2 with two labels "+" reveal

a good potential relationship between x and z, and such path with both labels "+" and "-" reveals some incompatibility between x and z. New problems could arise from this paradigm. Notice that is also possible to consider directed signed graphs in which there is an edge from a vertex x to a vertex y if the member x has some (positive or negative) judgment about y. We can then discuss several situations that often occur in real life: if x likes y and y like z, it is likely that x should like z, independently from the fact that y likes x or not. Directed graphs could then be worth of interest to modelize such situation. Given one of these paradigms, it could be interesting to find a community of a fixed size k that maximize the quality of cohesion. Such quality could be evaluated according to real life observations as we suggested previously.

List of Figures

Bibliography

[1] N. Abbas and L. K. Stewart. Clustering bipartite, chordal graphs: Complexity, sequential, parallel algorithms. *Discrete Applied Mathematics*, 91(1-3):1–23, 1999.

[2] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. In *Proceedings of the 5th Latin American Symposium on Theoretical Informatics (LATIN 2002)*, pages 598–612, 2002.

[3] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[4] R. D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:3–113, 1973.

[5] V. E. Alekseev. On the local restrictions effect on the complexity of finding the graph independence number. *Combinatorial-Algebraic Methods in Applied Mathematics*, pages 3–13, 1983.

[6] P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In *Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC 1997)*, pages 288–298, 1997.

[7] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1):123 – 134, 2000.

[8] T. Antal, P. L. Krapivsky, and S. Redner. Dynamics of social balance on networks. *Physical Review E*, 72(3):036121, 2005.

[9] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221, 2000.

[10] Y. Asahiro, E. Miyano, and K. Samizo. Approximating maximum diameter-bounded subgraphs. In *Proceedings of the 9th Latin American Symposium on Theoretical Informatics (LATIN 2010)*, pages 615–626, 2010.

[11] H. Aziz, F. Brandt, and P. Harrenstein. Fractional hedonic games. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2014)*, pages 5–12, 2014.

[12] B. Balasundaram, S. Butenko, and S. Trukhanov. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10(1):23–39, 2005.

[13] M. J. Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76:066102, 2007.

[14] C. Bazgan, J. Chlebíková, C. Dallard, and T. Pontoizeau. Family of graphs without 2-community structure. In *The 10th International Colloquium on Graph Theory and combinatorics (ICGT 2018)*, 2018.

[15] C. Bazgan, J. Chlebíková, and T. Pontoizeau. New insight into 2-community structures in graphs with applications in social networks. In *Proceedings of the 9th International Conference on Combinatorial Optimization and Applications (COCOA 2015)*, pages 236–250, 2015.

[16] C. Bazgan, J. Chlebíková, and T. Pontoizeau. Structural and algorithmic properties of 2-community structures. *Algorithmica*, 80(6):1890–1908, 2018.

[17] C. Bazgan, T. Pontoizeau, and Z. Tuza. On the complexity of finding a potential community. In *Proceedings of the 10th International Conference on Algorithms and Complexity (CIAC 2017)*, pages 80–91, 2017.

[18] C. Bazgan, T. Pontoizeau, and Z. Tuza. Finding a potential community in networks. *Theoretical Computer Science*, accepted.

[19] C. Bazgan, Z. Tuza, and D. Vanderpooten. Degree-constrained decompositions of graphs: Bounded treewidth and planarity. *Theoretical Computer Science*, 355(3):389–395, 2006.

[20] C. Bazgan, Z. Tuza, and D. Vanderpooten. The satisfactory partition problem. *Discrete Applied Mathematics*, 154(8):1236–1245, 2006.

[21] C. Bazgan, Z. Tuza, and D. Vanderpooten. Approximation of satisfactory bisection problems. *Journal of Computer and System Sciences*, 74:875–883, 2008.

[22] A. A. Bertossi. Dominating sets for split and bipartite graphs. *Information Processing Letters*, 19(1):37–40, 1984.

[23] D. Bilò, L. Gualà, and G. Proietti. Improved approximability and non-approximability results for graph diameter decreasing problems. *Theoretical Computer Science*, 417:12–22, 2012.

[24] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[25] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

[26] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[27] C. Bordenave, M. Lelarge, and L. Massoulié. Non-backtracking spectrum of random graphs: community detection and non-regular ramanujan graphs. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 1347–1357, 2015.

[28] S. P. Borgatti, M. G. Everett, and P. R. Shirey. LS sets, lambda sets and other cohesive subsets. *Social Networks*, 12(4):337–357, 1990.

[29] A. S. Brahim, B. L. Grand, L. Tabourier, and M. Latapy. Citations among blogs in a hierarchy of communities: Method and case study. *Journal of Computational Science*, 2(3):247–252, 2011.

[30] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On finding graph clusterings with maximum modularity. In *Proceedings of the 33rd International Conference on Graph-theoretic Concepts in Computer Science (WG 2007)*, pages 121–132, 2007.

[31] S. Cafieri, P. Hansen, and L. Liberti. Improving heuristics for network modularity maximization using an exact algorithm. *Discrete Applied Mathematics*, 163:65–72, 2014.

[32] A. Cami, H. Balakrishnan, N. Deo, and R. D. Dutton. On the complexity of finding optimal global alliances. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 58:23–31, 2006.

[33] D. Cartwright and F. Harary. Structural balance: A generalization of Heider's theory. *Psychological review*, 63:277–293, 1956.

[34] R. Cazabet, R. Baccour, and M. Latapy. Tracking bitcoin users activity using community detection on a network of weak signals. *CoRR*, abs/1710.08158, to appear.

[35] M. R. Cerioli, L. Faria, T. O. Ferreira, C. A. J. Martinhon, F. Protti, and B. Reed. Partition into cliques for cubic graphs: Planar case, complexity and approximation. *Discrete Applied Mathematics*, 156(12):2270–2278, 2008.

[36] M. R. Cerioli, L. Faria, T. O. Ferreira, and F. Protti. A note on maximum independent sets and minimum clique partitions in unit disk graphs and penny graphs: complexity and approximation. *RAIRO-Theoretical Informatics and Applications*, 45(3):331–346, 2011.

[37] J. Chang, J. Yang, and S. Peng. On the complexity of graph clustering with bounded diameter. In *Proceedings of the 18th International Computer Science and Engineering Conference (ICSEC 2014)*, pages 18–22, 2014.

[38] M. Chang, L. Hung, C. Lin, and P. Su. Finding large k-clubs in undirected graphs. *Computing*, 95(9):739–758, 2013.

[39] C.-K. Cheng and Y. C.-A. Wei. An improved two-way partitioning algorithm with stable performance [VLSI]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(12):1502–1511, 2006.

[40] J. Chlebíková. Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters*, 60(5):225–230, 1996.

[41] T.-Y. Choe and C.-I. Park. A k-way graph partitioning algorithm based on clustering by eigenvector. In *Proceedings of the 4th International Conference of Computational Science (ICCS 2004)*, pages 598–601. Springer, 2004.

[42] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158. ACM, 1971.

[43] D. G. Corneil, H. Lerchs, and L. S. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, pages 163 – 174, 1981.

[44] D. G. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9(1):27–39, 1984.

[45] D. G. Corneil, Y. Perl, and L. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[46] B. Courcelle. The monadic second-order logic of graphs III : tree-decompositions, minors and complexity issues. *RAIRO - Informatique Théorique et Applications*, 26:257–286, 1992.

[47] B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

[48] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[49] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000.

[50] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. In *Proceedings of the 19th International Colloquium on Trees in Algebra and Programming (CAAP 1994)*, pages 68–84, 1994.

[51] D. Delling, A. Goldberg, T. Pajor, and R. Werneck. Customizable route planning. In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA 2011)*, pages 376–387, 2011.

[52] J. S. Deogun, D. Kratsch, and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *Information Processing Letters*, 61(3):121–127, 1997.

[53] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1):109–131, 1995.

[54] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer, 2012.

[55] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72:027104, 2005.

[56] A. Dumitrescu and J. Pach. Minimum clique partition in unit disk graphs. In *Graphs and Combinatorics*, volume 27, pages 399–411. Springer, 2011.

[57] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[58] V. Estivill-Castro and M. Parsa. On connected two communities. In *Proceedings of the 36th Australasian Computer Science Conference (ACSC 2013)*, volume 135, pages 23–30, 2013.

[59] V. Estivill-Castro and M. Parsa. Hardness and tractability of detecting connected communities. In *Proceedings of the 39th Australasian Computer Science Week Multiconference (ACSW 2016)*, pages 25:1–25:6, 2016.

[60] T. S. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80:016105, 2009.

[61] M. G. Everett and S. Borgatti. Role colouring a graph. *Mathematical Social Sciences*, 21(2):183–188, 1991.

[62] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster analysis*. John Wiley & Sons, Ltd, 5th edition, 2011.

[63] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[64] A. Feldmann and L. Foschini. Balanced partitions of trees and applications. *Algorithmica*, 71:354–376, 2015.

[65] H. Fernau and D. Raible. Alliances in graphs: a complexity-theoretic study. In *Proceedings on the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2007)*, pages 61–70, 2007.

[66] H. Fernau and J. A. Rodríguez-Velázquez. A survey on alliances and related parameters in graphs. *Electronic Journal of Graph Theory and Applications*, 2(1):70–86, 2014.

[67] J. Fiala and D. Paulusma. A complete complexity classification of the role assignment problem. *Theoretical Computer Science*, 349(1):67–81, 2005.

[68] G. Flake, S. Lawrence, and C. Giles. Efficient identification of web communities. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000)*, pages 150–160, 2000.

[69] H. Fleischner, E. Mujuni, D. Paulusma, and S. Szeider. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science*, 410(21-23):2045–2053, 2009.

[70] L. R. Ford Jr. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, pages 399–404, 1956.

[71] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3–5):75 – 174, 2010.

[72] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[73] A. Frank. Some polynomial algorithms for certain graphs and hypergraphs. *Congressus Numerantium No. XV*, pages 3–13, 1976.

[74] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[75] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(1-2):25–66, 1967.

[76] Y. Gao, D. Hare, and J. Nastos. The parametric complexity of graph diameter augmentation. *Discrete Applied Mathematics*, 161(10-11):1626–1631, 2013.

[77] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[78] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[79] B. Gaume. Balades aléatoires dans les petits mondes lexicaux. *I3 Information Interaction Intelligence*, 4(2):39–96, 2004.

[80] M. Gerber and D. Kobler. Partitioning a graph to satisfy all vertices. *Technical report,Swiss Federal Institute of Technology*, 1998.

[81] M. Gerber and D. Kobler. Algorithmic approach to the satisfactory graph partitioning problem. *European Journal of Operational Research*, 125(2):283–291, 2000.

[82] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences*, volume 99, pages 7821–7826, 2002.

[83] A. V. Goldberg. Finding a maximum density subgraph. Technical report, 1984.

[84] O. Goldschmidt and D. S. Hochbaum. Polynomial algorithm for the k-cut problem. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS 1988)*, pages 444–451, 1988.

[85] P. Golovach, P. Heggernes, D. Kratsch, and A. Rafiey. Finding clubs in graph classes. *Discrete Applied Mathematics*, 174:57–65, 2014.

[86] M. González, H. Herrmann, J. Kertész, and T. Vicsek. Community structure and ethnic preferences in school friendship networks. *Physica A: Statistical Mechanics and its Applications*, 379(1):307–316, 2007.

[87] M. Grötschel, L. Lovász, and A. Schrijver. Stable sets in graphs. In *Geometric Algorithms and Combinatorial Optimization*, pages 272–303. Springer, 1988.

[88] L. Gulikers, M. Lelarge, and L. Massoulié. A spectral method for community detection in moderately sparse degree-corrected stochastic block models. *Advances in Applied Probability*, 49(03):686–721, 2017.

[89] U. I. Gupta, D. T. Lee, and J. Y. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12(4):459–467, 1982.

[90] M. Habib and C. Paul. A simple linear time algorithm for cograph recognition. *Discrete Applied Mathematics*, 145(2):183–197, 2005.

[91] M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[92] F. Harary. On the notion of balance of a signed graph. *The Michigan Mathematical Journal*, 2(2):143–146, 1953.

[93] S. Hartung, C. Komusiewicz, and A. Nichterlein. Parameterized algorithmics and computational experiments for finding 2-clubs. *Proceedings of the 7th International Symposium on Parameterized and Exact Computation (IPEC 2012)*, pages 231–241, 2012.

[94] S. Hartung, C. Komusiewicz, A. Nichterlein, and O. Suchý. On structural parameterizations for the 2-club problem. *Discrete Applied Mathematics*, 185(C):79–92, 2015.

[95] J. He, J. Hopcroft, H. Liang, S. Suwajanakorn, and L. Wang. Detecting the structure of social networks using (α,β)-communities. *Proceedings of the 8th International Workshop on Algorithms and Models for the Web Graph (WAW 2011)*, pages 26–37, 2011.

[96] P. Heggernes and D. Kratsch. Linear-time certifying recognition algorithms and forbidden induced subgraphs. *Nordic Journal of Computing*, 14:87–108, 2007.

[97] F. Heider. Attitudes and cognitive organization. *The Journal of Psychology*, 21(1):107–112, 1946.

[98] B. Hendrickson and R. W. Leland. A multi-level algorithm for partitioning graphs. In *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, pages 28–41, 1995.

[99] S. D. Hochbaum. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., 1997.

[100] L. H. Jamieson, S. T. Hedetniemi, and A. A. McRae. The algorithmic complexity of alliances in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 68:137–150, 2009.

[101] S. Jespersen, I. M. Sokolov, and A. Blumen. Relaxation properties of small-world networks. *Physical Review. E*, 62:4405–4408, 2000.

[102] P. F. Jonsson and P. A. Bates. Global topological features of cancer proteins in the human interactome. *Bioinformatics*, 22(18):2291–2297, 2006.

[103] P. F. Jonsson, T. Cavanna, D. Zicha, and P. A. Bates. Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metastasis. *BMC Bioinformatics*, 7(1), 2006.

[104] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, 1972.

[105] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[106] A. Kenneth and H. Wolfgang. Every planar map is four colorable : Part i. discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.

[107] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Journal*, 49:291–307, 1970.

[108] S. Khanna, R. Motwani, M. Sudan, and U. V. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1998.

[109] J. Kleinberg. An impossibility theorem for clustering. In *Proceedings of the 15th International Conference on Neural Information Processing Systems (NIPS 2002)*, pages 463–470. MIT Press, 2002.

[110] C. Komusiewicz. Multivariate algorithmics for finding cohesive subnetworks. *Algorithms*, 9(1):21, 2016.

[111] P. Kristiansen, S. Hedetniemi, and S. Hedetniemi. Alliances in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 48:157–177, 2004.

[112] C. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930.

[113] S. Laan, M. Marx, and R. J. Mokken. Close communities in social networks: boroughs and 2-clubs. *Social Network Analysis and Mining*, 6(1):20, 2016.

[114] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80:056117, 2009.

[115] E. Lawler. Cutsets and partitions of hypergraphs. *Networks*, 3:275–285, 1973.

[116] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web (WWW 2008)*, pages 695–704. ACM, 2008.

[117] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*, pages 631–640. ACM, 2010.

[118] C. Li, S. McCormick, and D. Simchi-Levi. On the minimum-cardinality-bounded-diameter and the bounded-cardinality-minimum-diameter edge addition problems. *Operations Research Letters*, 11:303–308, 1992.

[119] D. Lokshtanov, N. Misra, G. Philip, M. S. Ramanujan, and S. Saurabh. Hardness of r-dominating set on graphs of diameter $(r+1)$. In *Proceedings of the 8th International Symposium on Parameterized and Exact Computation (IPEC 2013)*, pages 255–267, 2013.

[120] L. Lovász. A characterization of perfect graphs. *Journal of Combinatorial Theory, Series B*, 13(2):95–98, 1972.

[121] F. Luccio and M. Sami. On the decomposition of networks in minimally interconnected subnetworks. *IEEE Transactions on Circuit Theory*, 16(2):184–188, 1969.

[122] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Proceedings of the 6th International Workshop on Program Comprehension (IWPC 1998)*, pages 45–52, 1998.

[123] L. Massoulié. Community detection thresholds and the weak ramanujan property. In *Symposium on Theory of Computing (STOC 2014)*, pages 694–703, 2014.

[124] H. Matsuda, T. Ishihara, and A. Hashimoto. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science*, 210(2):305–325, 1999.

[125] N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan. Finding strongly knit clusters in social networks. *Internet Mathematics*, 5:155–174, 2008.

[126] R. J. Mokken. Cliques, clubs and clans. *Quality and Quantity*, 13(2):161–173, 1979.

[127] R. J. Mokken, E. M. Heemskerk, and S. Laan. Close communication and 2-clubs in corporate networks: Europe 2010. *Social Network Analysis and Mining*, 6(1):40, 2016.

[128] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965.

[129] M. Newman. Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330, 2004.

[130] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67:026126, 2003.

[131] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69:026113, 2004.

[132] R. Niedermeier. *Invitation to fixed-parameter algorithms*. 2006.

[133] M. Olsen. A general view on computing communities. *Mathematical Social Sciences*, 66(3):331–336, 2013.

[134] F. M. Pajouh and B. Balasundaram. On inclusion wise maximal and maximum cardinality k-clubs in graphs. *Discrete Optimization*, 9(2):84 – 97, 2012.

[135] G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446, 2007.

[136] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[137] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[138] A. Parley, S. Hedetniemi, and A. Proskurowski. Partitioning trees: Matching, domination, and maximum diameter. *International Journal of Computer & Information Sciences*, 10(1):55–61, 1981.

[139] I. A. Pirwani and M. R. Salavatipour. A weakly robust PTAS for minimum clique partition in unit disk graphs. *Algorithmica*, 62(3-4):1050–1072, 2012.

[140] J. Plesnik. The complexity of designing a network with minimum diameter. *Networks*, 11(1):77–85, 1981.

[141] P. Pons and M. Latapy. *Computing Communities in Large Networks Using Random Walks*, pages 284–293. Springer, 2005.

[142] A. Puig-Centelles, O. Ripolles, and M. Chover. Surveying the identification of communities. *International Journal of Web Based Communities*, 4:334–347, 2008.

[143] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[144] V. Raman and S. Saurabh. Triangles, 4-cycles and parameterized (in-)tractability. *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT 2006)*, pages 304–315, 2006.

[145] F. S. Roberts and L. Sheng. How hard is it to determine if a graph has a 2-role assignment? *Networks*, 37(2):67–73, 2001.

[146] S. E. Schaeffer. Survey: Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[147] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012.

[148] A. Schoone, H. Bodlaender, and J. Leeuwen. Diameter increase caused by edge deletion. *Journal of Graph Theory*, 11(3):409–427, 1987.

[149] S. B. Seidman. LS sets as cohesive subsets of graphs and hypergraphs. *Mathematical Social Sciences*, 6:87–91, 1983.

[150] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

[151] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.

[152] D. Shmoys. Cut problems and their application to divide-and-conquer. In *Approximation Algorithms for NP-hard Problems*, pages 192–235. 1997.

[153] J. Sigarreta, S. Bermudo, and H. Fernau. On the complement graph and defensive k-alliances. *Discrete Applied Mathematics*, 157(8):1687–1695, 2009.

[154] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.

[155] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[156] L. Wang, J. Hopcroft, J. He, H. Liang, and S. Suwajanakorn. Extracting the core structure of social networks using (α, β)-communities. *Internet Mathematics*, (1):58–81, 2013.

[157] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4):43:1–43:35, 2013.

[158] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, MDS '12, pages 3:1–3:8. ACM, 2012.

[159] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[160] H. Zhou and R. Lipowsky. Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In *Proceedings of the 4th International Conference on Computational Science (ICCS 2004)*, pages 1062–1069. Springer Berlin Heidelberg, 2004.

[161] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

Résumé

Cette thèse étudie la détection de communautés dans le contexte des réseaux sociaux. Un réseau social peut être modélisé par un graphe dans lequel les sommets représentent les membres et les arêtes représentent les relations entre les membres. En particulier, nous étudions quatre différentes définitions de communauté. D'abord, une structure en communautés peut être définie par une partition des sommets telle que tout sommet a une plus grande proportion de voisins dans sa partie que dans toute autre partie. Cette définition peut être adaptée pour l'étude d'une seule communauté. Ensuite, une communauté peut être vue comme un sous-graphe tel que tout couple de sommets sont à distance 2 dans ce sous-graphe. Enfin, dans le contexte des sites de rencontre, nous proposons d'étudier une définition de communauté potentielle dans le sens où les membres de la communauté ne se connaissent pas, mais sont liés par des connaissances communes. Pour ces quatre définitions, nous étudions la complexité et l'approximation de problèmes liés à l'existence ou la recherche de telles communautés dans les graphes.

Abstract

This thesis deals with community detection in the context of social networks. A social network can be modeled by a graph in which vertices represent members, and edges represent relationships. In particular, we study four different definitions of a community. First, a community structure can be defined as a partition of the vertices such that each vertex has a greater proportion of neighbors in its part than in any other part. This definition can be adapted in order to study only one community. Then, a community can be viewed as a subgraph in which every two vertices are at distance 2 in this subgraph. Finally, in the context of online meetup services, we investigate a definition for potential communities in which members do not know each other but are related by their common neighbors. In regard to these proposed definitions, we study computational complexity and approximation within problems that either relate to the existence of such communities or to finding them in graphs.

Mots Clés

Graphes, réseaux sociaux, algorithme, complexité, approximation, structures en communautés, s-clubs, independent 2-cliques

Keywords

Graphs, social networks, algorithm, complexity, approximation, community structures, s-clubs, independent 2-cliques