Centre Val de Loire UNIVERSITÉ Spirit of the Loire Valley

UNIVERSITÉ D'ORLÉANS

UNIVERSITE D'ORLEANS

ÉCOLE DOCTORALE MATHÉMATIQUES, INFORMATIQUE, PHYSIQUE THÉORIQUE ET INGÉNIERIE DES SYSTÈMES

Laboratoire d'Informatique Fondamentale d'Orléans

Thèse

présentée par :

Pedro MONTEALEGRE BARBA

soutenue le : **28 Février 2017**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/ Spécialité : **Informatique**

Algorithmes de Graphes Séquentiels et Distribués

Algorithmes Paramétrés via des Cliques Maximales Potentielles ; Modèle de Diffusion dans une Clique Congestionnée

THÈSE dirigée par :

Ioan Todinca	Professeur des Universités, Université d'Orléans

RAPPORTEURS :

Pierre Fraigniaud	Directeur de Recherche CNRS
Christophe Paul	Directeur de Recherche CNRS

JURY :

Florent Becker	Maître de Conférences, Université d'Orléans
Pierre Fraigniaud	Directeur de Recherche CNRS
Cyril Gavoille	Professeur des Universités, Université Bordeaux I
Nicolas Nisse	Chargé de Recherche CNRS
Christophe Paul	Directeur de Recherche CNRS
Ioan Todinca	Professeur des Universités, Université d'Orléans

Dédié à Patricia,
et nos mille routes.

Remerciements

Je voudrais remercier premièrement mon directeur de thèse Ioan Todinca. Ça a été un énorme privilège d'apprendre de lui tous les passionnants sujets sur lesquels j'ai travaillé dans ma thèse. Je lui suis très reconnaissant pour son énorme qualité humaine, et pour tous les conseils, pas uniquement scientifiques, que j'ai reçus de sa part.

Je voudrais également remercier mes rapporteurs Pierre Fraigniaud et Christophe Paul pour leurs remarques sur ma thèse. Je suis très touché par l'intérêt qu'ils ont manifesté pour mon manuscrit. Je remercie de la même manière mes co-auteurs Florent Becker, Mathieu Liedloff, Iván Rapaport et Fedor V. Fomin, avec lesquels j'ai travaillé pendant cette période.

Cette thèse n'aurait pas été possible sans le soutien du Programa de Formación de Capital Humano Avanzado – Becas Chile de CONICYT. Merci pour la confiance accordée à ma recherche, et j'espère que celle-ci constitue une réelle contribution à mon pays.

Je suis très content d'avoir réalisé ma thèse au sein du Laboratoire d'Informatique Fondamentale d'Orléans, que je remercie, en particulier les membres de l'équipe de Graphes, algorithmes et modèles de calcul (GAMoC), pour tout le support pendant ces années de thèse. Je remercie notamment à Isabelle Renard pour sa patience et son amabilité constante, et mes collègues doctorants et ATER, à commencer par Diego, Valentin, Bruno et Martin.

Je ne peux pas oublier de mentionner ici mes amis Chiliens, avec qui nous avons entrepris l'aventure de venir en France pour réaliser nos thèses : mon grand ami Emilio, mais aussi Sebastián, Valentina, Sandra, et Jean-Marc et Anne (qui comptent aussi comme des amis Chiliens).

Je suis très reconnaissant aussi envers Eric Goles, que je ne considère pas seulement un professeur mais également un ami. Son soutien constant a été fondamental pour ma formation en tant que chercheur. Merci de m'encourager dans ce processus, pour tes conseils, et tes recommandations littéraires.

Je pense également à ma famille, à mes parents Alberto et Pilar, à mes sœurs Sofía et Teresa et à mon frère Pablo, qui de manière présentielle ou virtuelle ont toujours su m'accompagner et m'encourager dans ce voyage professionnel et personnel.

Finalement, je voudrais remercier Patricia. Son amour, sa compagnie et sa foi constante en moi sont mes principales sources d'inspiration. Pour la femme de ma vie, avec amour et gratitude ; je lui dédie cette thèse.

Contents

Chapitre 1

Introduction

Depuis ses débuts, la théorie des graphes a été un outil fondamental pour modéliser plusieurs structures et problèmes d'intérêt théorique et pratique. Des *sept ponts de Königsberg*, jusqu'au problème du *voyageur de commerce*. Des réseaux de régulation en biologie, jusqu'aux graphes quantiques en physique théorique. Des réseaux de neurones, jusqu'à des réseaux sociaux. Cette thèse parle des aspects structuraux et algorithmiques des graphes. Elle est divisée en deux parties, qui en développent deux perspectives différentes : une perspective centralisée-séquentielle, et une perspective parallèle-distribuée.

La première partie de cette thèse concerne des algorithmes de graphes dans le sens *classique*, séquentiel : le graphe est stocké en un seul lieu, et l'objectif est de concevoir des algorithmes qui résolvent des problèmes d'optimisation autour de ces graphes, par exemple le problème du voyageur de commerce, ou le problème de l'arbre couvrant de poids maximum, ou même le problème de la coloration. Cette étude suit le point de vue classique de la *théorie de la complexité*, en particulier de l'étude des problèmes NP-Difficiles dans des instances particulières.

De manière plus précise, la Partie I – intitulée « Algorithmes paramétriques via cliques maximales potentielles » - étudie des applications algorithmiques de deux structures de graphes appelées *séparateurs minimaux* et *cliques maximales potentielles*. Ces deux objets sont au cœur d'un méta-théorème, formulé d'après Fomin, Todinca and Villanger [67], qui déclare qu'une grande famille de problèmes d'optimisation en graphes peut être résolue en temps polynomial, si le graphe d'entrée contient un nombre polynomial de séparateurs minimaux. La contribution de cette partie de la thèse est l'extension du méta-théorème de Fomin *et al.*, en deux directions différentes : d'un côté, on l'adapte pour le rendre valide pour une famille plus grande de problèmes ; de l'autre, on prolonge ce résultat vers une version paramétrée, pour certains paramètres des graphes.

La deuxième partie de cette thèse est consacrée à *l'approche distribuée*. Certaines applications des graphes, par exemple celles que l'on retrouve dans les réseaux sociaux, dans les télécommunications ou dans les systèmes des capteurs, n'arrivent pas à être bien modélisées par l'approche *classique*. Des phénomènes comme l'existence des données dynamiques ou décentralisées, par exemple, rendent les algorithmes classiques inutiles ou inutilisables dans la pratique. On étudie des algorithmes de graphes sous l'hypothèse que les données ne se situent pas en un même lieu, mais sont *distribuées* dans un système. Chaque partie du système en constitue une entité autonome, qui peut seulement accéder à une certaine information locale. Ces entités exécutent des protocoles de communication, dont le but est de combiner des parties essentielles des données locales, pour en obtenir une certaine information globale.

Plus spécifiquement, la Partie II constitue une étude du modèle de « Diffusion dans une Clique Congestionnée » (en anglais : Broadcast Congested Clique). Dans ce modèle, les sommets d'un graphe communiquent entre eux en des rondes synchroniques, en diffusant un message de petite taille visible par tout autre sommet. On y conçoit des protocoles qui reconnaissent des propriétés de graphes en minimisant la taille des messages et le nombre de rondes. La contribution de cette partie est l'exploration du rôle du hasard dans ce modèle, ainsi que des protocoles pour la reconnaissance et la reconstruction des certaines classes des graphes.

Les deux parties de cette thèse sont indépendantes, bien qu'elles partagent certaines définitions, cer-

taines classes des graphes, et certains paramètres. En préambule, le Chapitre 2 présente les définitions et les résultats qui sont à la base des deux parties de cette thèse. La Partie I sera abordée dès le Chapitre 3 jusqu'au Chapitre 7, tandis que les Chapitres 8 à 13 forment la Partie II. Chaque partie contiendra son propre chapitre de préliminaires, et sa propre conclusion.

Dans les prochaines sections on introduira les motivations et les objectifs des deux parties qui seront développées dans cette présentation.

1.1 Partie I : Algorithmes paramétrées via cliques maximales potentielles

Plusieurs problèmes en graphes, comme par exemple le problème stable maximale ou le problème de la coloration, sont connus comme difficiles, dans le sens qu'il n'existe pas un algorithme efficace – c'est-à-dire, qui prends un temps polynomial – pour les résoudre, à moins qu'une conjecture, ici $P \neq NP$ (c.f. [7]), soit fausse. Ce phénomène est encore plus évident lorsqu'on parle des problèmes *réels*, comme ceux qu'on retrouve dans le domaine des télécommunications. Il existe diverses techniques pour aborder ces problèmes : des algorithmes d'approximation, des algorithmes probabilistes (ou randomisés), des algorithmes exacts exponentiels, et l'étude de cas particuliers. Dans cette partie de la thèse on va s'intéresser principalement à l'étude des cas particuliers, ou, plus précisément, à l'étude des problèmes difficiles, lorsque leur entrée est restreinte à une famille –ou classe- de graphes particulière, ou lorsqu'un certain paramètre du problème -ou de l'entrée- reste assez petit.

1.1.1 Décompositions arborescentes

L'une des méthodes pour attaquer des problèmes difficiles c'est la *décompositions des graphes*. L'idée c'est de découper le graphe dans plusieurs morceaux, définis par une certaine règle de découpage, et en résoudre le problème morceau par morceau. La règle est définie de manière qu'il soit possible de *coller* les solutions partielles obtenues dans chaque morceau, pour en déduire une solution globale. Pour résoudre le problème dans chaque morceau, on peut y reappliquer la règle de découpage, afin d'obtenir des morceaux de plus en plus petits, en obtenant une *structure arborescente*.

L'idée de décomposer un graphe vient des travaux en *décomposition modulaires* de Gallai [74]. Le graphe y est décomposé en *modules*, c'est-à-dire, des ensembles de sommets qui ont le même voisinage hors de l'ensemble. La décomposition modulaire est unique, et elle est calculable en temps polynomial [47]. Pour résoudre un problème difficile, il suffirait – grosso modo – de résoudre le problème dans les *graphes premiers*, qui sont les graphes où tous les modules son des singletons.

Les *décompositions arborescentes* constituent un autre type de décomposition des graphes, introduites au début des années 1980 par Robertson et Seymour [126], dans le contexte de l'étude des *mineurs de graphes*. Une décomposition arborescente est une décomposition d'un graphe dans plusieurs ensembles appelés « sacs ». Chaque sac est identifié avec un sommet d'un arbre, de manière que l'arbre et l'ensemble de sacs satisfassent : (1) pour chaque sommet il y a un sac qui le contient ; (2) pour chaque arête il y a un sac qui contient les deux extrémités de l'arête ; (3) pour chaque sommet, l'ensemble des sacs qui le contient forment un sous arbre, i.e., les sacs sont connectés dans l'arbre. La largeur d'une décomposition arborescente est égale à la taille du plus grand sac moins un. La largeur arborescente d'un graphe G correspond à la largeur minimum, parmi toutes les décompositions arborescentes de G.

Même si les décompositions arborescentes ont été introduites pour résoudre un problème combinatoire, ces outils ont été rapidement utilisés pour concevoir des algorithmes. Le célèbre Théorème de Courcelle [44] affirme qu'une grande famille de problèmes de graphes qui sont NP-Hard, comme le problème de couverture par sommets ou le cycle Hamiltonien, peuvent être résolus en temps polynomial – voire linéaire – lorsque l'entrée est restreinte une famille des graphes avec une largeur arborescente bornée par une constante. La famille de problèmes qui sont résolubles dans ce contexte sont ceux qui consistent à décider des propriétés exprimables dans la logique monadique de second ordre – notée CMSO_2 d'après son nom en anglais.

Dans CMSO_2, les variables sont des sommets et des arêtes d'un graphe. Les opérations permises sont l'égalité ($=$), la conjonction ($\vee$), la disjonction ($\wedge$), et la négation ($\neq$) avec l'adjacence entre deux sommets ($adj(x, y)$). Les quantificateurs peuvent être appliqués à des sommets, des arêtes, des ensembles

de sommets, et des ensembles d'arêtes. On accepte aussi un opérateur de comptage. Par exemple, la formule suivante exprime la propriété d'avoir une couverture par sommets de taille k :

$$\exists v_1, \dots v_k \in V \ (\forall u, v \in V \ (adj(u,v) \Rightarrow ((u = v_1) \vee \dots \vee (u = v_k) \vee (v = v_1) \vee \dots \vee (v = v_k))))$$

On peut définir aussi dans CMSO_2 des formules avec des variables libres, ce qui permet d'exprimer des problèmes d'optimisation qui consistent à trouver un ensemble maximum ou minimum de sommets ou d'arêtes qui satisfassent une certaine propriété.

Le Théorème de Courcelle affirme que le problème de décider si un graphe satisfait une propriété $\mathcal{P}$ exprimable en CMSO_2 peut être résolu en temps $f(\text{tw}, \mathcal{P}) \cdot n$, où n est le nombre de sommets du graphe d'entrée G et tw est le largeur arborescente du G. Il s'agit ici d'un exemple d'un problème résoluble à paramètre fixé -en anglais : FPT pour *fixed parameter tractable*. Ce sont des problèmes résolubles en temps $f(k) \cdot n^c$, où n est la taille de l'entrée, f est une fonction arbitraire -normalement on suppose que f est calculable-, $c > 0$ est une constante et k est un paramètre du problème qui ne dépends pas de n. Dans le cas du Théorème de Courcelle, le paramètre est la largeur arborescente du graphe d'entrée, plus la taille de la formule qui exprime $\mathcal{P}$.

Le point faible des algorithmes basés sur la largeur arborescente c'est qu'ils ont besoin de disposer d'une décomposition arborescente optimale du graphe d'entrée. Le calcul de la largeur arborescente d'un graphe arbitraire est NP-Difficile [5], même pour des familles restreintes des graphes, comme les graphes de dégré borné [29], ou les graphes bipartits [88]. Néanmoins, il existe certaines familles des graphes où le calcul de la largeur arborescente est polynomial. C'est le cas des graphes faiblement triangulés [32], les graphes de permutation [25], les graphes distance-héréditaires [37], les graphes d'intervalles circulaires [133], et les graphes de cercle [89]. Toutes ces classes des graphes ont en commun le fait d'avoir une quantité de séparateurs minimaux polynomiale en le nombre de sommets.

1.1.2 Séparateurs Minimaux, triangulations et cliques maximales potentiels

Une conjecture proposée par Kloks *et al.*, [90, 91] et démontrée par Bouchitté et Todinca [32, 33] confirme que, pour toute classe de graphes ayant un nombre polynomial des séparateurs minimaux, le calcul de la largeur arborescente se fait en temps polynomial.

Un graphe est appelé *triangulé* si tout cycle de largeur quatre ou plus contient une corde –c'est-à-dire, une arête qui connecte deux sommets non consécutifs dans le cycle. Pour un graphe G, une *triangulation* H est un graphe triangulé contenant G comme sous-graphe et ayant le même ensemble de sommets. La largeur arborescente d'un graphe G peut être caractérisée comme le minimum, parmi toutes les triangulations H de G, de $\omega(H) - 1$, où $\omega(H)$ est la taille d'une clique de cardinalité maximum dans H. Clairement, la triangulation optimale H de G -celle où $\omega(H) - 1$ est égale à la largeur arborescente de G- est trouvée parmi les *triangulations minimales* de G. Une triangulation H de G sera minimale si aucun sous-graphe strict de H ne forme une triangulation de G.

Bouchitté et Todinca [32] ont constaté que les ensembles de sommets d'un graphe G qui forment une clique maximale dans une triangulation minimale de G sont des objets ayant leur propre intérêt. Ils les ont appelés *cliques maximales potentielles*, et ont démontré qu'on peut calculer la largeur arborescente d'un graphe G en temps polynomial, si la liste des cliques maximales potentielles de G fait partie de l'entrée du problème. Plus précisément : soit Π_G la liste des cliques maximales potentielles d'un graphe G ; l'algorithme conçu par Bouchitté et Todinca calcule le largeur arborescente d'un graphe G en temps $\mathcal{O}(|\Pi_G|^d \cdot n)$. La technique principale derrière cet algorithme est un schéma de programmation dynamique sur toutes les solutions partielles, coupées par un séparateur minimal où une clique maximale potentielle. L'algorithme calcule la largeur arborescente du graphe au même temps qu'il construit la triangulation minimale correspondante.

Ultérieurement, Fomin, Kratsch et Todinca [64] ont amélioré l'algorithme qui calcule la largeur arborescente dû à Bouchitté et Todinca, en obtenant un nouvel algorithme linéaire par rapport au nombre de cliques maximales potentielles et polynomial en la taille du graphe -i.e., de complexité $\mathcal{O}(|\Pi_G| \cdot n^4)$, pour une constante $c > 0$. Par un résultat dû à Fomin, Todinca et Villanger [67], le nombre de cliques maximales potentielles dans un graphe quelconque à n sommets est $\mathcal{O}^*(1.7347^n)$. Ce deux résultats impliquent que la largeur arborescente est calculable en temps $\mathcal{O}^*(1.7347^n)$, ce qui améliore la borne $\mathcal{O}^*(2^n)$ obtenue par la force brute.

Bouchitté et Todinca [33] ont proposé aussi des caractérisations des cliques maximales potentielles en termes de séparateurs minimaux. Ce résultat a été utilisé pour montrer que le nombre de cliques maximales potentielles d'un graphe est borné par un polynôme qui dépend du nombre de ses séparateurs minimaux. D'ailleurs, ils montrent que la liste des cliques maximales potentielles est calculable dans les mêmes bornes. Ainsi, la largeur arborescente est calculable en temps polynomial pour toute classe de graphes ayant un nombre polynomial de séparateurs minimaux.

1.1.3 Sous-graphe induit maximum de largeur arborescente t et propriété $\mathcal{P}$

En 2010, Fomin et Villanger [68] ont trouvé une application nouvelle et assez surprenante pour les séparateurs minimaux et les cliques maximales potentielles. Considérons l'observation suivante : pour tout ensemble de sommets F d'un graphe G et pour toute triangulation H_F de $G[F]$, il existe une triangulation H de G tel que $H[F] = H_F$. Ils ont utilisé cette observation pour montrer que l'on peut trouver, dans un graphe G, un plus grand sous-graphe de largeur arborescente au plus t en temps $O(|\Pi_G| \cdot n^{t+3})$. Cet algorithme utilise un schéma de programmation dynamique similaire à celui conçu par Bouchitté et Todinca [33] pour calculer la largeur arborescente.

Ce nouveau résultat a montré que plusieurs problèmes NP-Difficiles, au-delà du calcul de la largeur arborescente, sont résolubles en temps polynomial dans des classes de graphes avec « peu » de séparateurs minimaux. Par exemple, des problèmes comme le stable maximum ou forêt induite maximum peuvent être exprimés comme des problèmes qui consistent à retrouver un sous-graphe induit maximum de largeur arborescente 0 ou 1, respectivement. Il était connu que certains de ces problèmes étaient polynomiaux dans des classes particulières des graphes avec « peu » de séparateurs minimaux, mais ces algorithmes utilisaient jusque-là des propriétés spécifiques de chaque classe de graphes.

Dans le schéma de programmation dynamique utilisé pour calculer le sous-graphe induit maximum de largeur arborescente t, la solution est construite au même temps qu'ne de ses décompositions arborescentes optimales. Fomin, Todinca et Villanger[67] ont montré qu'en même temps qu'une solution est construite -i.e. un sous-graphe induit maximum de largeur arborescente au plus t- on peut vérifier une propriété $\mathcal{P}$ exprimable en CMSO_2. Plus précisément, si $t > 0$ est une constante et $\mathcal{P}$ est une propriété CMSO_2, alors le problème Sous-graphe induit maximum de largeur arborescente t et propriété $\mathcal{P}$ consiste à calculer un ensemble de sommets F de cardinalité maximum tel que $G[F]$ est un graphe de largeur arborescente au plus t, qui satisfait la propriété $\mathcal{P}$. Les auteurs montrent que ce problème peut être résolu en temps polynomial pour les classes de graphes qui ont un nombre polynomial des séparateurs minimaux. Par exemple le problème de trouver un arbre induit de taille maximum peut être exprimé comme le problème de trouver un sous-graphe induit de largeur arborescente 1 avec la propriété d'être connecté.

Les résultats obtenus dans cette partie de la thèse se situent dans cette ligne de recherche. Ici, on étend légèrement la famille de problèmes qu'on peut résoudre en temps polynomial dans classes de graphes avec « peu » des séparateurs minimaux, mais principalement on élargit ce résultat vers une version paramétrée, pour certaines paramètres de graphes.

1.1.4 Contribution

La contribution de cette partie est présentée en cinq chapitres.

Le Chapitre 3 introduit les définitions et les résultats principaux pour comprendre les chapitres suivants. On commence avec les décompositions arborescentes et leur relation avec les triangulations minimales. Après, on donne quelques idées générales sur la logique monadique de second ordre et sur les propriétés régulières. Ensuite, on énonce le Théorème de Courcelle dans la version donnée par Borie, Parker et Tovey [31]. Ultérieurement, on présente les définitions et les principaux résultats autour des séparateurs minimaux et des cliques maximales potentielles, notamment leur caractérisation et leurs algorithmes d'énumération. Finalement, on esquisse les preuves et les algorithmes calculant la largeur arborescente, et un sous-graphe induit maximum de largeur arborescente t avec propriété $\mathcal{P}$.

Le Chapitre 4 présente un résultat de nature combinatoire. On montre que dans un graphe quelconque, le nombre de séparateurs minimaux et le nombre de cliques maximales potentielles sont bornés par une fonction de deux paramètres bien connus : le transversal minimum et la largeur modulaire. Le *transversal minimum* d'un graphe G, dénoté $\text{vc}(G)$ par son nom en anglais, est égal à la taille de la plus petite couverture par sommets de G. La *largeur modulaire* d'un graphe G, notée $\text{mw}(G)$, est égale au dégrée maximum d'un nœud premier dans l'arbre de décomposition modulaire de G (voir Chapitre 3 pour des

définitions plus précises). On montre que le nombre de séparateurs minimaux d'un graphe G est $3^{\mathrm{vc}(G)}$ et $\mathcal{O}^*(1.6181^{\mathrm{mw}(G)})$, et le nombre de cliques maximales potentielles est $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ et $\mathcal{O}^*(1.7347^{\mathrm{mw}(G)})$. D'ailleurs, ces objets son énumérables dans les mêmes bornes. Ici la notation $\mathcal{O}^*$ indique que les facteurs polynomiaux son cachés, i.e., $\mathcal{O}^*(f(n))$ doit être lu comme $\mathcal{O}(f(n))\cdot n^{\mathcal{O}(1)}$. Combiné avec les applications algorithmiques des cliques maximales potentielles, ces bornes montrent que le sous-graphe induit maximum de largeur arborescente t avec la propriété $\mathcal{P}$ est calculable en un temps d'exécution *simplement exponentiel* en le transversal minimum ou la largeur modulaire du graphe d'entrée. On finit ce chapitre avec une autre application, qui consiste à calculer la *profondeur arborescente* (en anglais, *tree-depth*).

La plupart des résultats de ce Chapitre ont été publiés dans le 14ème Symposium et atelier scandinave en théorie des algorithmes (SWAT 2014), en collaboration avec Fedor Fomin, Mathieu Liedloff et Ioan Todinca [66].

Le Chapitre 5 parle des graphes qui sont « proches » d'avoir un nombre polynomial de séparateurs minimaux. Pour un polynôme poly, on appelle $\mathcal{G}_{\mathrm{poly}}$ la classe des graphes G qui ont au plus $\mathrm{poly}(n)$ séparateurs minimaux, où n est le nombre de sommets de G. Pour $k > 0$, on appelle $\mathcal{G}_{\mathrm{poly}} + kv$ la classe des graphes G qui contient un ensemble $M \subseteq V(G)$, de taille au plus k, appelé modulateur, tel que $G^{\smile}M$ appartient à $\mathcal{G}_{\mathrm{poly}}$. On montre que le problème du sous-graphe maximum induit de largeur arborescente t avec propriété $\mathcal{P}$, paramétré par la taille du modulateur, est FPT (résoluble à paramètre fixé) si le modulateur fait partie de l'entrée. Ensuite, on montre que le problème de calculer le modulateur est $W[2]$-Difficile. Plus précisément, on montre que pour tout polynôme poly $= \Omega(n)$, le problème qui consiste à déterminer si un graphe appartient à la classe $\mathcal{G}_{\mathrm{poly}} + kv$, paramétré par k, est $W[2]$-Hard.

La plupart de ces résultats ont été publiés dans les 41ème Colloque international sur les concepts théoriques de graphes en informatique (WG 2015), et ont été obtenus en collaboration avec Mathieu Liedloff et Ioan Todinca [102].

Le Chapitre 6 est un chapitre beaucoup plus exploratoire. Son objectif est d'étendre les résultats de Fomin *et al.* [67] et leur algorithme résolvant le problème du sous-graphe induit maximum de largeur arborescente t avec propriété $\mathcal{P}$, vers une famille plus grande de problèmes. On montre un résultat de nature combinatoire, qui est que pour tout graphe G, le nombre de séparateurs minimaux des puissances impaires de G est au plus le nombre de séparateurs minimaux de G. Cela signifie que si G appartient à $\mathcal{G}_{\mathrm{poly}}$, pour un certain polynôme poly, alors G^k appartient aussi à $\mathcal{G}_{\mathrm{poly}}$ si k est impair. Il faut se rappeler que la k-ème puissance G^k d'un graphe G est le super-graphe de G où deux sommets en G^k sont adjacents s'ils sont à une distance au plus k dans G. Ce résultat combinatoire montre que des problèmes comme ensemble d-dispersé est résoluble en temps polynomial quand d est pair. Le problème ensemble d-dispersée, appelle d-scattered set en anglais, consiste à trouver l'ensemble de taille maximum tel que tout paire de sommets dans l'ensemble soit à distance au mois d (par exemple le problème stable maximum est équivalent au problème ensemble 2-dispersé). Ce problème a été posé par Eto et al. [59] où les auteurs montrent que le problème ensemble d-dispersé avec d impair est NP-Dur restreint à des graphes triangulés, et les auteurs se demandent si le problème est polynomial dans les graphes triangulés-bipartites pour d pair.

Ensuite, on explore des problèmes comme la couverture par sommets connectée ou l'ensemble dominant stable dans des classes spécifiques des graphes avec « peu » des séparateurs minimaux.

La plupart de ces résultats ont été publiés dans les 42ème Colloque international sur les concepts théoriques de graphes en informatique (WG 2016) en ont été obtenus en collaboration avec Ioan Todinca [112].

Le Chapitre 7 conclut cette partie de la thèse en proposant des questions ouvertes.

1.2 Partie II : Une étude du modèle de Diffusion dans une clique congestionnée

Comme on l'a déjà dit au début de ces pages, la deuxième partie de cette thèse parle d'algorithmes distribués. L'étude des systèmes distribués commence avec l'émergence des réseaux d'ordinateurs. Parfois inéluctables, comme dans les algorithmes qui manipulent le graphe d'Internet, les algorithmes distribués fournissent aussi des outils pour améliorer les algorithmes classiques -via la parallélisation, par exemple. La caractéristique principale d'un système distribué est qu'il y a plusieurs processeurs autonomes actifs, qui manipulent des données de manière parallèle. Les processeurs ont seulement accès à une certaine

partie de l'information, de manière locale, mais ils sont capables de communiquer entre eux pour accéder à plus d'information et prendre une décision globale.

Il existe plusieurs réseaux larges et complexes, dont le cerveau, l'Internet ou la société humaine constituent quelques exemples, qui sont modélisables comme un graphe et où chaque sommet est une entité capable de communiquer avec ses voisins dans le graphe. Dans tous ces exemples, la nature montre que les éléments qui composent ces systèmes sont capables de se coordonner entre eux, en fournissant ainsi une réponse globale.

Dans le modèle $\mathcal{CONGEST}$, un ensemble de n sommets –connus comme *nœuds*- sont connectés dans un graphe appelé *réseau de communication*. Chaque sommet de G correspond à un nœud, et chaque arête de G correspond à un canal de communication entre deux sommets. On assume que les nœuds ont des identifiants uniques, représentables avec $\mathcal{O}(\log n)$ bits -bien que certaines auteurs considèrent aussi des réseaux *anonymes* [71]). Les canaux de communication sont *congestionnées*, c'est-à-dire qu'ils ne peuvent pas transmettre que b bits par ronde. La valeur de b est appelée la *bande passante*. Dans la littérature, classiquement la bande passante des modèles congestionnée est bornée à $\mathcal{O}(\log n)$ bits.

La connaissance initiale d'un nœud x correspond à sa liste des arêtes -éventuellement avec leur poids- incidentes à x dans G. Les nœuds ont pour tâche de calculer ensemble une fonction du graphe G. Par exemple, une tâche d'un sommet x peut être de choisir des arêtes, parmi toutes les arêtes incidentes à x, afin que l'ensemble des arêtes choisies forme un arbre couvrant de poids maximal. La tâche peut être aussi un problème de *vérification*. Dans un tel cas, les nœuds reçoivent aussi comme entrée les arêtes incidentes à eux dans un autre graphe H, et la tâche s'avère de décider si G et H satisfont une certaine propriété. Par exemple, la tâche peut être de déterminer si H est un arbre couvrant de G. Pour accomplir ces objectifs, les sommets peuvent communiquer à travers les canaux de communication dans des rondes synchroniques, en envoyant dans chaque ronde un message par chaque arête, de taille bornée par la bande passante.

Un protocole du modèle $\mathcal{CONGEST}$ détermine les algorithmes locaux exécutés dans chaque nœud. Après chaque ronde de communication, les nœuds doivent décider de s'arrêter ou d'utiliser leur information locale et tous les messages reçus jusque-là pour en créer des nouveaux messages. Le protocole finit lorsque tous les sommets se sont arrêtés ; il est correct si tout nœud a correctement calculé la fonction de G. Dans le cas des problèmes de vérification, un protocole est correct si, lorsque G et H satisfont la propriété, tous les nœuds acceptent, ou au moins un nœud rejette lorsque G et H ne satisfont pas la propriété. Dans ce modèle on assume que la partie la plus lente d'un protocole est la transmission des messages par les canaux de communication. C'est pour cela qu'on considère que les temps d'exécution locaux des nœuds sont négligeables. Ainsi, dans le modèle $\mathcal{CONGEST}$, le temps d'exécution d'un protocole correspond au nombre de rondes nécessaires avant que tout nœud s'arrête -naturellement le temps d'exécution dépend de la taille de l'entrée et de le bande passante.

Le modèle $\mathcal{CONGEST}$ combine deux des principaux défis des systèmes distribués : la localité et la congestion. D'un point de vue théorique, c'est intéressant de séparer ces deux enjeux pour les mieux comprendre. Le modèle LOCAL est un exemple d'un modèle qu'isole le problème de la localité. Ce modèle est similaire au model $\mathcal{CONGEST}$, sauf que la bande passante n'est plus bornée. Ce modèle a été assez bien étudié, et il existe plusieurs résultats pour le modèle où les nœuds ont des identifiants uniques, aussi bien que pour la version anonyme [119, 96, 70].

De l'autre côté, on retrouve le modèle Clique Congestionnée, qui isole le problème de la congestion. Dans le modèle $\mathcal{CONGEST}$, plusieurs bornes inférieures pour le temps de exécution des protocoles impliquent le diamètre du graphe des canaux de communication. Par exemple, la borne pour le nombre des rondes nécessaires pour calculer un arbre couvrant de poids maximum est $\Omega(\sqrt{n}/b + D)$ pour les graphes de diamètre $D = \Omega(\log n)$ [129]. La borne devient $\Omega(\sqrt[3]{n}/\sqrt{b})$ ou $\Omega(\sqrt[4]{n}/\sqrt{b})$, lorsque le graphe d'entrée est restreint à avoir un diamètre trois ou quatre, respectivement [108]. Une question naturelle à poser est de savoir qu'est ce qui se passe lorsque le diamètre D est égal à un, i.e., lorsque le graphe de chaînes de communication est une clique.

1.2.1 Le modèle de la Clique Congestionnée

Le modèle de la clique congestionnée correspond au modèle $\mathcal{CONGEST}$ lorsque les n nœuds sont tous connectés entre eux, et a été proposé par Lotker, Patt-Shamir, Pavlov, et Peleg [107] ; ici, les auteurs montrent que le problème du calcul d'un arbre couvrant de poids maximum est calculable en $\mathcal{O}(\log \log n)$ rondes. Par la suite, on considère que le modèle Clique Congestionné a toujours une bande passante de

$\mathcal{O}(\log n)$ bits.

La motivation derrière l'étude de ce modèle n'est pas seulement théorique [76]. Dans certaines applications concrètes, comme les algorithmes de haut-niveau exécutés dans le réseau Internet, il n'y a pas de moyen de contrôler tous les sommets du graphe pour exécuter un protocole type $\mathcal{CONGEST}$. À sa place, on monte un *réseau de superposition*, où un plus petit nombre des nœuds sont répartis sur le réseau, et chacun contrôle un certain nombre de sommets. Ces processeurs exécutent le protocole dans les sommets qu'ils contrôlent, et en communiquent les résultats aux autres nœuds du réseau de superposition. Même si les nœuds du réseau de superposition contrôlent des sommets qui ne sont pas adjacents entre eux, ce n'est pas irréaliste de supposer une communication directe entre les nœuds du réseau de superposition.

Au cours des dernières années la recherche associée au modèle de la Clique Congestionnée a été assez active. Lenzen a en obtenu [100] un protocole pour acheminer (routing en anglais) n messages par sommet vers leurs destinations en temps constant, ainsi qu'un protocole en temps constant pour trier $\mathcal{O}(n^2)$ nombres, si chaque nœud connait initialement $\mathcal{O}(n)$ nombres. Ultérieurement, Hegeman *et al.* [82, 81] ont donné des protocoles randomisés en temps espéré constant pour des problèmes comme le calcul d'un ensemble 3-dispersé, et -sous certaines conditions- une approximation au facteur constant d'un arbre couvrant de poids maximal. Plus récemment, Censor-Hillel *et al.* [38] ont démontré l'existence d'un protocole qui exécute une sorte de multiplication des matrices en temps $\mathcal{O}(n^{(1-2/\omega)})$, où $\omega < 2.3728639$ est l'exposant dans le temps d'exécution du meilleur algorithme centralisé pour la multiplication des matrices. Ce résultat montre que des problèmes comme le fait de compter le nombre des triangles, calculer la maille, ou de calculer une $(1 + o(1))$ approximation des plus courts chemins entre toutes les paires de sommets, peuvent être résolus en temps $\mathcal{O}(n^{0.158})$ dans le modèle de la Clique Congestionnée. Très récemment, Ghaffari *et al.* [76] ont montré un autre protocole randomisé, qui calcule un arbre couvrant de poids maximum en temps $\mathcal{O}(\log^*(n))$, avec une haute probabilité.

Calculer des bornes inférieures pour le nombre des rondes dans le modèle de la Clique congestionnée est beaucoup plus difficile que dans le modèle $\mathcal{CONGEST}$. Comme le graphe de canaux de communication est un graphe complet, chaque sommet peut communiquer $\mathcal{O}(n \log n)$ bits à chaque ronde, ce qui fait un total de $\mathcal{O}(n^2 \log n)$ bits communiqués dans chaque ronde. La quantité d'information communiquée à chaque ronde est par conséquent énorme, ce qui suggère que les bornes inférieures provenant de la complexité de communication ne seront pas utiles dans ce modèle. Druker *et al.* [57] ont montré –grosso modo- que pour toute fonction f, une borne inférieure dans le temps de calcul de f dans le modèle de la Clique Congestionnée implique une borne inférieure dans la taille du circuit Booléen qui calcule f. En d'autres termes, de manière simplifiée, une borne inférieure dans le modèle de la Clique Congestionnée aurait des impacts forts dans d'autres domaines de l'informatique fondamentale.

1.2.2 Le modèle de Diffusion dans une Clique Congestionnée

La difficulté extrême pour montrer des bornes inférieures dans le modèle de la Clique Congestionnée a motivé l'étude des restrictions du modèle, pour mieux comprendre sa puissance et ce qui rend les bornes inférieures si difficiles de trouver. Une manière de s'y attaquer consiste à affaiblir la notion irréalisable qui dit que chaque nœud peut communiquer potentiellement $n - 1$ messages différents à chaque ronde, i.e. un message différent par chaque canal de communication. Le modèle de *Diffusion dans une Clique Congestionnée* est le modèle de la Clique Congestionnée, où chaque nœud ne peut pas communiquer qu'un seul message -le même pour tous ses canaux de communication-, à chaque ronde.

On connait très peu des protocoles rapides pour le modèle de Diffusion dans une Clique Congestionnée. Becker *et al.* [13] ont montré que, en une seule ronde, chaque nœud peut apprendre toutes les arêtes d'un graphe si le graphe d'entrée est un arbre, un graphe planaire, ou n'importe quel graphe de dégénérescence bornée. Plus formellement, si au début du protocole, le nœud x connait toutes les arêtes incidents à x dans un graphe d-dégénéré G, alors il existe un protocole dans le modèle de Diffusion dans une Clique Congestionnée qui permet de reconstruire G dans une ronde, et avec une bande passante $\mathcal{O}(d^2 \log n)$. Reconstruire signifie que chaque nœud apprend toutes les arêtes du graphe G, et donc toute nœud peut calculer n'importe quelle fonction dans le graphe G -même une fonction qui soit NP-Difficile à calculer, car on ne limite pas le temps d'exécution locale. Il existe d'autres protocoles qui marchent aussi en une seule ronde, par exemple pour décider si le graphe d'entrée est un certaine forêt fixée [57], ou un cograph [87].

Sans surprise, c'est plus facile de démontrer des bornes inférieures pour des protocoles dans le modèle de Diffusion dans une Clique Congestionnée. Dans [13] les auteurs montrent qu'il n'existe pas un protocole qui marche en une seule ronde et qui puisse détecter si le graphe d'entrée contient un triangle, ou s'il a

un diamètre plus grand que trois, au moins d'utiliser une bande passante de $\Omega(n)$ bits. Dans[87] Kari *et al.*, les auteurs montrent que si H est un graphe non-biparti, alors tout protocole d'un seule ronde qui détecte que le graphe d'entrée contient H sous-graphe doit avoir une bande passante $\Omega(n)$ bits.

Drucker *et al.* [57] ont été les premiers à montrer des bornes inférieures dans le modèle de Diffusion dans une Clique Congestionnée avec plus d'une ronde. Ils ont montré que pour détecter si le graphe d'entrée contient un cycle de longueur 4 on a besoin de $\Omega(n/b)$ rondes de communication, lorsque la bande passante est $\mathcal{O}(b)$. Récemment, Holzer *et al.* [85] ont montré que l'on a besoin de $\Omega(n/b)$ rondes pour calculer une $(2 - o(1))$ approximation des plus courtes chemins entre toutes les pairs de sommets.

Le fait de calculer un arbre couvrant de poids maximal ou, plus simplement, de calculer les composantes connexes d'un graphe, sont des problèmes fondateurs du modèle de la Clique Congestionnée. Un algorithme classique qui calcule un arbre couvrant de poids maximale dû à Borůvka [115] peut être implémenté dans le modèle de Diffusion dans une Clique Congestionnée avec un nombre $\mathcal{O}(\log n)$ de rondes et de bande passante $\mathcal{O}(\log n)$. On ignore si ce protocole est optimal ou pas. L'un des problèmes qui ont motivé cette thèse c'est précisément de savoir s'il existe un protocole non-trivial qui puisse calculer les composantes connexes avec une ronde dans le modèle de Diffusion dans une Clique Congestionnée.

Ahn *et al.* [1, 2] ont conçu un protocole randomisé dans le modèle de Diffusion dans une Clique Congestionnée qui calcule les composantes connexes -voire un arbre couvrant- d'un graphe avec une haute probabilité et une bande passante poly-logarithmique. Leur protocole utilise une très belle technique appelée « esquisses linéaires » (*linear sketches*), qui comprime l'information via une fonction linéaire. Les auteurs utilisent cette technique pour simuler l'algorithme de Borůvka en une seule ronde, avec une haute probabilité.

Une particularité des protocoles randomisés existants -par exemple le protocole mentionné dû à Ahn *et al.*, est l'utilisation des « pièces publiques », qui signifie que les bits randomisés utilisées par chaque sommet sont les mêmes parmi toutes les nœuds. Cette situation contraste avec le modèle « pièces privées », où les bits aléatoires utilisés par les nœuds son générés de manière indépendante. Cette partie de cette thèse explore le rôle du hasard dans le modèle de Diffusion dans une Clique Congestionnée, et sous certaines conditions, on arrive à simuler des protocoles randomisés par des protocoles déterministes.

1.2.3 Contribution

La contribution de cette partie sera présentée dans six chapitres.

Le Chapitre 8 commence avec les définitions formelles du modèle de Diffusion dans une Clique Congestionnée, et il est accompagné des principaux résultats connus. On donne formellement les distinctions entre des protocoles déterministes, des protocoles avec des « pièces privées », et des protocoles avec des « pièces publiques ». Ensuite, on présente les principaux outils et techniques qui seront utilisés dans les prochains chapitres.

Le Chapitre 9 parle de la distinction, du point de vue de la bande passante, des trois modalités différentes de protocoles dans le modèle de Diffusion dans une Clique Congestionnée : des protocoles déterministes, des protocoles avec des « pièces privées » et des protocoles avec « pièces publiques ». Cette séparation est faite à partir d'une fonction très naturelle appelée Jumeaux (*Twins* en anglais), qui consiste à décider si le graphe d'entrée contient deux sommets avec le même voisinage.

La plupart de ces résultats ont été publiés dans le 21éme Colloque international sur la structure de l'information et la complexité de communication (SIROCCO 2016), et ils ont été obtenus avec Florent Becker, Ivan Rapaport et Ioan Todinca [15].

Le Chapitre 10 donne des protocoles pour la reconnaissance et la reconstruction de petites classes de graphes. On y améliore le résultat de [13] pour la reconstruction des graphes d-dégénérées en une ronde avec un protocole de bande passante $\mathcal{O}(d \cdot \log n)$ -le protocole dû à Becker *et al.* [13] fonctionne avec une bande passante de taille $\mathcal{O}(d^2 \cdot \log n)$. Après, on étend le résultat de Kari *et al.* [87], en donnant des algorithmes randomisés pour la reconnaissance des graphes de distance héréditaire ou des graphes de largeur modulaire bornée. Finalement, on donne un protocole déterministe à deux rondes –ou randomisé à une ronde-, qui reconnait n'importe quelle famille de graphes de taille $2^{o(n^2)}$, avec une bande passante sous-linéaire -par exemple les graphes d'intervalle peuvent être reconnus dans deux rondes et avec une bande passante $\mathcal{O}(\sqrt{n}\log n)$.

Le Chapitre 11 étudie le problème de la détection des cycles -induits ou pas- dans le modèle de Diffusion dans une Clique Congestionnée. On donne ici des protocoles d'une ronde qui détectent si le graphe d'entrée

contient un cycle de taille au plus k -induit ou pas- avec une bande passante $\mathcal{O}(n^{2/(k-1)} \log n)$, si k est pair et $\mathcal{O}(n^{2/k} \log n)$ si k est impair. Si k est plus grand que 4, on montre que ces bornes sont serrés -modulo des facteurs logarithmiques. Ce borne inférieure est valide même si on permet que les nœuds ne reçoivent pas seulement les arêtes incidentes dans le graphe d'entrée, mais toutes les arêtes des voisins à distance $\lfloor k/4 \rfloor$.

D'un autre côté, la détection de cycles induits de taille au moins k est équivalente à montrer que le graphe d'entrée est $k-1$-triangulé. On prouve qu'on peut détecter un graphe k-triangulé en deux rondes et bande passante poly-logarithmique dans le modèle où les nœuds peuvent voire jusqu'au distance $\lfloor k/2 \rfloor + 1$. En particulier, cela montre que l'on peut détecter des graphes triangulés dans deux rondes et avec une bande passante poly-logarithmique, dans le modèle où les sommets peuvent voire jusqu'au distance deux.

Dans le Chapitre 12 on discute le problème de la connectivité. Ce problème a été l'une des motivations principales pour étudier le modèle de Diffusion dans une Clique Congestionnée. Le fait de ne pas avoir trouvé des bornes inférieures pour ce problème dans ce modèle représente l'une des frustrations de cette recherche. Cependant on donne des protocoles en nombre constant de rondes qui calculent les composantes connexes du graphe d'entrée, avec une bande passante sous-linéaire. On montre aussi que si les nœuds peuvent voir jusqu'au distance k, on peut calculer les composantes connexes en une ronde avec une bande passante $\mathcal{O}(n^{1/k} \cdot \log n)$.

Ces résultats ont été publiés comme une annonce courte dans le Symposium sur les principes du calcul distribué (PODC 2016), et ils ont été obtenus en collaboration avec Ioan Todinca [112].

Le Chapitre 13 conclut cette partie de la thèse en proposant quelques pistes de recherche.

Chapter 1

Introduction

From its origins, graph theory has been a fundamental tool to model countless *discrete* structures and problems, both of theoretical and practical interest. From the famous bridges of Königsberg, to the traveling salesman problem. From regulatory networks in biology, to quantum graphs in theoretical physics. From neural networks to social networks. This thesis is about structural and algorithmic aspects of graphs. It is divided in two parts, that develop two different perspectives: a centralized-sequential one, and a parallel-distributed one.

The first part of this thesis is about graph algorithms in the *classic* centralized and sequential approach: the graph is stored in a single machine, and the goal is to design efficient algorithms to solve optimization problems, for example traveling salesman problem, minimum spanning tree or coloring. The study follows the classical *computational complexity* point of view, in particular the study of NP-Hard problems restricted to particular instances.

More concretely, Part I - entitled *Parameterized algorithms via potential maximal cliques* - studies algorithmic applications of two graph structures called *minimal separators* and *potential maximal cliques*. These two objects are in the core of a meta-theorem due to Fomin, Todinca and Villanger [67], which states that a large family of graph optimization problems can be solved in polynomial time, when the input is restricted to the family of graphs with polynomially many minimal separators. The contribution of this part of the thesis is to extend the meta-theorem of Fomin *et al.* in two ways. On the one hand, we adapt it to be valid into a larger family of problems. On the other hand, we extend it into a parameterized version, for several graph parameters.

The second part of the thesis falls under the *distributed approach*. Several applications of graphs, for example the ones found in social networks, telecommunications or sensor systems, do not suit the *classic* approach. Indeed, phenomena like decentralized information, and dynamic information makes unusable or unfeasible classic centralized algorithms. We study graph algorithms assuming that the information is not located in the same place, but *distributed* in a system. Each part of the system is an autonomous entity with access to local information. The entities run communication protocols in order to combine crucial parts of the local knowledges and obtain some global information.

More concretely, Part II is a study of the *Broadcast Congested Clique* model. In this model, the nodes of a graph communicate in synchronous rounds, broadcasting a message of small size visible to every other node. The goal is to design protocols that recognize graph classes minimizing the number of rounds and the message sizes. The contribution of this part is to explore the role of randomness on this model, and provide protocols for the recognition and reconstruction of some graph classes.

The two parts of this thesis are independent and relatively unrelated, though several definitions, graph classes and parameters are studied on both parts. Before exposing each part, Chapter 2 presents the main definitions and results that will be common to the two perspectives of this thesis. Later, Chapters 3 to 7 contain Part I, while Chapters 8 to 13 contain Part II. Each part will contain its own specific preliminaries chapters and its own conclusions.

In the following sections, we introduce the main motivations and objectives of the two perspectives that are developed in this presentation.

1.1 Part I: Parameterized Algorithms via Potential Maximal Cliques

Many relevant graph problems, like maximum independent set or coloring, are known to be *hard* in the sense that there are no efficient (polynomial time) algorithms to solve them, unless some wide-believed conjecture is false, e.g., $P \neq NP$ (c.f. [7]). This phenomenon occurs even more often when we consider *real-world* problems, like the ones that one may encounter in telecommunications. In this context, there exist several techniques to cope with these *hard* problems, for example: approximation algorithms, randomized algorithms, moderately exponential algorithms and the study particular cases. We will mainly focus on the analysis of particular cases, which consists in restricting the inputs of our problems to belong to a particular family (for example a class of graphs), or restricting the problem to show efficient algorithms if some parameter of the problem (or of the input) is *small*.

1.1.1 Tree decompositions

One of the tools to address hard problems are *graph decompositions*. The idea is to cut the graph in several parts defined according to some rule, and solve the problem on each part. The rule is defined so it is possible to *glue* the partial solutions, obtaining a solution for the whole graph. To solve the problem, we may cut each each part into smaller parts recursively, according to the same rule, obtaining an *tree-structure*.

The idea of decomposing graphs goes back to the work of Gallai [74] on *modular decompositions*. The graph is decomposed into *modules*, which are sets of nodes that have the same neighborhood outside the set. The modular decomposition is unique, and computable in linear time [47]. To solve our hard problems, it would be enough (very roughly) to solve the problem in the *prime graphs*, which are the ones that only have singleton modules.

Tree decompositions are another type of graph decompositions, introduced in the beginning of the 80's by Robertson and Seymour [126], in the context of their study of graph minors. A tree decomposition is a decomposition of a graph into several (non-disjoint) vertex subsets called *bags*. The bags are identified with the vertices of a tree, and both the bags and the tree must satisfy (1) each vertex of the graph must belong to some bag (2) for each edge of the graph there is a bag that contain both endpoints of the edge and (3) for every vertex, the bags containing that vertex must induce a (connected) subtree. The width of a tree decomposition is the size of the largest bag minus one. The treewidth of a graph G corresponds to the minimum width among all possible tree decompositions of G.

Even if tree decompositions were introduced to solve a combinatorial problem, these tools were rapidly used from the algorithmic point of view. The celebrated Theorem of Courcelle [44] states that a wide family of NP-hard graph problems, like Minimum Vertex Cover or Hamiltonian Cycle can be solved in polynomial (even linear) time on graphs that admit tree decompositions with small sized bags. The family of problems solvable in this context are the ones consisting in decide graph properties expressible in CMSO_2 (*counting monadic second order logic*).

In CMSO_2, the variables are the vertices and edges of the graph. The allowed operations between variables are equality ($=$), conjunction ($\vee$), disjunction ($\wedge$), negation ($\neg$) together with the adjacency of two vertices ($adj(u, v)$). Quantifiers are allowed on vertices, edges, vertex sets or edge sets, and also the counting operation over sets. For example, the graph property consisting in containing a vertex cover of size exactly k can be expressed by the following CMSO_2 formula (which is in fact a *first order* formula):

$$\exists v_1, \dots v_k \in V \ (\forall u, v \in V \ (adj(u, v) \Rightarrow ((u = v_1) \vee \dots \vee (u = v_k) \vee (v = v_1) \vee \dots \vee (v = v_k))))$$

We can also define CMSO_2 formulas with free variables, which allow to express optimization problems consisting in find a set of vertices or edges satisfying a property.

Courcelle's Theorem states that deciding if a graph satisfies a CMSO_2 property $\mathcal{P}$ can be solved in time $f(\text{tw}, \mathcal{P}) \cdot n$, where n is the size of the input graph, and f is a computable function that depends on the treewidth tw of the input graph, and the size of a CMSO_2 formula that describes a property $\mathcal{P}$. This is an example of a *fixed parameter tractable* (FPT) problem, which are the problems that can be solved by an algorithm of time $f(k) \cdot n^c$, where $c > 0$, f is an arbitrary computable function and k is some parameter of the problem (in the case of Courcelle's Theorem, the parameter is treewidth of the input graph plus the size of the CMSO_2 formula that expresses $\mathcal{P}$).

The weakness of the algorithms based on treewidth is that they usually require that a minimal tree decomposition of the input graph is given in the input, so it must be computed beforehand. Computing

the treewidth of a graph is NP-Hard [5], even for restricted classes of graphs, like graphs of bounded degree [29] or bipartite graphs [88]. However there exist some graph classes for which computing treewidth is polynomial time solvable. It is the case of weakly chordal graphs [32], permutation graphs [25], distance-hereditary graphs [37], circle graphs [89], circular-arc graphs [133]. All these classes have in common, that the graphs that they contain have a number of *minimal separators* which is bounded by a polynomial on the size of graph.

1.1.2 Minimal separators, triangulations and potential maximal cliques

The property of having polynomially many minimal separators has been used in algorithms for decades in an ad-hoc manner - i.e., algorithms were based on minimal separators but also other specific features of particular graph classes. It was conjectured by Kloks *et al.* [90, 91], and proved by Bouchitté and Todinca [32, 33] that the property of having polynomially many minimal separators is sufficient to compute the treewidth.

A graph is called *chordal* if every cycle of length at least four has an edge connecting two nonconsecutive vertices of the cycle. Given a graph G, a *triangulation* H of G is a chordal supergraph of G. The treewidth of a graph G can be characterized then as the minimum, among all the triangulations H of G, of $\omega(H)-1$, where $\omega(H)$ is the size of a maximum clique in H. Clearly the optimal triangulation H of G (the one which the trewidth of G equals $\omega(H)-1$) is found among the *minimal triangulations* of G, where a triangulation H of G is minimal if no strict subgraph of H is a triangulation of G.

Bouchitté and Todinca [32] observed that the sets of vertices inducing a maximal clique in some minimal triangulation of a graph are objects with their own interest. They called them *potential maximal cliques*, and showed that the treewidth can be computed in polynomial time if the list of these objects is given in the input. More formally, let Π_G be the set of all potential maximal cliques of a graph G. The algorithm of Bouchitté and Todinca computes the treewidth in time $\mathcal{O}(|\Pi_G|^2 n)$. The main technique is a dynamic programing scheme over partial solutions cut by minimal separators and potential maximal cliques. The algorithm computes the treewidth at the same time that it constructs the corresponding minimal triangulation.

Later, Fomin, Kratsch and Todinca [64] improved the algorithm computing treewidth due to Bouchitté and Todinca. They obtained an algorithm computing treewidth in a time that is linear in the number of potential maximal cliques, and polynomial in the size of the input graph (i.e., $\mathcal{O}(|\Pi_G| \cdot n^c)$, for a constant $c > 0$). By a result due to Fomin, Todinca and Villanger [67], the number of potential maximal cliques of an arbitrary graph of size n is in $\mathcal{O}^*(1.7347^n)$. These two results imply that the treewdith can be computed in time $\mathcal{O}^*(1.7347^n)$, beating the bound $\mathcal{O}^*(2^n)$ obtained via brute force search.

Bouchitté and Todinca [33] also gave new characterizations of potential maximal cliques in terms of minimal separators. This new tool was used to show that the number of potential maximal cliques is bounded by a polynomial that depends on the number of minimal separators and the size of the graph. Moreover, they showed that the list of potential maximal cliques can be generated in the same bounds. Thus, the treewidth can be computed in polynomial time when the input graph belongs to a class with polynomially many minimal separators.

1.1.3 Maximum induced subgraph of treewidth t satisfying $\mathcal{P}$

In 2010, Fomin and Villanger [68] found a new surprising application of minimal separators and potential maximal cliques. Consider the following simple observation: for any subset of nodes F of a graph G, and any triangulation H_F of $G[F]$, there exists a triangulation H of G such that $H[F] = H_F$. They used this observation to show that, using a similar dynamic programing scheme that Bouchitté and Todinca used to compute the treewidth, one can found the maximum induced subgraph of treewidth t (where t is fixed), in a running time in $\mathcal{O}(|\Pi_G| \cdot n^{t+3})$, when Π_G is part of the input.

This new result showed that many NP-Hard problems, other than computing treewidth, are solvable in any class with few minimal separators. For example, a maximum independent set, or maximum induced forest are the problems consisting on finding a maximum induced subgraph of treewidth zero and one, respectively. Many of these problems already had polynomial time algorithms in particular classes with few minimal separators. However, those algorithms not only used the fact that the class contain polynomially many minimal separators, but also specific properties of the classes, like the intersection model.

In the dynamic programing scheme used to compute a maximum induced subgraph of treewidth t, the solution is found at the same time that is computed a minimal tree decomposition of it. Fomin, Todinca and Villanger used this fact in [67] to show that at the same time that the solution is constructed (an induced subgraph of treewidth at most t), an extra CMSO_2 property $\mathcal{P}$ can be checked. More formally, MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ is the problem consisting on find a maximum set of vertices F in a graph G, such that $G[F]$ is of treewidth at most t and satisfies property $\mathcal{P}$. The authors showed in [67] that this problem can be solved in polynomial time in graph classes with polynomially many minimal separators. This result extended even more the set of NP-hard problems that become polynomial in graph classes with polynomially many minimal separators. For example, the problem consisting in find a maximum induced tree can be solved in polynomial time thanks to this result, since it consists in finding a maximum induced graph of treewitdh one that is connected (here the property $\mathcal{P}$ express that $G[F]$ is connected).

The results presented in this part of the thesis are located in the above line of research. We slightly extend the algorithm of Fomin *et al.* [67] solving MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ to a bigger family of problems, but mostly enlarge this result into a parameterized version, with respect to several graph parameters.

1.1.4 Contribution

The contribution of this part of the thesis is presented in five chapters.

In Chapter 3 we give the basic definitions for this part of the thesis and the main results that are going to be the keys to understand the next chapters. We begin with tree decompositions and their relation with minimal triangulations. We will give some definitions and hints on counting monadic second order logic and regular properties. Then we describe Courcelle's Theorem together with the version of Borie, Parker and Tovey [31]. Later we present the definitions and main results involving minimal separators and potential maximal cliques, including their characterizations and enumeration algorithms. Finally, we sketch a proof of the algorithms used to compute the treewidth, the maximum induced subgraph of treewidth at most t, and its variants.

In Chapter 4 we present a result of combinatorial nature. We show that the number of minimal separators and the number of potential maximal cliques are bounded by a function of two well-known parameters: the vertex cover and the modular width. The *vertex cover* of a graph G, denoted $\text{vc}(G)$, is the size of a minimum vertex cover of G. The *modular width* $\text{mw}(G)$ of a graph G is the maximum degree of a prime node in the modular decomposition of G (See Chapter 2 for definitions). We show that the number of minimal separators in a graph G is at most $3^{\text{vc}(G)}$ and $\mathcal{O}^*(1.6181^{\text{mw}(G)})$, and the number of potential maximal cliques is $\mathcal{O}^*(4^{\text{vc}(G)})$ and $\mathcal{O}^*(1.7347^{\text{mw}(G)})$. Moreover, these objects can be listed within the same running times. Here the $\mathcal{O}^*$ notation suppresses polynomial factors in the size of the input. Pipelined with the algorithmic applications of minimal separators and potential maximal cliques, these bounds imply that the problems consisting in computing the treewidth, or find a maximum induced subgraph of treewidth at most t satisfying a property $\mathcal{P}$ can be solved in a running time that is *single exponential* in the number vertex cover and modular width of the graph. We finish this chapter giving another application, consisting in the computation of the *treedepth*.

Most of the results of this chapter were published in the 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2014) and were obtained in collaboration with Fedor Fomin, Mathieu Liedloff and Ioan Todinca [66].

Chapter 5 is about graphs that are *close* to having polynomially many minimal separators. For a polynomial poly we call $\mathcal{G}_{\text{poly}}$ the class of graphs G with at most $\text{poly}(n)$ minimal separators, where n is the size of G. For $k > 0$, we call $\mathcal{G}_{\text{poly}} + kv$ the class of graphs G, that contain a set $M \subseteq V(G)$ of size at most k, called modulator, such that $G - M \in \mathcal{G}_{\text{poly}}$. We showed that MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ on $\mathcal{G}_{\text{poly}} + kv$, parameterized by the size of the modulator k, is *fixed parameter tractable*, when the modulator is part of the input. Finally, we show that is $W[2]$-hard to compute a modulator. More precisely, we show that for any fixed polynomial $\text{poly}(n) = \Omega(n)$, the problem of determine if a graph belongs to $\mathcal{G}_{\text{poly}} + kv$ parameterized by k is $W[2]$-Hard.

Most of the results of this chapter were published in the 41st International Workshop of Graph-Theoretic Concepts in Computer Science (WG 2015) and were obtained in collaboration with Mathieu Liedloff and Ioan Todinca [102].

Chapter 6 is a more exploratory chapter. Our goal is to extend the results of Fomin *et al.* [67] solving MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$, to a wider family of problems. First, we give a combinatorial result that shows that for any graph G, all its odd powers have at most the number of minimal separators of G. This means that if for some polynomial poly a graph G belongs to $\mathcal{G}_{\text{poly}}$, then for all odd number $k \geq 1$, the graph G^k also belongs to $\mathcal{G}_{\text{poly}}$. Recall that the k-th power G^k of a graph G consists in the graph on the same vertex set than G, where two vertices are adjacent if they are at distance at most k in G. This combinatorial result implies that the problem d-Scattered Set (also called d-Independent Set), consisting in find a set of vertices that are pairwise at distance at least d (e.g., an Independent Set is a 2-Scattered Set), is polynomial time solvable for every even distance d. This problem was posed by Eto *et al.* in [59], where the authors show that the problem is NP-hard on chordal graphs for any odd $d \geq 3$ [59], and ask whether this problem is polynomial in chordal-bipartite graphs for even d. Later, we explore problems like Connected Vertex Cover or Independent Dominating Set in specific graph classes with polynomially many minimal separators.

The results of this chapter were published in the 42st International Workshop of Graph-Theoretic Concepts in Computer Science (WG 2016) and were obtained in collaboration with Ioan Todinca [112].

Chapter 7 concludes this part of the thesis proposing open problems and further research.

1.2 Part II: A study of the Broadcast Congested Clique Model

The second part of the thesis is about distributed algorithms. The main characteristic of a distributed system is that there are several autonomous processors acting in the system at any moment. The processors have access only to partial local information, but are able to communicate within each other in order to accomplish some task. Distributed systems are sometimes unavoidable, as in the case of problems handling the graph of Internet, other times they are used to improve the running time of a sequential algorithm parallelizing tasks.

There are several large and complex networks, such as the Internet, human society or even the brain, that can be modeled as distributed systems. Such networks can be represented by a graph, where each vertex is only allowed to communicate with its adjacent vertices. Even if each part acts locally, nature shows that systems are able to coordinate to give a global response.

In the $\mathcal{CONGEST}$ model, a set of n processors (called *nodes*) are connected in a graph G called *communication network*. Each vertex of G corresponds to a node, and each edge of the graph corresponds to a *communication channel* between two nodes. We assume that the nodes have unique identifiers, representable in $\mathcal{O}(\log n)$ bits (although for some authors consider *anonymous networks* [71]). The communication channels are *congested*, meaning that they can transmit at most b of bits per round. The value b is called the *bandwidth*. In literature, the bandwidth is normally restricted to be *logarithmic* in the size of the graph, i.e., only $\mathcal{O}(\log n)$ bits can be transmitted through each communication channel.

The initial knowledge of a node x is the set of edges (and possibly their weights) incident to x in G. The nodes have as task to compute a function of G. For example, a task for node x might consist in determining which of its incident edges belong to a maximum spanning tree of G. The task might also consist in a *verification* problem. In that case, the nodes receive as input their incident edges in another graph H, and the goal is to decide if G and H satisfy some property. For example, the nodes receive as input their incident edges in graphs G and H, and the task is to decide if G is a spanning tree of H. To accomplish their tasks, processors can communicate in synchronous rounds, sending at each round a message over each communication channel.

A protocol in the $\mathcal{CONGEST}$ model consists in the local algorithms that run on each node to produce the messages to be sent at each round. After each round of communication, the nodes decide to *halt* or use their local information and all the received messages to create new messages to be transmitted through the communication channels. A protocol finishes when every node halts. A protocol is correct if every vertex had correctly computed the function of G. For verification problems, a protocol is correct if every node accepts when G and H satisfy the desired property, and otherwise at least one vertex rejects. The running time of a protocol is the number of communication rounds until each processor halts. In this model, it is assumed that the slower part of the computation occurs in the transmission of information through the communication channels, so the running times of local computations are not considered. The *time of a protocol* is the maximum number of rounds before every node stops, in the worst case.

The $\mathcal{CONGEST}$ model mixes two important challenges in distributed systems, which are locality and congestion. It is interesting from a theoretical point of view to try to separate these two issues, in order to understand them better. The $\mathcal{LOCAL}$ model is a nice example of a model that isolates the locality issue. This model is very similar to the $\mathcal{CONGEST}$ model, except that the communication channels are unrestricted, i.e., vertices can pass any quantity of information through the communication channels. This model is quite well studied, and several positive and negative results exist both for the model with vertices with unique identifiers, as well as the anonymous case [119, 96, 70].

The other extreme is to isolate the congestion issue. Several bottlenecks that lead to lower bounds in the $\mathcal{CONGEST}$ model involve the diameter D of the graph of communication channels G. For example, the bound on the number of rounds required to compute a maximum spanning tree (MST) is $\Omega(\sqrt{n}/b+D)$, for graphs of diameter $D = \Omega(\log n)$ [129]. This bound is $\Omega(\sqrt[3]{n}/\sqrt{b})$ or $\Omega(\sqrt[4]{n}/\sqrt{b})$, when the input graph is restricted to be of diameter three or four, respectively [108]. A natural question is what happens with this bound D equals one, i.e., when the graph of communication channels is a clique.

1.2.1 The Congested Clique Model

The Congested Clique model, corresponds to the $\mathcal{CONGEST}$ model, but when the n processors are connected in a network of *complete communication channels*, i.e., G is a clique. This model was first presented by Lotker, Patt-Shamir, Pavlov, and Peleg [107], where the authors studied the problem of computing a minimum spanning tree (MST). They show a protocol in the Congested Clique that computes a MST in time $\mathcal{O}(\log \log n)$.

The motivation behind this model is not merely theoretical [76]. In some real-world applications, such as high-level algorithms running in the Internet, there is no way to control all the nodes of the graph in order to compute a protocol. In change, an *overlay* network is defined, where a smaller number of processors is spread over the network, each of them controlling a group of nodes. These processors run the distributed algorithm in their controlled group of nodes, communicating to the others the results of their local computations. Even if they are not necessarily adjacent in the original network, it is not unrealistic to suppose direct communication between any pair of them.

In the recent years, the research on the Congested Clique model has been quite active. Lenzen obtained in [100] a constant round protocol in the Congested Clique model for simultaneously routing n messages per vertex to their assigned destination nodes, as well as an $\mathcal{O}(1)$ protocol for sorting $\mathcal{O}(n^2)$ numbers, given that each vertex begins the algorithm knowing $\mathcal{O}(n)$ numbers. Independently Patt-Shamir and Teplitsky [118] showed a similar, but slightly weaker result on sorting in the Congested Clique model. Later Hegeman *et al.* [82, 81] provided constant and near-constant (expected) time protocols for problems such as computing a 3-ruling set, a constant-approximation to metric facility location, and (under some assumptions) a constant-factor approximations to the minimum spanning tree. More recently, Censor-Hillel *et al.* [38] showed that fast matrix multiplication can be performed in time $\mathcal{O}(n^{(1-2/\omega)})$, where $\omega < 2.3728639$ is the exponent in the running time of the best centralized matrix multiplication algorithm. This result implies that problems as counting the number of triangles, computing the girth, or computing an $(1 + o(1))$ approximation of all-pairs shortest paths, can be done in time $\mathcal{O}(n^{0.158})$ in the Congested Clique model. Recently Ghaffari *et al.* [76] obtained a randomized protocol in the Congested Clique model that computes a MST in time $\mathcal{O}(\log^*(n))$ with high probability.

What about lower bounds? Since the graph of communication channels is a complete graph, each vertex can send $\mathcal{O}(n \log n)$ bits per round, so the total number of bits sent in each communication round is $\mathcal{O}(n^2 \log n)$ bits. The enormous quantity of information transmitted at each round suggests that lower bounds based on communication complexity will not be helpful for proving lower bounds in the Congested Clique model. Recently, Druker *et al.* [57] showed that for any function f, a lower bound in the number of rounds in order to compute function f in the Congested Clique Model implies, roughly speaking, a lower bound in the complexity of the circuit computing f. In other words, a nontrivial lower bound for some explicit function f in the Congested Clique might have a major impact in other fields of Theoretical Computer Science.

1.2.2 The Broadcast Congested Clique

The extreme difficulty to obtain lower bounds in the Congested Clique model, motivates the study some restrictions, to limit its power and to understand better what makes lower bounds so hard. One possible

restriction is to weaken the unrealistic situation where in a single round a vertex may send $n-1$ different messages, i.e., one different message per communication edge. The *Broadcast Congested Clique* model is the Congested Clique model (which consequently is also called *Unicast Congested Clique* [57]), except that in each round, each vertex sends one single message to every other vertex in the graph.

Few fast protocols are known in the broadcast congested clique model. Becker *et al.* [13] show that in just one round every node can learn every edge of trees, planar graphs, or any graph of bounded degeneracy. More formally, the authors show that if the input graph is of degeneracy d, then there exists a one-round protocol with bandwidth $\mathcal{O}(d^2 \log n)$ such that, at the end of the round, each vertex is able to completely reconstruct the input graph. Reconstruction means that every vertex knows all the edges in the graph, and then can compute any function or parameter (even if computing the function is NP-Hard, since local computation is unbounded). There exist other one-round protocols, e.g., to decide if the input graph contains a fixed forest [57], or is a cograph [87].

Non-surprisingly, there are more results involving lower-bounds in the Broadcast Congested Clique. In [13] the authors show that there is no one-round protocol that detects if the input graph contains a triangle or diameter greater than three, unless some vertex communicates $\Omega(n)$ bits. In [87] it is shown that if H is a non-bipartite graph, then any one-round protocol detecting graphs that contain H as a subgraph or induced subgraph must have bandwidth $\Omega(n)$.

Drucker *et al.* [57] were the first to provide lower bounds in the multi-round Broadcast Congested Clique model. They showed, reducing to the disjointness problem in two player communication complexity, that detecting if the input graph contains a 4-cycle requires $\Omega(n/b)$ rounds of communication if $\mathcal{O}(b)$ bit are sent at each round. Recently Holzer *et al.* [85] showed that $\Omega(n/b)$ bits are required to compute all pairs shortest paths (APSP), or even a $(2-o(1))$ approximation. They also provide a $\mathcal{O}(\sqrt{n}\log n/b)$ round protocol computing an $(2+o(1))$-approximation of APSP.

Computing the connected components and a maximum spanning tree are fundamental problems in the Congested Clique model. A classical parallel algorithm computing a MST due to Borůvka [115], can be implemented in the Broadcast Congested Clique model to run in $\mathcal{O}(\log n)$ rounds with bandwidth $\mathcal{O}(\log n)$. It is unknown if this protocol is optimal. One of the problems that motivate this thesis is whether there exist a non-trivial one-round deterministic protocol in the Broadcast Congested Clique model that computes the connected components of the input graph.

In [1, 2] Ahn *et al.* provide a one-round randomized protocol in the Broadcast Congested Clique, computing a spanning forest of the input graph with high probability and poly-logarithmic bandwidth. Their protocol was in fact conceived for another related model called *dynamic graph streams*, but it is directly implementable in the Broadcast Congested Clique model. The spanning forest protocol uses an extremely beautiful technique called *linear sketches*, which is a technique to compress information through a linear function. The authors use this technique to simulate Borůvka's Algorithm in one-round with high probability.

One particularity of the existent randomized protocols, e.g., the one of Ahn *et al.* for connectivity, is the requirement of *public coins*, which means that the random bits generated in the protocol and used by the different processors are the same for every node. This situation contrasts with the *private coins* model, where the random bits are generated independently on each node. This part of the thesis explores the role of randomness in the Broadcast Congested Clique, and under certain conditions we manage to simulate randomized protocols deterministically.

1.2.3 Contribution

The contribution of this part of the thesis is presented in six chapters.

In Chapter 8, we begin giving the formal definitions and known results on the Broadcast Congested Clique model. We give the relevant distinctions in randomized protocols, and some definitions that take special relevance in one round protocols, like the distinction between public and private randomness. Later, we present the main graph problems that are going to be mentioned in the following chapters. Finally, we give an overview of the main tools and techniques to be used in our algorithms and lower-bounds.

In Chapter 9, we separate from the point of view of the bandwidth three different settings: deterministic protocols, randomized protocols with private coins, and randomized protocols with public coins.

The separation is based on the natural function TWINS (which returns 1 if and only if there are two vertices with the same neighborhood) and variants of it.

The results of this chapter were published in the 21st International Colloquium, on Structural Information and Communication Complexity (SIROCCO 2016) and were obtained in collaboration with Florent Becker, Ivan Rapaport and Ioan Todinca [15].

In Chapter 10 we give some constant-round protocols to recognize some small graph classes. First, we improve the result of [13] showing a deterministic protocol for detecting and reconstructing d-degenerate graphs with bandwidth $\mathcal{O}(d \cdot \log n)$. Later, we extend the results of Kari *et al.* [87] giving public-coins randomized one-round protocols to detect distance-hereditary graphs and graphs with bounded modular width. Finally, we provide sublinear protocols recognizing and reconstructing small classes of graphs (graphs classes with $2^{o(n^2)}$ graphs of size n, for example interval graphs).

In Chapter 11 we explore the problem of the detection of cycles (induced and as subgraphs) in the Broadcast Congested Clique model. We exhibit a deterministic one-round deterministic protocol that detects if the input graph has a cycle (induced or not) of length at most k with bandwidth $\mathcal{O}(n^{2/k} \log n)$ if k if k is an odd number and $\mathcal{O}(n^{2/(k-1)} \log n)$ if k is even. We prove that this result is tight for $k \geq 4$, up to logarithmic factors. The lower bound is valid even for randomized multi-round protocols, and even if each vertex, instead of being able to see only its immediate neighbors, is allowed to see up to distance $\lfloor k/4 \rfloor$. On the other hand, we show that if each node is allowed to see up to distance $\lfloor k/2 \rfloor + 1$, then a polylogarithmic bandwidth and two rounds are sufficient to detect if the input graph has long induced cycles. In particular, this implies that we can detect chordal graphs in two rounds with polylogarithmic bandwidth if the vertices can see up to distance two.

In Chapter 12 we discuss the connectivity problem, which was a main motivation to study the Broadcast Congested Clique model, and where the lack of lower bounds for one-round deterministic protocols is the biggest frustration of this research. We give a constant-round protocol that computes the connected components of the input graph in the Broadcast Congested Clique with sublinear bandwidth. We show that in the model where the vertices can see up to distance k, then the connected components can be computed in one round with bandwidth $\mathcal{O}(n^{1/k} \cdot \log n)$.

The results of this chapter were published as a brief announcement in the 2016 ACM Symposium on Principles of Distributed Computing (PODC 2016) and were obtained in collaboration with Ioan Todinca [112].

Chapter 13 concludes the thesis proposing open problems and further research.

Chapter 2

Preliminaires

In this chapter, we give some general definitions and results, relevant for both parts the thesis. We start giving some specific notations. We continue presenting a general overview of basic graph theory. Later, we present the definitions of some graph classes. Finally, we briefly give some definitions in parameterized complexity.

2.1 Notation

Let $n > 0$, we denote by $[n]$ the set $\{1, \ldots, n\}$. The cardinality of a set S, denoted $|S|$ is the number of elements in the set. For a function $f : \mathbb{N} \to \mathbb{N}$ we denote $\mathcal{O}(f)$ the set of functions $g : \mathbb{N} \to \mathbb{N}$ such that $\lim_{n\to+\infty} g(n)/f(n) < +\infty$. In other words, g belongs to $\mathcal{O}(f)$ if and only if there exists $c > 0$ and $n_0 > 0$ such that $g(n) \leq c \cdot f(n)$ for all $n > n_0$. The set $o(f)$ is the set of functions $g : \mathbb{N} \to \mathbb{N}$ that satisfy $\lim_{n\to+\infty} g(n)/f(n) = 0$. We call $\mathcal{O}^*(f)$ the set of functions $g : \mathbb{N} \to \mathbb{N}$ such that there exists a polynomial poly such that $g \in \mathcal{O}(f \cdot poly)$. The set $\Omega(f)$ is the set of functions g that satisfy $g \in \Omega(f)$ if and only if $f \in \mathcal{O}(g)$. We abuse notation and write $g(n) = \mathcal{O}(f(n))$ (resp. $g(N) = o(f(n))$, $g(n) = \Omega(f(n))$, $g = \mathcal{O}^*(f(n))$) to denote $g \in \mathcal{O}(f)$ (resp. $g \in o(f)$, $g \in \Omega(f)$, $g \in \mathcal{O}^*(f)$), where n is a dummy variable.

2.2 Graph Theory

Here we give a short overview of standard graph terminology. For a more complete introduction to graph theory we recommend for example the book by R. Diestel [53].

A *graph* will be denoted $G = (V, E)$, where V is the set of *vertices* and E is the set of *edges* of G. Conversely, $V(G)$ and $E(G)$ represent the set of vertices, respectively of edges, of a graph G. We denote by n the number of vertices and by m the number of edges. Two vertices $u, v \in V(G)$ such that $\{u, v\} \in E(G)$ is an edge of the graph G are called *adjacent* in G, and *incident* to edge $e = \{u, v\}$.

Let $G = (V, E)$ be a graph, $W \subseteq V$ and $F \subseteq \{\{x, y\} \mid x, y \in W\}$. The graph $H = (W, E)$ is called a *sub-graph* of G, and we denote this by $H \subseteq G$. If $F = \{\{x, y\} \in E \mid x, y \in W\}$ then H is called the *sub-graph of G induced by W*, and it is denoted $H = G[W]$. We denote $G - W$ the graph $G[V \setminus W]$. A graph $H = (V, F)$ such that $E \subset F$ is called a *super-graph* of G. A sub-graph H of G such that $V(H) = V(G)$ is called a *spanning* sub-graph of G.

The *neighborhood* of a vertex $x \in\subset V$, denoted $N_G(x) = \{y \in V \mid \{x, y\} \in E\}$, is the set of vertices adjacent to v in G. In the following, the subscript is omitted if clear from the context. We say that a vertex x *sees* a vertex subset $S \subseteq V$ (or vice-versa) if $N_G(x)$ intersects S. For a vertex set $S \subseteq V$, the *open* neighborhood of S, denoted by $N_G(S)$, is the set $\bigcup_{v\in S} N_G(v) \setminus S$. The *closed neighborhood* of S (resp. x), denoted $N_G[S]$ (resp. $N[x]$) is $N_G(S) \cup S$ (resp. $N_G(x) \cup \{x\}$). We denote by $d_G(x)$ the *degree* of a vertex x in graph G, which is just the number of edges incident to v.

Let $p > 1$. A set of different vertices $\{x_1, \ldots, x_p\}$ of G is called a *path* of length p from x_1 to x_p (also called *x_1, x_p-path*) if $\{x_i, x_{i+1}\}$ are edges of G for every $i \in [p-1]$. If $p > 2$ and $\{x_1, x_p\}$ is also an edge, then $\{x_1, \ldots, x_p\}$ is called a *cycle* of length p. In both cases, we call *chord* an edge that connects two nonconsecutive vertices in a path or a cycle.

Let $\text{dist}_G(x, y)$ denote the *distance* between vertices x and y, defined as the minimum number of edges of a xy-path. The *diameter* of a subset of vertices S in a graph G is defined as $\text{diam}(S) = \max_{x,y \in S} \text{dist}_G(x, y)$ We denote by $N_G^k[x]$ the set of vertices at distance at most k from x. Let also $N_G^k(x) = N_G^k[x] \setminus \{x\}$, and we call these sets the *closed* and *open neighborhoods at distance* k of x, respectively. Similarly, for a set of vertices $U \subseteq V$, we call the sets $N_G^k(U) = \bigcup_{u \in U} N_G^k(u) \setminus U$ and $N_G^k[U] = \bigcup_{u \in U} N_G^k[u]$ the open and closed neighborhoods at distance k of U, respectively. Note that $N_G(U) = N_G^1(U)$, respectively $N_G[U] = N_G^1[U]$. Again, the subscripts in these definitions are omitted if clear from the context.

The *girth* of a graph G is the shortest length of a cycle in G. A *Hamiltonian path (cycle)* in a graph G is a path (cycle) passing through all the vertices of G. A graph is *bipartite* if $V(G)$ can be partitioned in two subsets (A, B) such that every edge of $E(G)$ has one endpoint in A and the second in B. Partition (A, B) is also called *bipartition* of G. Equivalently, G is bipartite if and only if G does not have any cycle of odd length.

A graph G is called *connected* if for every pair of vertices x, y, there exists a x, y-path in G. A vertex set $X \subseteq V(G)$ is *connected* if the subgraph $G[X]$ is connected. A subset of vertices that induces a connected sub-graph of G, and is maximal by inclusion with this property is called a *connected component* of G.

A *forest* is graph without cycles. A *tree* is a connected forest. A tree T can be *rooted* at vertex $r \in V(T)$, which imposes on $V(T)$ natural *parent-child* and *ancestor-descendant* relations. A *spanning tree* T of a graph G is a tree that is a spanning subgraph of G, i.e., T is such that $V(T) = V(G)$ and $E(T) \subseteq E(G)$.

A *matching* M in a graph G is a set of edges that pairwise do not share endpoints. A *perfect matching* is a matching M where every vertex of the graph is an endpoint of an edge in M. An *independent set* I in a graph G is a subset of the vertex set $V(G)$ such that the vertices of I are pairwise non adjacent. A *clique* C in a graph G is a subset of the vertex set $V(G)$ such that the of C are pairwise adjacent. We denote by K_n the graph of size n that is a clique, and call it the *complete graph* of size n. We call $K_{n,m}$ the *complete bipartite graph*, which consists in a bipartite graph with parts of size n and m, which contain all the possible edges between the two parts.

A *vertex cover* X of a graph G is a subset of the vertex set $V(G)$ such that X covers the edge set $E(G)$, i.e., every edge of G has at least one endpoint in X. A *dominating set* D of a graph G is a subset of the vertex set $V(G)$ such that $N[D] = V(G)$, in words, every vertex of $V(G) \setminus D$ has a neighbor in D.

For a graph G and edge $e = \{x, y\} \in E(G)$, by contracting edge e we mean the following operation. We remove x and y from the graph, introduce a new vertex $z_{x,y}$, and connect it to all vertices x or y are adjacent to (if both x and y are connected to the same vertex w, then we just add one edge $\{z, w\}$, i.e., the graph stays *simple*). A graph H is a *minor* of G if we can obtain H from G performing a sequence of edge contractions, and vertex or edge deletions.

Two nodes $x, y \in G$ are called *twins* if they share the same neighborhood, i.e., if $N(y) \setminus \{x\} = N(x) \setminus \{y\}$. An *$a, b$-minimal separator* of a graph G is an inclusion minimal subset of the vertex set $V(G)$ such that a and b are in distinct connected components of $G - S$. We also say that S is a minimal separator if it is an a, b-minimal separator for some pair of vertices a and b. The set of all minimal separators of G is denoted Δ_G.

2.3 Graph classes and parameters

In the following, we define some graph classes and parameters.

2.3.1 Treewdith and cliquewidth

A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i, i \in I\}, \mathcal{T} = (I, E_T))$ where X_i are subsets of vertices of G, called the *bags of the decomposition*, and $\mathcal{T}$ is a tree such that:

1. $\bigcup_{i \in I} X_i = V$,
2. for each edge $\{x, y\}$ in G, there exists $i \in I$ such that $x, y \in X_i$,
3. for every $x \in V$, the set T_x of vertices $i \in I$ of $\mathcal{T}$ sucht that $x \in X_i$, induces a (connected) sub-tree in $\mathcal{T}$.

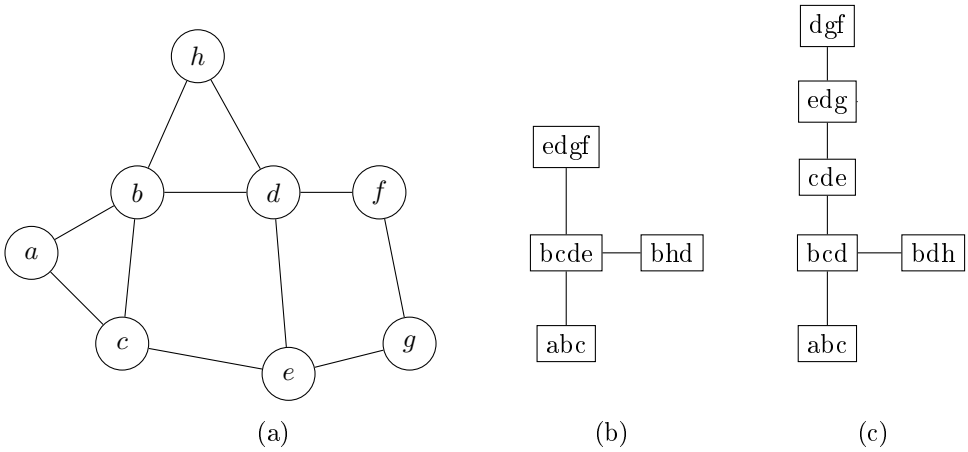


Figure 2.1 – Example of two tree decompositions of a graph. In left (a) is the example graph G, which is of treewdith 2. In the middle (b) is a tree decomposition of G of width 3. In right (c) there is a tree decomposition of G of minimum width.

The *width* of a tree-decomposition $(\{X_i, i \in I\}, \mathcal{T} = (I, E_T))$ is defined as $\max_{i \in I} |X_i| - 1$. The *treewidth* of G, denoted $\mathrm{tw}(G)$, is the minimum width, taken over all the possible tree-decompositions of G. The *length* of a tree-decomposition $(\{X_i, i \in I\}, \mathcal{T} = (I, E_T))$ is defined as the maximum distance between to nodes in a bag, i.e., $\max_{i \in I} \mathrm{diam}_G(X_i)$ (note that the distance is taken in G, not in $G[X_i]$). The *treelength* of G, denoted $\mathrm{tl}(G)$, is the minimum length, taken over all the possible tree-decompositions of G. A graph class $\mathcal{G}$ is said to have *bounded* treewidth (respectively treelength) if there exists $k > 0$ such that $\mathrm{tw}(G) \leq k$ (resp. $\mathrm{tl}(G) \leq k$) for all graphs $G \in \mathcal{G}$.

The *clique-width* of a graph G is the minimum number of labels needed to construct G using the following four operations:

1) The creation of a new vertex x with label i, denoted $\mathrm{id}_i(x)$.

2) The disjoint union of two labeled graphs G_1 and G_2, denoted $G \oplus H$.

3) Taking a labeled graph $G = (V, E)$ and two different labels i and j, and join by an edge each vertex with label i to each vertex with label j.

4) Taking a labeled graph $G = (V, E)$ and two labels i and j, and relabel with j all vertices labeled i .

The cliquewidth of a graph G is denoted $\mathrm{cw}(G)$. A graph class $\mathcal{G}$ has *bounded cliquewidth* if there exists $k > 0$ such that $\mathrm{cw}(G) \leq k$ for all $G \in \mathcal{G}$.

2.3.2 Chordal graphs and minimal triangulations

A graph H is *chordal* if it has no induced cycle with four or more vertices, i.e., every cycle of length at least four has a chord, which is an edge between two nonconsecutive vertices of the cycle. A classic characterization of chordal graphs due to Dirac [54] states that a graph is chordal if and only if all its minimal separators induce a clique.

By a classical result due to Gavril [75], every chordal graph G has a tree decomposition such that each bag of the tree decomposition is a maximal clique of G. Such tree is called a *clique-tree*, and the edges of the tree correspond to minimal separators which are the intersection of two or more maximal cliques (bags).

Let $G = (V, E)$ be an arbitrary graph. We say that $H = (V, F)$ is a *triangulation* of G if H is a chordal super-graph of G. If, moreover, H is inclusion-minimal for this property, then H is a *minimal triangulation* of G. The treewidth of G is also equal to the minimum integer w such that G has a (minimal) triangulation H, and each clique of H has at most $w + 1$ vertices.

2.3.3 Distance hereditary graphs, modular width and cographs

A graph $G = (V, E)$ is a *cograph* if G does not contain the four node path P_4 as an induced subgraph. The following is a characterization of cographs,

Proposition 1 ([10]). *A graph $G = (V, E)$ is a cograph if there exists an ordering $\{v_1, \ldots, v_n\}$ of the nodes of G such that for every i, vertex v_i has a twin in $G[\{v_i, \ldots, v_n\}]$. Such ordering is called a* twin-decomposition.

A graph $G = (V, E)$ is *distance-hereditary* if the distance between any two nodes of a same connected component is preserved in any connected induced subgraph that contains them.

The class of cographs is contained in the class of distance-hereditary graphs, as follows from the next characterization of the latter.

Proposition 2 ([10]). *A graph $G = (V, E)$ is distance-hereditary if there exists an ordering $\{v_1, \ldots, v_n\}$ of the nodes of G such that every i satisfies one of the following two conditions:*

1. *v_i has a twin in $G[\{v_i, \ldots, v_n\}]$.*
2. *v_i has degree at most one in $G[\{v_i, \ldots, v_n\}]$ (v_i is a pendant node).*

Such ordering is called a twin-pendant vertex decomposition.

A *module* of a graph $G = (V, E)$ is a set of nodes $M \subseteq V$ that is indistinguishable to every node outside M. Formally, for any node $x \in V \backslash M$, either $M \subseteq N(x)$ or M does not intersect $N(x)$. A module is *trivial* if it contains a single vertex, or the whole vertex set; otherwise it is *non-trivial*. A graph that only has trivial modules is called *prime*. A graph G is of modular width at most w if:

1. G has at most one node (the base case).
2. G is a disjoint union of two graphs of modular width at most w.
3. G is a *join* of two graphs of modular width at most w, i.e., G is obtained from two disjoint graphs of modular width at most w by taking their disjoint union and then adding all possible edges between these graphs.
4. the node set of G can be partitioned into $p \leq k$ modules $V_1, \ldots, V_p$ such that $G[V_i]$ is of modular width at most w, for all i, $1 \leq i \leq p$.

Note that if operation 4 is not used, we obtain the class of cographs. Therefore, the class of cographs is contained in the class of graph of modular width at most 2.

2.3.4 Bounded degeneracy

A graph G is called d-degenerate if each subgraph of G contains a vertex of degree at most d. This kind of graphs have the following characterization: a graph G is d-degenerate if the vertices of G can be ordered in a sequence $\{v_1, \ldots, v_n\}$ such that v_i has degree at most d in $G[\{v_i, \ldots, v_n\}]$. A class of graphs $\mathcal{G}$ has *bounded degeneracy* if there exists $d > 0$ such that G is d-degenerate for all $G \in \mathcal{G}$.

A graph G is *planar* if it can be drawn in the two dimensional euclidean plane without crossing edges. The celebrated theorem of Kuratowski (also atributed to Wagner) implies that a graph is planar if it does not contain $K_{3,3}$ or K_5 as a minor. Planar graphs are 5-degenerate.

2.4 Computational complexity

We briefly describe some concepts of computational complexity. For a more complete introduction we recommend the books by Sipser, Arora and Barak or Downey and Fellows [130, 7, 56].

Parameterized Complexity can be seen as multi-variable computational complexity, in the sense that we treat decision problems with two or more inputs. It deals with *parameterized problems*, which are decision problems that depend on a parameter k. In this context, the time complexity will be measured in terms of simultaneously $|x|$ and k. The *parameterized version* of a problem is a parameterized problem

where the role of the parameter will be normally obvious from the context. For example the VERTEX COVER parameterized by the size of the solution, is the problem consisting in decide if, for input graph G and parameter k, graph G contains a vertex cover of size at most k.

We say that a parameterized problem with parameter k is *fixed-parameter tractable* (FPT) if there is an algorithm that solves the problem in time $f(k)\cdot|x|^c$, where $c > 0$ and f is an arbitrary function and $|x|$ is the size of the input. Similarly, a parameterized problem with parameter k is in the class XP if there is an algorithm that solves the problem in time $|x|^{f(k)}$, where f is an arbitrary function. Clearly FPT is contained in XP. There is a notion of *parameterized reduction* between two parameterized problems. A parameterized problem $\mathcal{P}_1$ reduces to another parameterized problem $\mathcal{P}_2$ if there a function h that transforms an instance (x_1, k_1) of $\mathcal{P}_1$ into an instance $(x_2, k_2) = h(x_1, k_1)$ of problem $\mathcal{P}_2$ that satisfies (1) $h(x_1, k_1)$ can be computed in time $f(k_1)\cdot|x_1|^c$, where f is an arbitrary function and $c > 0$; (2) there exist a function g such that $k_2 = g(k_1)$ (i.e., parameter k_2 depends only in k_1); (3) (x_1, k_1) is a *Yes-instance* of $\mathcal{P}_1$ if and only if (x_2, k_2) is a *Yes-instance* of $\mathcal{P}_2$. The parameterized analogous to NP will be a hierarchy of parameterized problems, called $W[1], W[2]$, and so on, where $W[i] \subseteq W[i+1]$ for each integer $i > 0$. The definitions of these classes is quite technical, and we refer to the book of Downey and Fellows for precise definitions [56]. For the propose of this presentation is enough to point that the class FPT is contained in $W[1]$ (therefore in all $W[i]$, for $i \geq 1$) and it is conjectured that the containments of all the hierarchy is is strict, i.e., there are problems that belong to $W[i+1]$ that do not belong to $W[i]$, and also is conjectured that there are problems contained in $W[1]$ that are not contained in FPT.

Let $i \geq 1$ be an integer. The problems in $W[i]$ that most likely do not belong to FPT are the $W[i]$-Hard problems: a parameterized problem $\mathcal{P}$ is $W[i]$-Hard if and only if any other parameterized problem in $W[i]$ can be reduced to $\mathcal{P}$ via a parameterized reduction. As usual in complexity theory, a language is $W[i]$-Complete if it is contained in $W[i]$ and it is $W[i]$-Hard. Parameterized complexity can be used as a tool to distinguish intractable problems, that seem indistinguishable in terms of classical computational complexity. For example the problems VERTEX COVER, CLIQUE, HITTING SET are all NP-Hard, but in terms of parameterized complexity are FPT, $W[1]$-Complete, $W[2]$-Complete, respectively, in the versions of the problems that are parameterized by the size of the solutions [56, 48].

Another way to distinguish intractable problems is the research in *exact exponential algorithms*. The objective of such algorithms is to exactly solve NP-Hard (or even harder) problems in a running time that is *moderately exponential*. The goal is to look for techniques that beat trivial brute force algorithms. A classic example is INDEPENDENT SET, where a brute force algorithm runs in time $\mathcal{O}^*(2^n)$ in the worst case, while a very simple branching algorithm leads to a running time in $\mathcal{O}^*(1.41421^n)$ (in fact much faster algorithms can be obtained with more technical and involved branching algorithms, the current best exact exponential algorithm for Independent Set runs in time $\mathcal{O}^*(1.2002^n)$ [136]). We refer to the book of Fomin et Kratsch [63] for an excellent survey on exact exponential algorithms.

There are several NP-Hard problems for which the best exponential algorithm is the one obtained by a brute force algorithm. Is the case of SAT. In fact the *Exponential Time Hypothesis* (ETH) conjectures that there is no algorithm solving 3-SAT with running time in $2^{o(n)}$ (the actual statement of ETH is slightly different but implies it; we point out that ETH is less widely believed than other conjectures in computational complexity). The *Strong Exponential Time Hypothesis* states that, for any $\delta < 1$, there is no algorithm solving n-variable instances of SAT in time $\mathcal{O}(2^{\delta n})$.

Part I

Parameterized Algorithms via Potential Maximal Cliques

Chapter 3

Preliminaires

In this chapter, we give the main definitions and results to be used in this part of the thesis. We begin with a characterization of treewidth in terms of a *graph grammar*, similar than to one we used to define cliquewidth or modular width. We give the definition of graph properties expressible in monadic second order logic, and how it is related with graphs of bounded treewidth and the aforementioned grammar. Later, we give the main definition and characterizations of minimal separators and potential maximal cliques, together with sketches of their enumeration algorithms. Finally, we give some important algorithmic applications of minimal separators and potential maximal cliques.

3.1 Terminal recursive graphs and regular properties.

The research of Bruno Courcelle about graph grammars and monadic second-order logic can be interpreted as an effort to generalize to graphs, concepts of formal languages and universal algebra. A context-free grammar is a set of recursive rewriting rules, used to generate patterns of strings (languages). Roughly, a context-free grammar consists in a set of terminal symbols, nonterminal symbols, and rules for replacing nonterminal symbols by a string with other nonterminal or terminal symbols. In analogy with context-free grammars, Bauderon and Courcelle [11] defined the concept of *context-free graph-grammars*. These grammars roughly correspond to sets of graphs with terminal and nonterminal vertices, and the replacement of hyperedges of terminal vertices, by graphs containing other terminal and nonterminal vertices. Courcelle shows that for every graph property expressible in counting monadic-second order logic, the set of graphs definable by a context-free graph-grammar satisfying f property, is also definable by a context-free graph-grammar [44].

Graphs of bounded treewidth can be defined recursively, based on a graph grammar. Let w be a non-negative integer. A *w-terminal graph* is a triple (V, T, E), where (V, E) is a graph and T is a *totally ordered* subset of V, of size at most w. The vertices of T are called the *terminals* of the graph. Since T is totally ordered, we can speak of the ith terminal, for $i \leq |T|$.

The class of *w-terminal recursive graphs* is defined by the following operations. A *base graph* is a w-terminal recursive graph of the form (V, T, E) with $T = V$. Hence it has at most w vertices, all of them being terminals.

The *gluing* operation takes two disjoint w-terminal recursive graphs $G_1 = (V_1, T_1, E_1)$ and $G_2 = (V_2, T_2, E_2)$ and creates a new graph $G = glue_m(G_1, G_2)$, depending on a matrix m. The matrix m has two rows and at most $w + 1$ columns, with elements in $\{0, 1 \ldots, w\}$. The gluing operation takes the disjoint union of G_1 and G_2, and then identifies the terminal number i in G_1 (resp. in G_2) to terminal m_{1i} (resp. m_{2i}) in G. Each terminal of G_1 (resp. G_2) is mapped on at most one terminal of G. We take $m_{ji} = 0$, for $j \in \{1, 2\}$, if terminal number i in G_j does not exist or it is not mapped on any terminal of G.

The *forget* operation takes a w-terminal recursive graph $G_1 = (V, T, E)$ and creates the graph $G = forget_m(G_1)$ with $G = (V, T', E)$ such that T' is a subset of T. The matrix m has only one row and $|T|$ columns, and m_{1i} specifies as before that the ith terminal of G_1 is mapped on terminal m_{1i} of G. The mapping is injective, and if $m_{1i} = 0$ then the ith terminal of G_1 is removed, in G, from the set of terminals.

We point out that the number of possible different matrices and hence of different operations is bounded by a function on w.

Proposition 3 ([21, 67]). *Graph $H = (V, T, E)$ is $(w+1)$-terminal recursive if and only if there exists a tree decomposition of $G = (V, E)$, of width at most w, having T as one of its bags. Hence the grammar of $(w+1)$-terminal recursive graphs constructs exactly the graphs of treewidth at most w (see [21, 67]).*

3.1.1 Monadic Second-Order Logic

We use Counting Monadic Second Order Logic (CMSO_2), an extension of MSO_2, as a basic tool to express properties of vertex/edge sets in graphs.

The syntax of Monadic Second Order Logic (MSO_2) of graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, variables for vertices, edges, sets of vertices, and sets of edges, the quantifiers $\forall$, $\exists$ that can be applied to these variables, and the following five binary relations:

1. $u \in U$ where u is a vertex variable and U is a vertex set variable;
2. $d \in D$ where d is an edge variable and D is an edge set variable;
3. $\mathbf{inc}(d, u)$, where d is an edge variable, u is a vertex variable, and the interpretation is that the edge d is incident with the vertex u;
4. $\mathbf{adj}(u, v)$, where u and v are vertex variables and the interpretation is that u and v are adjacent;
5. equality of variables representing vertices, edges, sets of vertices, and sets of edges.

The MSO_1 is a restriction of MSO_2 in which one cannot use edge set variables (in particular the incidence relation becomes unnecessary). For example Hamiltonian Cycle is expressible in MSO_2 but not in MSO_1.

In addition to the usual features of monadic second-order logic, if we have atomic sentences testing whether the cardinality of a set is equal to q modulo r, where q and r are integers such that $0 \leq q < r$ and $r \geq 2$, then this extension of the MSO_2 (resp. MSO_1) is called the *counting monadic second-order logic* CMSO_2 (resp. CMSO_1). So essentially CMSO_2 (resp. CMSO_1) is MSO_2 (resp. MSO_1) with the following atomic sentence for a set S:

$\mathbf{card}_{q,r}(S) = \mathbf{true}$ if and only if $|S| \equiv q \pmod{r}$.

Informally, CMSO_2 allows logic formulae with quantifiers over vertices, edges, edge sets and vertex sets, and counting modulo constants. The CMSO_1 formulae are more restricted, we are not allowed to quantify over edge sets. We refer to [6, 44] and the book of Courcelle and Engelfriet [45] for a detailed introduction on different types of logic.

3.1.2 Regular properties

Let $\mathcal{P}(G, X)$ be a property assigning to each graph G and vertex subset X of G a Boolean value. We extend the gluing and forget operations to pairs (G, X) in the natural way (see, e.g., [31, 67]). In particular, when we perform a gluing on (G_1, X_1) and (G_2, X_2), the result is a pair (G, X) where X is obtained by the gluing of X_1 and X_2. Therefore the intersections of sets X_1 and X_2 with the terminals of G_1 and respectively G_2 must be coherent with the gluing, in the sense that if two terminals x_1 of G_1 and x_2 of G_2 are identified in G, then we either have $x_1 \in X_1$ and $x_2 \in X_2$, or we have $x_1 \notin X_1$ and $x_2 \notin X_2$.

Definition 1 (regular property). *Property $\mathcal{P}$ is called* regular *if, for any value w, we can associate a finite set $\mathcal{C}$ of* classes *and a* homomorphism function *h, assigning to each w-terminal recursive graph G and to each vertex subset X a class $h(G, X) \in \mathcal{C}$ such that:*

1. *If $h(G_1, X_1) = h(G_2, X_2)$ then $\mathcal{P}(G_1, X_1) = \mathcal{P}(G_2, X_2)$.*

2. *For each gluing operation* $glue_m$ *there exists a function* $\odot_{glue_m} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ *such that, for any two pairs* (G_1, X_1) *and* (G_2, X_2)*,*

$$h(glue_m((G_1, X_1), (G_2, X_2)) = \odot_{glue_m}(h(G_1, X_1), h(G_2, X_2))$$

and for each operation $forget_m$ *there is a function* $\odot_{forget_m} : \mathcal{C} \to \mathcal{C}$ *such that, for any pair* (G, X)*,*

$$h(forget_m(G, X)) = \odot_{forget_m}(h(G, X)).$$

The first condition separates the classes into *accepting* ones (i.e., classes $c \in \mathcal{C}$ such that $h(G, X) = c$ implies that $\mathcal{P}(G, X)$ is true) and *rejecting* ones (s.t. $h(G, X) = c$ implies that $\mathcal{P}(G, X)$ is false). In full words, the second condition states that, if we perform a *glue* (resp. *forget*) operation on two graphs (resp. one graph) and corresponding vertex subsets, the homomorphism class of the result can be obtained from the homomorphism classes of the graphs on which these operations are applied. Therefore, if a w-terminal recursive graph is given together with its expression in this grammar, and if moreover we know how to compute the classes of the base graph, then the homomorphism class of the whole graph, for a regular property $\mathcal{P}$, can be obtained by dynamic programming. We simply need to parse the expression from bottom to top and, at each node, we compute the class of the corresponding sub-expression thanks to the second condition of regularity. At the root, the property is true if and only if we are in an accepting class.

Proposition 4 (Borie, Parker, and Tovey [31], Courcelle [44])**.** *Any property* $\mathcal{P}(G, X)$ *expressible by a* CMSO_2 *formula is regular.*

Moreover, the result of Borie, Parker, and Tovey shows how to compute explicitly the set of classes, the homomorphism function for base graphs as well as the composition functions $\odot_{glue_m}$ and $\odot_{forget_m}$. Altogether, this provides an effective algorithm for checking the property in $O(n)$ time in graphs of bounded treewidth.

Proposition 5 (Courcelle [44])**.** *Any graph property* $\mathcal{P}$*, expressible by a* CMSO_2 *property* ϕ *can be checked in time* $f(\mathcal{P}, \text{tw}) \cdot n$*, where* tw *is the treewidth of the input graph and* f *is a computable function.*

The reader may try to express the homomorphism classes and function for his/her favorite CMSO_2 property. Let us consider the property "$G[X]$ is connected". We can choose, as homomorphism class $h((V, T, E), X)$, the partitions of T into subsets, such that each subset corresponds to the intersection of T with a component of $G[X]$. Observe that each such subset T_i of T is encoded by the indices of its elements in the ordered set T. Hence each homomorphism class will be a set of (disjoint) subsets of $\{1, \ldots, w\}$. We may assume w.l.o.g. that the homomorphism class $c = h(G, X)$, for $G = (V, T, E)$, encodes the intersection of X with the set of terminals. This is not explicitly required by the definition of regular properties, but it can be done since it only costs w bits to encode the number of the terminals contained in X. Therefore we assume there is a function $trm(c, T)$ that, given a homomorphism class c and an ordered set of terminals T returns the unique possible set $X \cap T$, over all pairs $(G = (V, T, E), X)$ mapped to c. Thanks to this function, when we will glue two terminal recursive graphs with their corresponding vertex subsets, we will be able to check that the gluing is coherent.

3.2 Minimal separators and potential maximal cliques

Let us recall the definition of a minimal separator: Let G be a graph and x, y two vertices of G. We say that $S \subset V(G)$ is a x, y-separator of G if x and y are in different connected components of $G - S$. Moreover, if S is inclusion-minimal among all x, y-separators, we say that S is a *minimal* x, y*-separator*. A vertex subset S is called a *minimal separator* of G if S is a minimal x, y-separator for some pair of vertices x and y. We call Δ_G the set of all minimal separators of a graph G. We remark that even if minimal separators are defined as inclusion minimal sets, it is possible to have a minimal separator contained in another one, as far as they separate different pairs of vertices. See, for example, Figure 3.1.

For a set of vertices X, we call $\mathcal{C}(X)$ the set of components C of $G - S$. We say that $C \in \mathcal{C}(S)$ is a *full component* associated to a minimal separator S, if $N(C) = S$, i.e., if every vertex of S is adjacent to some vertex in C. For example in Figure 3.1 sets $\{b\}$ and $\{d\}$ are full components associated to S, while $\{e\}$ and $\{f\}$ are not. The following proposition characterizes the minimal separators of a graph.

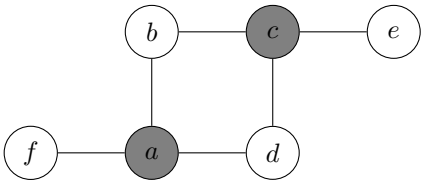


Figure 3.1 – Example a minimal separator, in this case the set $\{a, c\}$ is a minimal b, d-separator. Note that the set $\{c\}$ is also a minimal separator, e.g for pair e and b.

Proposition 6 (folklore)**.** *A vertex subset S of G is a minimal separator if $G - S$ has at least two full components associated to S. Moreover, S is a minimal x, y-separator if and only if x and y are in different full components associated to S.*

From this proposition, to test if a set S is a minimal separator we can simply compute the connected components of $G - S$ and test if two of them see the whole set S. Therefore, we have:

Corollary 1. *There is an $O(m)$ time algorithm testing if a given vertex subset S is a minimal separator of G.*

The first algorithms listing the minimal separators of a graph enumerate them by considering all the minimal separators between each pair of nonadjacent vertices. Since a single minimal separator may separate a big number of pair of nonadjacent vertices, this approach was somehow inefficient. Berry, Bordat and Cogis [16] improved those algorithms enumerating minimal separators *at the same time*. Their algorithm is based on the notion of *close separators*, and the definition of a *Galois lattice* representing all the minimal separators.

Proposition 7 (Berry, Bordat and Cogis [16])**.** *The set Δ_G of minimal separators of graph G can be listed in time $\mathcal{O}(n^3|\Delta_G|)$.*

Minimal separators behave well with respect to induced subgraphs, as shown by the following proposition.

Proposition 8. *Let G be a graph and $V' \subseteq V$ be a set of vertices of G. If S is a minimal x, y-separator of $G[V']$, then there is a minimal x, y-separator T of G, such that $T \cap V' = S$.*

This proposition implies that if G is a graph, a is a vertex of G, and S is a minimal separator of $G-\{a\}$, then either S or $S \cup \{a\}$ is a minimal separator of G. In particular this implies that $|\Delta_G| \geq |\Delta_{G[V']}|$ for every $V' \subseteq V(G)$.

We also have a way to construct a minimal separator associated to a given connected set of vertices.

Proposition 9 ([18])**.** *Let $G = (V, E)$ be a graph, C be a connected set of vertices, and let D be a component of $G - N[C]$. Then $N(D)$ is an a, b-minimal separator of G, for any $a \in C$ and $b \in D$.*

Recall that a *triangulation* of a graph $G = (V, E)$ is a chordal graph $H = (V, E')$ such that $E \subseteq E'$. Graph H is a *minimal triangulation* of G if for every edge set E'' with $E \subseteq E'' \subset E'$, the graph $F = (V, E'')$ is not chordal.

A set of vertices $\Omega \subseteq V$ of a graph G is called a *potential maximal clique* if there is a minimal triangulation H of G such that Ω is a maximal clique of H. We call Π_G the set of potential maximal cliques of a graph G.

Minimal separators and potential maximal cliques are deeply related, as it shows the following characterization of potential maximal cliques, due to Bouchitté and Todinca [32].

Proposition 10 ([32])**.** *Let $\Omega \subseteq V$ be a set of vertices of the graph $G = (V, E)$. We denote $\mathcal{S}(\Omega)$ the family of subsets of Ω such that $S \in \mathcal{S}(\Omega)$ if $S = N(C)$ for some $C \in \mathcal{C}(\Omega)$. Then Ω is a potential maximal clique of G if and only if*

1. *each $S \in \mathcal{S}(\Omega)$ is strictly contained in Ω;*
2. *the graph on the vertex set Ω obtained from $G[\Omega]$ by completing each $S \in \mathcal{S}(\Omega)$ into a clique is a complete graph.*

Moreover, if Ω is a potential maximal clique, then $\mathcal{S}(\Omega)$ is the set of minimal separators of G contained in Ω.

Another way of stating the second condition is that for any pair of vertices $u, v \in \Omega$, if they are not adjacent in G then there is a component C of $G - \Omega$ seeing both x and y.

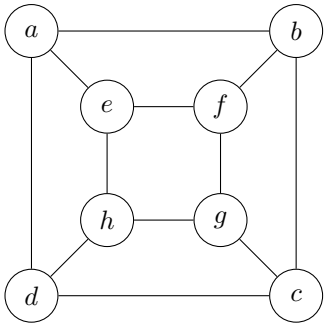


Figure 3.2 – Cube graph.

To illustrate Proposition 10, consider, e.g., the cube graph depicted in Figure 3.2. The set $\Omega_1 = \{a, e, g, c, h\}$ is a potential maximal clique and the minimal separators contained in Ω_1 are $\{a, e, g, c\}$ and $\{a, h, c\}$. Another potential maximal clique of the cube graph is $\Omega_2 = \{a, c, f, h\}$ containing the minimal separators $\{a, c, f\}$, $\{a, c, h\}$, $\{a, f, h\}$ and $\{c, f, h\}$.

Based on Propositions 6 and 10, one can easily deduce:

Corollary 2 (see e.g., [32]). *There is an $O(nm)$ time algorithm testing if a given vertex subset Ω is a potential maximal clique of G.*

Another observation that follows from Proposition 10 is the following.

Proposition 11 ([32]). *Let Ω be a potential maximal clique of G and let $S \subset \Omega$ be a minimal separator. Then $\Omega \setminus S$ is contained into a unique component C of $G - S$, and moreover C is a full component associated to S.*

For example in Figure 3.2, if we again consider the potential maximal clique $\Omega_1 = \{a, e, g, c, h\}$ and the minimal separator $S = \{a, e, g, c\}$ of G contained in Ω_1, then $\Omega_1 \setminus S$ is contained in the component $\{d, h\}$ of the graph $G - S$.

Bouchitté and Todinca also proposed in [33] an algorithm to enumerate the potential maximal cliques in polynomial time with respect to the number of potential maximal cliques and the size of the input graph. Their technique uses a particular family of potential maximal cliques, which have *active* separators.

Definition 2 ([33]). *Let $\Omega \subseteq V$ be a potential maximal clique of graph $G = (V, E)$, and let S be a minimal separator in $\mathcal{S}(\Omega)$. Consider now the graph G^+ obtained from G by completing into a clique all minimal separators in $\mathcal{S}(\Omega) \setminus \{S\}$.*

We say that S is an active separator *for Ω if Ω is not a clique in this graph G^+. A pair of vertices $x, y \in \Omega$ that are not adjacent in G^+ is called an* active pair. *Note that, by Proposition 10, we must have $x, y \in S$.*

Intuitively a minimal separator of a potential maximal clique Ω that is not active (that we call *inactive*) is a minimal separator that do not create edges in $G_{\mathcal{S}(\Omega)}$, which correspond to the graph where every minimal separator in $\mathcal{S}(\Omega)$ is completed into a clique. The following statement characterizes potential maximal cliques with active separators.

Proposition 12. *Let Ω be a potential maximal clique having an active separator $S \subset \Omega$, with an active pair $x, y \in S$. Denote by C the unique component of $G - S$ containing $\Omega \setminus S$. Then $\Omega \setminus S$ is a minimal x, y-separator in the graph $G[C \cup \{x, y\}]$.*

Again on the cube graph of Figure 3.2, for the potential maximal clique $\Omega_1 = \{a, e, g, c, h\}$, both minimal separators are active. E.g., for the minimal separator $S = \{a, e, g, c\}$ the pair $\{e, g\}$ is active.

Not all potential maximal cliques have active separators, as illustrated by the potential maximal clique $\Omega_2 = \{a, c, f, h\}$ of the same graph.

For counting and enumerating all potential maximal cliques of graph $G = (V, E)$, including the ones with no active separators, we apply the following statement.

Proposition 13 ([33]). *Let $G = (V, E)$ be a graph, let u be an arbitrary vertex of G and Ω be a potential maximal clique of G. Denote by $G - u$ the graph $G[V \setminus \{u\}]$. Then one of the following holds.*

1. *Ω has an active minimal separator S.*
2. *Ω is a potential maximal clique of $G - u$.*
3. *$\Omega \setminus \{u\}$ is a potential maximal clique of $G - u$.*
4. *$\Omega \setminus \{u\}$ is a minimal separator of G.*

Proposition 13 implies that the number of the potential maximal cliques Π_G of a graph G is polynomially bounded in the size of the graph n, the number of minimal separators of G, the number of minimal separators and potential maximal cliques of graph $G - u$, for every vertex u of G. It was proved that the number of potential maximal cliques is polynomially bounded in the number of its minimal separators and in the size of the graph.

Proposition 14 ([33]). *A graph G of size n has at most $n|\Delta_G|^2 + n|\Delta_G| + 1$ potential maximal cliques.*

Proposition 13 also leads to an algorithm that enumerates all the potential maximal cliques of the input graph. Intuitively, the algorithm fixes an arbitrary ordering $\{v_1, \ldots, v_n\}$ of the vertex set of the input graph G, and then uses Proposition 13 to compute the list potential maximal cliques of graph G_{i+1} from the active potential maximal cliques of G_{i+1} and the list of potential maximal cliques of graph G_i, where $G_i = G[\{v_1, \ldots, v_i\}]$, for $i \geq 1$. More details in [33].

Proposition 15 ([33]). *The potential maximal cliques of a graph can be listed in polynomial time in its size and the number of its minimal separators. More precisely, the potential maximal cliques of a graph are computable in $\mathcal{O}(n^2 m|\Delta_G|^2)$ time.*

3.2.1 Graph classes with *few* minimal separators

Let poly be some polynomial. We call $\mathcal{G}_{\text{poly}}$ the family of graphs such that $G \in \mathcal{G}_{\text{poly}}$ if and only if G has at most poly(n) minimal separators. We say that a class of graphs has *few minimal separators* if it is contained in $\mathcal{G}_{\text{poly}}$ for some polynomial poly. We also say that such a class has polynomially many minimal separators.

Note that from Proposition 14, the graphs in $\mathcal{G}_{\text{poly}}$ have also polynomially many potential maximal cliques. Moreover, the set of minimal separators Δ_G and the potential maximal cliques Π_G of these graphs can be listed in polynomial time.

Proposition 16 ([17, 33]). *For any polynomial* poly*, there is a polynomial-time algorithm enumerating the minimal separators and the potential maximal cliques of graphs on $\mathcal{G}_{\text{poly}}$.*

There are several structured and well defined graph classes are known to have polynomially many minimal separators. The first example is the family of chordal graphs, which have $\mathcal{O}(n)$ minimal separators: chordal graphs have at most n maximal cliques (which are also their potential maximal cliques), and all minimal separators of chordal graphs are the intersection of maximal cliques. This obviously implies that subclasses of chordal graphs like interval graphs or split graphs, have all $\mathcal{O}(n)$ minimal separators.

A graph G is called *weakly chordal* if G and its complement $\overline{G}$ have no induced cycle with more than four vertices. Weakly chordal graphs have $\mathcal{O}(n^2)$ minimal separators, and $\mathcal{O}(n^2)$ potential maximal cliques [18, 32]. This implies that graph classes contained in weakly chordal graphs but that are not chordal, also have polynomially many minimal separators. An example is the class of *chordal-bipartite* graphs, which are bipartite graphs where each cycle of length at least six have a chord (i.e., the longest induced cycles are of length four).

Other examples of graphs with polynomially many minimal separators are defined by *intersection models of geometric objects*. The class of *circular-arc graphs* are a generalization of interval graphs,

where each vertex is represented by an arc of a circle, and two vertices are adjacent if the corresponding arcs intersect. We may assume w.l.o.g. that, in the intersection model, no two chords (resp. no two arcs) share an endpoint. On the circle, we add a *scanpoint* between each two consecutive endpoints of the set of arcs. A *scanline* is a line segment between two scanpoints. Given an intersection model of a circular-arc graph G, Kloks *et al.* [92] showed that for any minimal separator S of G there is a scanline such that the vertices of S correspond exactly to the arcs intersecting the scanline. A potential maximal clique can be characterized as the set of vertices intersecting a triangle of scanpoints defining three scanlines [32]. Since the number of scanpoints is $2n$ and consequently the number of scanlines is $n(2n-1)$ we conclude that circular-arc graphs have $\mathcal{O}(n^2)$ minimal separators, and $\mathcal{O}(n^3)$ potential maximal cliques.

Another example of a class of graph defined by an intersection model in a circle, and which have polynomially many minimal separators, are *circle graphs*. A graph G is a *circle graph* if every vertex of the graph can be associated to a chord of a circle such that two vertices are adjacent in G if and only if the corresponding chords intersect. Again, we may assume w.l.o.g. that, in the intersection model, no two chords share an endpoint. We can define scanpoints and scanlines as before, where this time a *scanpoint* occurs between each two consecutive endpoints of the set of chords. Minimal separators and potential maximal cliques can be defined with respect to scanlines in the same way than for circular-arc graphs [92]. Therefore circle graphs have $\mathcal{O}(n^2)$ minimal separators, and $\mathcal{O}(n^3)$ potential maximal cliques.

Circle graphs can be generalized as *polygon-circle* graphs, where each vertex is defined as a polygon inscribed in a circle (i.e., the vertices of the polygon are points of the circle). As observed by Suchan [132], scanpoints and scanlines are defined for polygon circle graphs in the same way than for circle graphs. Therefore polygon-circle graphs have $\mathcal{O}(n^2)$ minimal separators, and $\mathcal{O}(n^3)$ potential maximal cliques.

The class of d-trapezoid graphs is another graph class that has polynomially many minimal separators. We give the definition of this family following [67]. Let $L_1, \ldots, L_d$ be d parallel lines in the plane. A d-trapezoid is the polygon obtained by choosing an interval l_i on every line L_i and connecting the left, respectively, right endpoint of I_i with the left, respectively, right endpoint of I_{i+1}. A graph is a d-trapezoid graph if it has an intersection model consisting of d-trapezoids between d parallel lines. Every d-trapezoid graph has at most $(2n-3)^{d-1}$ minimal separators [26]. The family of *permutation graphs* is a subclass of 2-trapezoid graphs, where each trapezoid is a line (the intervals l_i are reduced to a unique point).

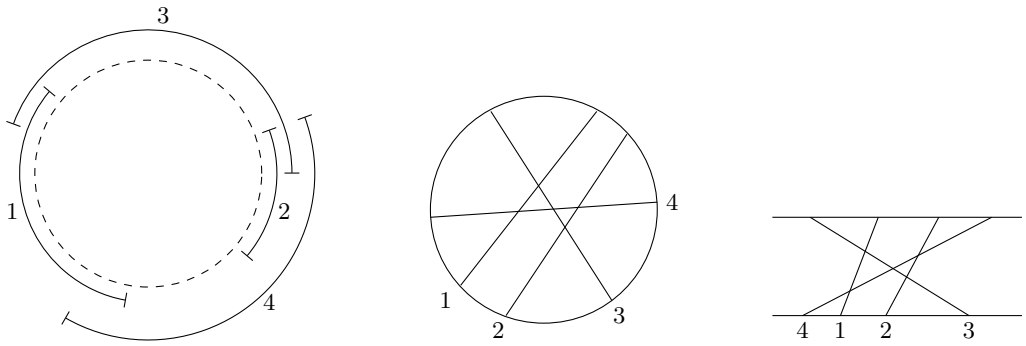


Figure 3.3 – Representations of graph $G = (\{1,2,3,4\}, \{\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$ as (a) arcs in a circle (b) chords in a circle (c) intersection of intervals between two parallel lines. Therefore G belong to the classes of circular-arc, circle and permutation graphs.

This table summarized the number of minimal separators and potential maximal cliques for the graph classes mentioned above, together with the corresponding references.

Class	**minimal separators**	**potential maximal cliques**
Weakly chordal	$\mathcal{O}(n^2)$ [32]	$\mathcal{O}(n^2)$ [32]
Chordal	$\mathcal{O}(n)$ [128]	$\mathcal{O}(n)$[128]
Polygon-circle	$\mathcal{O}(n^2)$ [132]	$\mathcal{O}(n^3)$ [132]
Circle	$\mathcal{O}(n^2)$ [92]	$\mathcal{O}(n^3)$ [92, 32]
Circular arc	$\mathcal{O}(n^2)$ [92]	$\mathcal{O}(n^3)$ [92, 32]
d-trapezoid	$\mathcal{O}(n^d)$ [27]	$\mathcal{O}(n^{d+2})$ [27, 65]

3.3 Algorithmic applications of minimal separators and potential maximal cliques

3.3.1 Computing treewdith

Remember that the *treewidth* of a graph $G = (V, E)$, denoted $\mathrm{tw}(G)$, is the minimum number k such that G has a triangulation $H = (V, E')$ of clique size at most $k+1$. The *minimum fill in* of G is the minimum size of F, over all (minimal) triangulations $H = (V, E \cup F)$ of G.

It is well known that computing the treewidth of a graph is NP-Hard [5], even for restricted classes of graphs, like graphs of bounded degree [29] or bipartite graphs [88]. However, for most of classes of polynomially many minimal separators cited in the last section, there exist polynomial-time algorithms computing the treewidth. It was conjectured in [90, 91] that the treewidth and the minimum-fill in should be tractable in polynomial-time for every class having polynomially many minimal separators. This conjecture was solved by Bouchitté and Todinca in [32], where they show that together with a graph G, the list of potential maximal cliques Π_G is given in the input, then the treewidth and the minimum-fill in can be solved in time $\mathcal{O}^*(|\Pi_G|)$. Together with Proposition 15, this result implies that Treewidth and Minimum Fill-In are polynomial time solvable in graphs with polynomially many minimal separators.

Proposition 17 ([32]). *Let Π_G denote the set of potential maximal cliques of graph G.* Treewidth *and* Minimum Fill-In *are solvable in $\mathcal{O}^*(|\Pi_G|)$ time, when Π_G is given in the input.*

The algorithm consists in a dynamic programming scheme over the *blocks* of the graph. Let $G = (V, E)$ be a graph, and S be a minimal separator of G. The pair (S, C) is called a *block* if C be a component of $G - S$ and $N(C) = S$. As a convention, we say that $(\emptyset, V)$ is a block of G. For a minimal separator S of G, call $\mathcal{B}(S)$ the set of components C of $G - S$ such that (S, C) is a block of G.

Let X be a set of vertices of a graph G, and call G_X the supergraph of G, where we *complete* X, i.e., we add an edge between each pair of nonadjacent vertices of X. Graph $G_S[S \cup C]$ is called a *realization* of the block (S, C). The treewidth of a graph G can be expressed as $tw(G_S[S \cup C])$, where $(S, C) = (\emptyset, V)$.

We show now how to express the treewidth of $G_S[S \cup C]$ in terms of the realizations of blocks contained in $S \cup C$. Let Ω be a potential maximal clique of a graph G. We say that (S, C) is a block associated to Ω if and only if (S, C) is a block where $S \in \mathcal{S}(\Omega)$ and C is a component of $G - \Omega$. For a potential maximal clique Ω of G, call $\mathcal{B}_G(\Omega)$ the set of blocks associated to Ω in G.

Proposition 18. *Let (S, C) be a block of graph G. A graph H is a minimal triangulation of $G_S[S \cup C]$ if and only if there is a potential maximal clique $\Omega \subseteq S \cup C$ of G such that $S \subset \Omega$, Ω is a clique in H, and for every block (S', C') associated to Ω in graph $G_S[S \cup C]$, the graph $H[S' \cup C']$ is a minimal triangulation of $G_{S'}[S' \cup C']$.*

As a corollary, we obtain the following expressions

$$\mathrm{tw}(G_S[S \cup C]) = \min_{S \subset \Omega \subseteq S \cup C} \mathrm{tw}(G_\Omega[S \cup C]), \text{ and}$$

$$\mathrm{tw}(G_\Omega[S \cup C]) = \max(|\Omega| - 1, \{\mathrm{tw}(G_{S'}[S' \cup C']) : (S', C') \in \mathcal{B}_{G_S[S \cup C]}(\Omega)\}).$$

Note that at the same time that we compute the treewidth, we find the corresponding minimal triangulation, as the realizations of blocks. We deduce the following expressions to compute the minimum fill-In $\mathrm{mfi}(G)$ of G (the minimum fill in of a graph G is the minimum number of edges to add to G in order to obtain a chordal supergraph).

$$\mathrm{mfi}(G) = \min_{S \in \Delta_G} \left(\{\mathrm{fill}(S) + \sum_{C \in \mathcal{C}(S)} \mathrm{mfi}(G_S[S \cup C])\} \right), \text{ and}$$

$$\mathrm{mfi}(G_S[S \cup C]) = \min_{S \subset \Omega \subseteq S \cup C} \left(\mathrm{fill}(\Omega) - \mathrm{fill}(S) + \sum_{(S', C') \in \mathcal{B}(\Omega)} \mathrm{mfi}(G_{S'}[S' \cup C'] \right),$$

where for a set of vertices X of G, the value of $\mathrm{fill}(X)$ is the number of pairs of nonadjacent vertices in $G[X]$.

These ideas can be translated in an algorithm that can be implemented in $\mathcal{O}(\Delta_G \cdot m + \Pi_G \cdot B_G \cdot n)$, where B_G is the set of blocks of G. However, doing a much careful counting in [65] the authors obtained that the algorithm runs in time $\mathcal{O}^*(\Pi_G)$. Moreover, it can be modified to work in the weighted versions of the problems.

Other graph parameters associated to minimal triangulations can be computed in graphs with a few minimal separators. The *treelength* of G is the minimum k such that there exists a minimal triangulation H, with the property that any two vertices adjacent in H are at distance at most k in graph G. In [105] Lokshtanov showed that Treelength can be solved in time $\mathcal{O}^*(|\Pi_G|)$ time, when Π_G is given in the input. We finish this section giving a proposition summarizing the previous result, for future reference.

Proposition 19. *Let Π_G denote the set of potential maximal cliques of graph G. The following problems are solvable in $\mathcal{O}^*(|\Pi_G|)$ time, when Π_G is given in the input:*

- (Weighted) Treewidth *[65, 28],*
- (Weighted) Minimum Fill-In *[65, 80],*
- Treelength *[105].*

Together with Proposition 15, we obtain that these problems are polynomially time solvable in graphs with polynomially many minimal separators. Moreover, the number of minimal separators and potential maximal cliques of an arbitrary graph can be enumerated by an algorithm running in a time that beats the brute force search.

Proposition 20 ([67, 69])**.** *Every n-vertex graph has $\mathcal{O}^*(\rho^n)$ minimal separators, where $\rho < 1.6181$ is the golden ratio, and $\mathcal{O}^*(1.7347^n)$ potential maximal cliques. Moreover, these objects can be enumerated within the same running times.*

This result combined with Proposition 19 gives $\mathcal{O}(1.7347^n)$ time algorithms for computing the treewidth, minimum fill-in and treelength.

3.3.2 Dynamic programming over minimal triangulations and potential maximal cliques

Minimal separators and potential maximal cliques have not been used only for computing treewidth and other parameters related to minimal triangulations. One may say that the main motivation to study the graph classes known to have polynomially many minimal separators, is because many NP-Hard problems were polynomial time solvable restricted to some of these classes. A natural question is whether these (or some of these) problems have polynomial time algorithms restricted to any class of graphs with a few minimal separators, rather than a particular class.

One positive result in this direction is due to Fomin and Villanger [68]. Consider the following problem.

Maximum Induced Subgraph of Treewidth at most t
Input: A graph G.
Task: Compute $F \subset V$ such that $G[F]$ has treewidth at most t and $\|F\|$ is maximum.

There are natural problems that can be expressed as the maximum induced forest of some fixed treewidth. For example Vertex Cover is equivalent to Independent Set, which consists in find a maximum induced subgraph of treewidth 0. The problem Feedback Vertex Set can be expressed as a Maximum Induced Forest, which is a maximum induced subgraph of treewidth 1. Fomin and Villanger [68] showed that for every fixed $t \geq 0$, problem Maximum Induced Subgraph of Treewidth at most t can be solved in polynomial time in graph classes with few minimal separators.

Proposition 21 ([68])**.** *For any fixed integer $t > 0$, problem* Maximum Induced Subgraph of Treewidth at most t *is solvable in $\mathcal{O}(|\Pi_G| \cdot n^{t+4})$ time, when Π_G is given in the input.*

Again, together with Proposition 20, this result $\mathcal{O}(1.7347^n)$ time algorithms for Maximum Induced Subgraph of Treewidth at most t on arbitrary graphs.

Later, Fomin, Todinca and Vilanger [67] extended this result to a larger family of problems. Fix an integer $t \geq 0$ and a CMSO_2 formula φ. Consider the problem of finding, in the input graph G, an induced

subgraph $G[F]$ together with a vertex subset $X \subseteq F$, such that the treewidth of $G[F]$ is at most t, the graph $G[F]$ together with the vertex subset X satisfy formula φ, and X is of maximum size under these conditions. This optimization problem is called MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$:

MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$
Input: A graph G.
Task: Compute

max	$\lvert X\rvert$
subject to	There is a set $F \subseteq V$ such that $X \subseteq F$;
	The treewidth of $G[F]$ is at most t;
	$(G[F], X) \models \varphi$.

Note that the formula φ has a free variable corresponding to the vertex subset X. For several examples, in formula φ the vertex set X is actually equal to F. E.g., even when φ only states that $X = F$, the obvious examples are for $t = 0$ the problem MAXIMUM INDEPENDENT SET PROBLEM, and for $t = 1$ the problem MAXIMUM INDUCED FOREST. If $t = 1$ and φ states that $X = F$ and $G[F]$ is a path we obtain the LONGEST INDUCED PATH problem. Still under the assumption that $X = F$, we can express the problem of finding the largest induced subgraph $G[F]$ excluding a fixed planar graph H as a minor, or the largest induced subgraph with no cycles of length $0 \mod l$, since these properties imply that the output solution has bounded treewidth (depending on $|H|$ and l, respectively). But X can correspond to other parameters, e.g., we can choose the formula φ such that $|X|$ is the number of connected components of $G[F]$. Based on this we can express problems like INDEPENDENT CYCLE PACKING, where the goal is to find an induced subgraph with a maximum number of components, and such that each component induces a cycle.

The result of [67] states that problem MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ can be solved in a running time of the type $|\Pi_G| \cdot n^{t+4} \cdot f(\varphi, t)$ where $|\Pi_G|$ is the number of potential maximal cliques of the graph, assuming that the set of all potential maximal cliques is also part of the input.

Proposition 22 ([67]). *For any fixed integer $t > 0$ and any fixed* CMSO_2 *formula φ, problem* MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ *is solvable in $\mathcal{O}(|\Pi_G| \cdot n^{t+4})$ time, when Π_G is given in the input.*

For sake of completeness, in the following[1] we will give a sketch of the proof of Proposition [67]. The first ingredient is the following observation.

Proposition 23 ([68]). *Let $G = (V, E)$ be a graph, $F \subseteq V$, and let H_F be a minimal triangulation of $G[F]$. There exists a minimal triangulation H_G of G such that $H_G[F] = H_F$. We say that H_G* respects *the minimal triangulation H_F of $G[F]$.*

Note that, for any clique Ω of H_G, we have that $F \cap \Omega$ induces a clique in H_F. In particular, if $\mathrm{tw}(G[F]) \leq t$ and the clique size of H_F is at most $t + 1$, then every maximal clique of H_G intersects F in at most $t + 1$ vertices.

The second ingredient is a dynamic programming scheme that we describe below. Recall that if S is a minimal separator of G, and C be a component of $G - S$ such that $N(C) = S$, then the pair (S, C) is called a *block*. Let Ω be a potential maximal clique such that $S \subset \Omega \subseteq S \cup C$. Then (S, C, Ω) is called a *good triple*. In the sequel, W denotes a set of at most $t + 1$ vertices.

Definition 3 (partial compatible solution). *Consider a block (S, C) and a good triple (S, C, Ω) of graph G. Let $W \subseteq S$ (resp. $W \subseteq \Omega$) a vertex set of size at most $t + 1$. Let c be a homomorphism class for property $\mathcal{P}$ on $(t+1)$-terminal recursive graphs. We say that a pair (F, X) is a* partial solution compatible with (S, C, W, c) *(resp. with (S, C, Ω, W, c)) if the following conditions hold :*

1. *$X \subseteq F$.*
2. *$F \subseteq S \cup C$ and $F \cap S = W$ (resp. $F \cap \Omega = W$).*
3. *$H = (F, W, E(G[F]))$ is a $(t + 1)$-terminal recursive graph, and the homomorphism class $h_{\mathcal{P}}(H, X)$ for property $\mathcal{P}$ is exactly c.*

[1] The following arguments were mainly taken from [102], which in its turn was highly based on [67].

4. *There is a minimal triangulation T_F of $G[F]$ and a minimal triangulation T_G of G respecting T_F, such that S is a minimal separator (resp. Ω is a maximal clique) of $T_{G'}$.*

With the same notations as above, let $\alpha(S, C, W, c)$ (resp. $\beta(S, C, \Omega, W, c)$) be the maximum size of X over all partial solutions (F, X) compatible with (S, C, W, c) (resp. (S, C, Ω, W, c)). The situation is depicted in Figure 3.4. For simplicity, we did not represent set X.

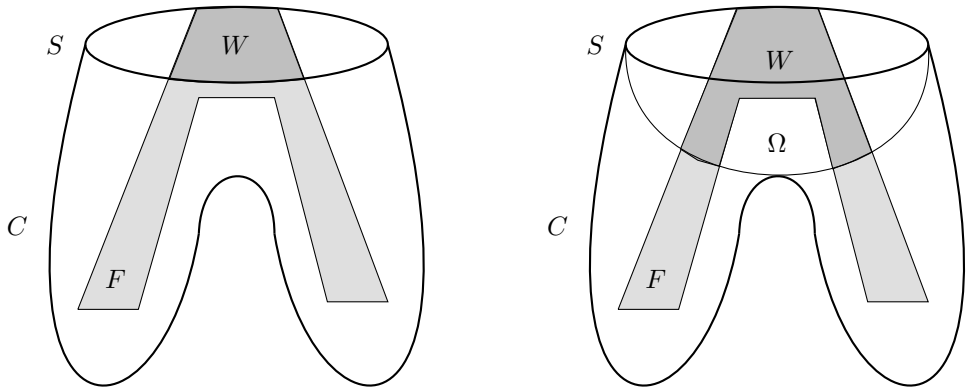


Figure 3.4 – (a) Partial solutions compatible with (S, C, W, c) (left), and (b) with (S, C, Ω, W, c) (right). Set F is depicted in grey. Note that set W corresponds to $F \cap S$ in the first case, and to $F \cap \Omega$ in the second case.

Algorithm 1: MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$

Input: graph $G = (V, E)$ together with the list of the potential maximal cliques of G
Output: sets $X \subseteq F \subseteq V(G)$ such that $G[F]$ has treewidth at most t, $\mathcal{P}(G[F], X)$ is true and, subject to these constrains, X is of maximum size

```
Order all blocks (S, C) of G by inclusion on S ∪ C. for all blocks (S, C) in this order do
    for all good triples (S, C, Ω) of G, all W ⊆ Ω of size ≤ t + 1 and all c ∈ C do
        if Ω = S ∪ C then Compute β(S, C, Ω, W, c) using Equation 3.1.;
        else Compute β(S, C, Ω, W, c) using Equations 3.3, 3.4, 3.5, and 3.6.;
    for all W ⊆ S of size ≤ t + 1 and all c ∈ C do
        Compute α(S, C, W, c) using Equation 3.2;
Compute an optimal solution using Equation 3.7;
```

The algorithm orders the blocks (S, C) by the size of $S \cup C$. It proceeds by dynamic programming on blocks (S, C) in this order, and on good triples (S, C, Ω), computing all possible values $\alpha(S, C, W, c)$ and $\beta(S, C, \Omega, W, c)$.

Base case: the good triple (S, C, Ω) is such that $\Omega = S \cup C$. In this case the only possible partial solutions (F, X) compatible with (S, C, Ω, W, c) correspond to base graphs $G[F]$, where all vertices are terminals. Indeed, we must have that $(W, W, E(G[F]))$ corresponds to a base $(t + 1)$-terminal graph. Thus set X is unique (or might not exist), because we must have $X = trm(c, W)$. For simplicity, we denote by $G[W]$ the base graph $(W, W, E(G[F]))$. We have:

$$\beta(S, C, \Omega, W, c) = \begin{cases} |X| & \text{if there is } X \subseteq W \text{ such that } h(G[W], X) = c \\ -\infty & \text{otherwise} \end{cases} \tag{3.1}$$

Each $\beta(S, C, \Omega, W, c)$, corresponding to a base case takes $\mathcal{O}(n)$ time, since we have to store the value in a table indexed by (S, C, Ω, c). For each good triple, we try at most n^{t+1} possible sets W. The number of good triples is $\mathcal{O}(n|\Pi_G|)$ so altogether these computations take $\mathcal{O}(n^{t+3}|\Pi_G|)$ time.

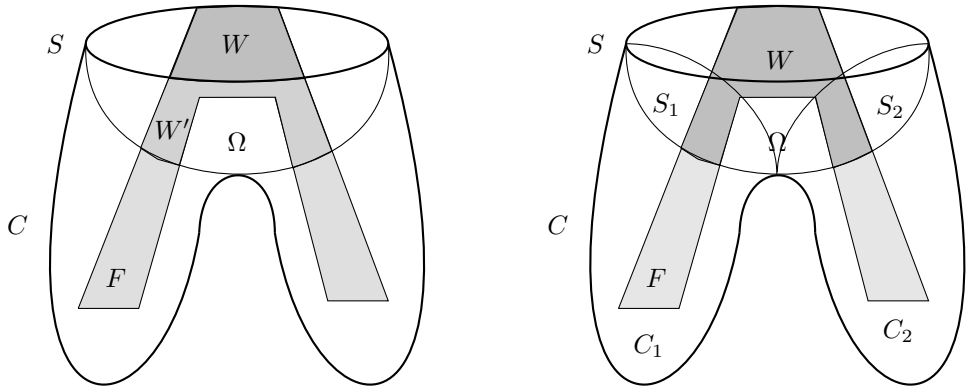


Figure 3.5 – Computing α form β (left), and β from α (right). Set M is not depicted for simplicity.

Computing α from β. We aim to compute $\alpha(S, C, W, c)$ from β values on good triples of type (S, C, Ω) (see also Figure 3.5 (left)).

Let (F, X) be an optimal solution compatible with (S, C, W, c). We denote by H the $(t+1)$-terminal recursive graph $(F, W, E(G[F]))$. By Definition 3, there is a minimal triangulation T_F of F and a minimal triangulation T_G of G respecting T_F, such that S is a minimal separator of T_G. By Proposition 11, there is a potential maximal clique Ω of G, inducing a maximal clique in T_G, and such that (S, C, Ω) form a good triple of G. Let $W' = F \cap \Omega$. Note that the graph $H' = (F, W', E(G[F]))$ is also a $(t+1)$-terminal recursive graph, and $(G[F], X)$ is a partial compatible solution for (S, C, Ω, W', c'), where $c' = h(H', X)$. Also note that $H = forget_{W' \to W}(H)$, where the $forget$ operation corresponds to the fact that the set of terminals W' is reduced to W. Therefore, we obtain

Proposition 24 ([67])**.**

$$\alpha(S, C, W, c) = \max \beta(S, C, \Omega, W', c'), \tag{3.2}$$

where the maximum is taken over potential maximal cliques Ω such that (S, C, Ω) is a good triple, all subsets $W' \subseteq \Omega$ of size at most $t+1$ such that $W' \cap S = W$ and all classes $c' \in \mathcal{C}$ such that $\odot_{forget_{W' \to W}}(c') = c$.

For computing all values $\alpha(S, C, W, c)$ from values $\beta(S, C, \Omega, W', c')$, we update the value of $\alpha(S, C, W, c)$ to the maximum between its previous value and $\beta(S, C, \Omega, W', c')$, if $\odot_{forget_{W' \to W}}(c') = c$. This cost $\mathcal{O}(n)$ for each quintuple (S, C, Ω, W', c'), and can be done in steps 3-4 of the algorithm. The number of quintuples is $\mathcal{O}(n^{t+2}|\Pi_G|)$, thus the total cost of these computations is $\mathcal{O}(n^{t+3}|\Pi_G|)$.

Computing β from α. Let (S, C, Ω) be a good triple of G. Denote by $C_1, \ldots, C_p$ the components of $G - \Omega$ contained in C, and let S_i be the neighborhood of C_i in G. Recall that we called (S_i, C_i) the blocks associated to Ω, and we suppose that they have been processed by the algorithm before (S, C). Our goal is to compute $\beta(S, C, \Omega, W, c)$. (see Figure 3.5 (right)). Let $W_i = W \cap S_i$, for all $i \in \{1, \ldots p\}$. We will use, two intermediate functions γ_i and δ_i.

Let $\delta_i(S, C, \Omega, W, c_i^+)$ denote $\max |X_i|$ over the partial solutions (F_i^+, X_i) compatible with (S, C, Ω, W, c_i^+) and such that $F_i^+ \subseteq \Omega \cap C_i$. Let H_i^+ and H_i be the $(t+1)$-terminal graphs $(F_i^+, W, E(G[F_i^+])$ and $(S_i \cup C_i, W_i, E(G[S_i \cup C_i])$, respectively. Note that H_i^+ is obtained by gluing H_i with the base graph $G[W]$, through the "canonical" gluing, obtained by identifying the vertices of S_i from both sides. Let $glue_{W_i;W}$ denote this gluing operation.

Proposition 25 ([67])**.**

$$\delta_i(S, C, \Omega, W, c_i^+) = \max \alpha(S_i, C_i, W_i, c_i) + |trm(c_W, W) \setminus trm(c_i, W_i)|, \tag{3.3}$$

where the maximum is taken over all $c_i, c_W \in \mathcal{C}$ s.t. $\odot_{glue_{W_i;W}}(c_i, c_W) = c_i^+$ and $c_W = h(G[W], X_W)$ for some $X_W \subseteq W$. Here $G[W]$ denotes the base graph with terminals W.

Notice the part $|trm(c_W, W) \setminus trm(c_i, W_)|$ in the formula, which avoids the overcounting of the vertices of $X_i \cap S_i$.

These partial solutions (F_i^+, X_i^+) corresponding to $\delta_i(S, C, \Omega, W, c_i^+)$ cannot be glued together in one step, since we are only allowed to glue two graphs at a time. Hence the need of the γ function which allows to add, one by one, the partial solutions to the gluing. Now let $\gamma_i(S, C, \Omega, W, c)$ denote the size of the optimal partial solution compatible with (S, C, Ω, W, c) and contained in $\Omega \cup C_1 \cdots \cup C_i$. So we only consider the first i components, the partial solution is the union of (F_1^+, X_1^+) to (F_i^+, X_i^+).

Proposition 26 ([67])**.**

$$\gamma_1(S, C, \Omega, W, c) = \delta_1(S, \Omega, C, W, c) \tag{3.4}$$

We then compute γ_i, for i from 2 to p as follows.

$$\gamma_i(S, C, \Omega, W, c) = \max \gamma_{i-1}(S, C, \Omega, W, c') + \delta_i(S, \Omega, C, W, c'') - |trm(c', W \cup F^M)|, \tag{3.5}$$

where the maximum is taken over all $c', c'' \in \mathcal{C}$ s. t. $\odot_{glue_{W;W}}(c', c'') = c$, where the gluing operation is the canonical gluing, the set of terminals for both arguments being W.

By definition of γ_p, we have

$$\beta(S, C, \Omega, W, c) = \gamma_p(S, C, \Omega, W, c). \tag{3.6}$$

For a fixed quadruple (S, C, Ω, W) computing the values $\beta(S, C, \Omega, W, c)$, given the values of function α in the smaller triples, takes $\mathcal{O}(n^2)$ time. The smaller blocks (S_i, C_i) can be listed in $\mathcal{O}(m)$ time. For each i, the computation of $\delta_i(S, \Omega, C, W, C_i^+)$ takes $\mathcal{O}(n)$ time, because we need to access the values $\alpha(S_i, W_i, C_i, c_i)$. Computing $\gamma_i(S, C, \Omega, W, c)$ from values γ_{i-1} and δ_i can be done in $\mathcal{O}(n)$ time for each i. Therefore the running time of the algorithm is the number of quadruples (S, C, Ω, W, c) times n^2, which is $\mathcal{O}(|\Pi_G| n^{t+4})$.

The global solution. It remains to retrieve the optimal solution for the algorithm.

Proposition 27 ([67])**.**

$$\max \alpha(\emptyset, V, \emptyset, c), \tag{3.7}$$

Where the maximum is taken over all accepting classes c, i.e., classes such that $h(G, X) = c$ implies that $\mathcal{P}(G, X)$:

Again from Proposition 15, we obtain that Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ is polynomially time solvable in graphs classes with polynomially many minimal separators. As remarked in [67], the only information required by this algorithm is the upper bound on the number of minimal separators in the specific graph class. For some specific graph class defined by an intersection model, the algorithms defined in such classes normally use the intersection model, which has to be computed beforehand. This algorithm produce correct output regardless of whether the input actually belongs to the specific class of graphs. If the number of minimal separators (and then potential maximal cliques) is bounded, the algorithm correctly solves the problem. Otherwise, the algorithm correctly outputs that the given input is not from the restricted graph class. Such types of algorithms were called robust by Raghavan and Spinrad [122]. For example, the recognition of d-trapezoid graphs is NP-complete [137, 67], however the algorithm of Proposition 22 either correctly solves the problem or outputs that the input graph is not d-trapezoid.

We re-emphasis that problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ generalizes many classical problems, e.g., Maximum Independent Set, Maximum Induced Forest, Longest Induced Path, Maximum Matching, Independent Cycle Packing, k-in-a-Path, k-in-a-Tree, Maximum Induced Subgraph With a Forbidden Planar Minor. More examples of particular cases are given in the following subsection (see also [67]).

3.3.3 More aplications

We give here several problems that are all known to be particular cases of Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ (see [67] proofs and more applications).

Let $\mathcal{F}_m$ be the set of cycles of length 0 (mod m). Let $\ell \geq 0$ be an integer. Our first example is the following problem.

MAXIMUM INDUCED SUBGRAPH WITH $\leq \ell$ COPIES OF $\mathcal{F}_m$-CYCLES
Input: A graph G.
Task: Find a set $F \subseteq V(G)$ of maximum size such that $G[F]$ contains at most ℓ vertex-disjoint cycles from $\mathcal{F}_m$.

MAXIMUM INDUCED SUBGRAPH WITH $\leq \ell$ COPIES OF $\mathcal{F}_m$-CYCLES encompasses several interesting problems. For example, when $\ell = 0$, the problem is to find a maximum induced subgraph without cycles divisible by m. For $\ell = 0$ and $m = 1$ this is MAXIMUM INDUCED FOREST.

For integers $\ell \geq 0$ and $p \geq 3$, the problem related to MAXIMUM INDUCED SUBGRAPH WITH $\leq \ell$ COPIES OF $\mathcal{F}_m$-CYCLES is the following.

MAXIMUM INDUCED SUBGRAPH WITH $\leq \ell$ COPIES OF p-CYCLES
Input: A graph G.
Task: Find a set $F \subseteq V(G)$ of maximum size such that $G[F]$ contains at most ℓ vertex-disjoint cycles of length at least p.

Next example concerns properties described by forbidden minors. Graph H is a *minor* of graph G if H can be obtained from a subgraph of G by a (possibly empty) sequence of edge contractions. A *model* M of minor H in G is a minimal subgraph of G, where the edge set $E(M)$ is partitioned into *c-edges (contraction edges)* and *m-edges (minor edges)* such that the graph resulting from contracting all c-edges is isomorphic to H. Thus, H is isomorphic to a minor of G if and only if there exists a model of H in G. For an integer ℓ a finite set of graphs $\mathcal{F}_{plan}$, containing a planar graph we define he following generic problem.

MAXIMUM IND. SUBGRAPH WITH $\leq \ell$ COPIES OF MINOR MODELS FROM $\mathcal{F}$
Input: A graph G.
Task: Find a set $F \subseteq V(G)$ of maximum size such that $G[F]$ contains at most ℓ vertex disjoint minor models of graphs from $\mathcal{F}_{plan}$

Even the special case with $\ell = 0$, this problem and its complementary version called the MINIMUM $\mathcal{F}$-DELETION, encompass many different problems.

Let $t \geq 0$ be an integer and φ be a CMSO-formula. Let $\mathcal{G}(t, \varphi)$ be a class of connected graphs of treewidth at most t and with property expressible by φ. Our next example is the following problem.

INDEPENDENT $\mathcal{G}(t, \varphi)$-PACKING
Input: A graph G.
Task: Find a set $F \subseteq V(G)$ with maximum number of connected components such that each connected component of $G[F]$ is in $\mathcal{G}(t, \varphi)$.

As natural sub cases studied in the literature we can cite INDEPENDENT TRIANGLE PAC KING or INDEPENDENT CYCLE PACKING.

The next problem is an example of *annotated version* of optimization problem MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$.

k-IN-A-GRAPH FROM $\mathcal{G}(t, \varphi)$
Input: A graph G, with k terminal vertices.
Task: Find an induced graph from $\mathcal{G}(t, \varphi)$ containing all k terminal vertices.

Many variants of k-IN-A-GRAPH FROM $\mathcal{G}(t, \varphi)$ can be found in the literature, like k-IN-A-PATH, k-IN-A-TREE, k-IN-A-CYCLE.

Chapter 4

Vertex cover, modular width and potential maximal cliques

In this chapter we give upper bounds on the number of *minimal separators* and *potential maximal cliques of graphs* with respect to two graph parameters, namely *vertex cover* (vc) and *modular width* (mw). We prove that for any graph, the number of its minimal separators is $\mathcal{O}^*(3^{\mathrm{vc}})$ and $\mathcal{O}^*(1.6181^{\mathrm{mw}})$, and the number of potential maximal cliques is $\mathcal{O}^*(4^{\mathrm{vc}})$ and $\mathcal{O}^*(1.7347^{\mathrm{mw}})$, and these objects can be listed within the same running times. Combined with Propositions 19 and 22), we deduce that a large family of problems, e.g., Treewidth, Minimum Fill-in, Longest Induced Path, Feedback vertex set and many others, can be solved in time $\mathcal{O}^*(4^{\mathrm{vc}})$ or $\mathcal{O}^*(1.7347^{\mathrm{mw}})$.

With slightly different techniques, we prove that the Treedepth problem can be also solved in single-exponential time, for both parameters.

Most of the results of this chapter were published in the 14th Scandinavian Symposium and Workshops (SWAT 2014) and were obtained in collaboration with Fedor Fomin, Mathieu Liedloff and Ioan Todinca [66].

4.1 Introduction

The main results of this chapter are of combinatorial nature: we show that the number of *minimal separators* and the number of *potential maximal cliques* of a graph are upper bounded by a function of its vertex cover and its modular with. More specifically, we prove the number of minimal separators in a graph G is at most $3^{\mathrm{vc}(G)}$ and $\mathcal{O}^*(1.6181^{\mathrm{mw}(G)})$, and the number of potential maximal cliques is $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ and $\mathcal{O}^*(1.7347^{\mathrm{mw}(G)})$. Moreover, these objects can be listed within the same running time bounds.

As we mentioned in the last chapter, minimal separators and potential maximal cliques have been used for solving several classical optimization problems, e.g., Treewidth, Minimum Fill-In [65], Longest Induced Path, Feedback Vertex Set or Independent Cycle Packing [67]. Pipelined with our combinatorial bounds, we obtain a series of algorithmic consequences in the area of FPT algorithms parameterized by the vertex cover and the modular width of the input graph. In particular, the problems mentioned above can be solved in time $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ and $\mathcal{O}^*(1.7347^{\mathrm{mw}(G)})$. These results are complementary in the sense that graphs with small vertex cover are sparse, while graphs with small modular width may be dense.

Vertex cover and modular width are strongly related to treewidth (tw) and cliquewidth (cw) parameters, since for any graph G we have $\mathrm{tw}(G) \leq \mathrm{vc}(G)$ and $\mathrm{cw}(G) \leq \mathrm{mw}(G) + 2$.

The celebrated theorem of Courcelle ([44] and Proposition 5) states that all problems expressible in Counting Monadic Second Order Logic (CMSO_2) can be solved in time $f(\mathrm{tw}) \cdot n$ for some function f depending on the problem. A similar result for cliquewidth [46] shows that all CMSO_1 problems can be solved in time $f(\mathrm{cw}) \cdot n$, if the clique-decomposition is also given as part of the input.

Typically function f is a tower of exponentials, and the height of the tower depends on the formula. Moreover Frick and Grohe [72] proved that this dependency on the treewidth or cliquewidth of an input graph G cannot be significantly improved in general.

Lampis [99] shows that the running time for CMSO_2 problems can be improved $2^{2^{\mathcal{O}(\mathrm{vc}(G))}} \cdot n$ when parametrized by vertex cover, but he also shows that this cannot be improved to $\mathcal{O}^*(2^{2^{o(\mathrm{vc}(G))}})$ (under the exponential time hypothesis). We are not aware of similar improvements for parameter modular width, but we refer to [73] for discussions on problems parameterized by modular width.

As we saw in Section 3.3.2, most of our algorithmic applications concern a restricted, though still large subset of CMSO_2 problems, but we guarantee algorithms that are single exponential in the vertex cover: $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ and in the modular width: $\mathcal{O}^*(1.7347^{\mathrm{mw}(G)})$. We point out that our result for modular width extends the result of [68, 67], who show a similar bound of $\mathcal{O}^*(1.7347^n)$ for the number of potential maximal cliques and for the running times for these problems, but parameterized by the number of vertices of the input graph. Thanks to our combinatorial bounds and Proposition 22, the problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ can be solved in times $\mathcal{O}(4^{\mathrm{vc}(G)}n^{t+c})$ and $\mathcal{O}(1.7347^{\mathrm{mw}(G)}n^{t+c})$, for some small constant c.

Naturally, our result can be applied in all other graph problems that can be solved in time $\mathcal{O}^*(|\Pi_G|)$ if the input graph is given together with the set of its potential maximal cliques. Examples of such problems are Treewidth, Minimum Fill-in [65], their weighted versions [23, 80] and several problems related to phylogeny [80], or Treelength [105]. Pipelined with our main combinatorial result, we deduce that all these problems can be solved in time $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ or $\mathcal{O}^*(1.7347^{\mathrm{mw}(G)})$. Recently Chapelle *et al.* [42] provided an algorithm solving Treewidth and Pathwidth in $\mathcal{O}^*(3^{\mathrm{vc}(G)})$, but those completely different techniques do not seem to work for Minimum Fill-in or Treelength. The interested reader may also refer., e.g., to [49, 60] for more (layout) problems parameterized by vertex cover. The best known parameterized algorithm deciding if a given n-vertex graph G is of treewidth at most t runs in time $\mathcal{O}(t^{t^3}n)$ [20]. The treewidth of a graph can be computed in time $\mathcal{O}(1.7347^n)$ [67].

Treedepth is a graph parameter that encountered a regain of interest in the area of sparse graphs, cf. the book of Nešetřil and Ossona de Mendes [116]. Actually the parameter used to be known under different names, e.g., *vertex ranking* [52]. A graph has treedepth/vertex ranking at most t if its vertices can be labeled from 1 to t, such that each path connecting vertices of the same label i contains an internal vertex with a label greater than i. The parameter is NP-hard to compute, but one can decide if the treedepth of an n-vertex graph is at most t in time $2^{\mathcal{O}(t^2)} \cdot n$ [124]. The treedepth can be also computed in time $\mathcal{O}(1.9602^n)$ [62].

Deogun *et al.* [52] provide polynomial algorithms computing the vertex ranking for several graph classes, using minimal separators. Based on their approach and new combinatorial boundss we show that the treedepth of a graph can also be computed in parameterized single-exponential time, when parameterized by the vertex cover or by the modular width of the input graph.

The rest of the chapter is organised as follows. Section 4.2 presents the combinatorial upper bounds on the number of minimal separators and potential maximal cliques, with respect to vertex cover. Section 4.3 provides similar bounds, with respect to modular width. Applications of these results are presented in Section 4.4. The results for treedepth do not rely directly on potential maximal cliques but on a different tool; they can be found in Section 4.5. Conclusion section discusses further research directions.

4.2 Relations to vertex cover

A vertex subset W is a *vertex cover* of G if each edge has at least one endpoint in W. Note that if W is a vertex cover, then $V \setminus W$ induces an *independent set* in G, i.e., $G - W$ contains no edges. We denote by $\mathrm{vc}(G)$ the size of a minimum vertex cover of G. The parameter $\mathrm{vc}(G)$ is called the *vertex cover number* or simply (by a slight abuse of language) the *vertex cover* of G. After a long sequence of improvements, the current fastest parameterized algorithm for Vertex Cover is the algorithm of Chen, Kanj, and Xia, running in time $\mathcal{O}(1.2738^{\mathrm{vc}(G)} + n\,\mathrm{vc}(G))$ [43]. However, for our purposes even a weaker result will do the job.

Proposition 28 (folklore)**.** *There is an algorithm computing the vertex cover of the input graph G in time $\mathcal{O}^*(2^{\mathrm{vc}(G)})$.*

Let us show that any graph G has at most $3^{\mathrm{vc}(G)}$ minimal separators.

Lemma 1. *Let $G = (V, E)$ be a graph, W be a vertex cover and $S \subseteq V$ be a minimal separator of G. Consider a three-partition (D_1, S, D_2) of V such that both D_1 and D_2 are formed by a union of components*

of $G - S$, and both D_1 and D_2 contain some full component associated to S. Denote $D_1^W = D_1 \cap W$ and $D_2^W = D_2 \cap W$.

Then $S \setminus W = \{x \in V \setminus W \mid N(x)$ intersects both D_1^W and $D_2^W\}$.

Proof. Let $C_1 \subseteq D_1$ and $C_2 \subseteq D_2$ be two full components associated to S. Let $x \in S \setminus W$. Vertex x must have neighbors both in C_1 and C_2, hence both in D_1 and D_2. Since $x \notin W$ and W is a vertex cover, we have $N(x) \subseteq W$. Consequently x has neighbors both in D_1^W and D_2^W.

Conversely, let $x \in V \setminus W$ s.t. $N(x)$ intersects both D_1^W and D_2^W. We prove that $x \in S$. By contradiction, assume that $x \notin S$, thus x is in some component C of $G - S$. Suppose w.l.o.g. that $C \subseteq D_1$. Since $N(x) \subseteq C \cup N(C)$, we must have $N(x) \subseteq D_1 \cup S$. Thus $N(x)$ cannot intersect D_2—a contradiction. □

Theorem 1. *Any graph G has at most $3^{\mathrm{vc}(G)}$ minimal separators. Moreover the set of its minimal separators can be listed in $\mathcal{O}^*(3^{\mathrm{vc}(G)})$ time.*

Proof. Let W be a minimum size vertex cover of G. For each three-partition (D_1^W, S^W, D_2^W) of W, let $S = S^W \cup \{x \in V \setminus W \mid N(x)$ intersects D_1^W and $D_2^W\}$. According to Lemma 1, each minimal separator of G will be generated this way, by an appropriate partition (D_1^W, S^W, D_2^W) of W. Thus the number of minimal separators is at most $3^{\mathrm{vc}(G)}$, the number of three-partitions of W.

These arguments can be easily turned into an enumeration algorithm, we simply need to compute an optimum vertex cover then test, for each set S generated from a three-partition, if S is indeed a minimal separator. The former part takes $\mathcal{O}^*(2^{\mathrm{vc}(G)})$ time by Proposition 28, and the latter takes polynomial time for each set S using Corollary 2. □

Observe that the bound of Theorem 1 is tight up to a constant factor. Indeed consider the watermelon graph $W_{k,3}$ formed by k disjoint paths of three vertices plus two vertices u and v adjacent to the left, respectively right ends of the paths (see Figure 4.1). Note that this graph has vertex cover $k + 2$ (the minimum vertex cover contains the middle of each path and vertices u and v) and it also has 3^k minimal u, v-separators, obtained by choosing arbitrarily one of the three vertices on each of the k paths.

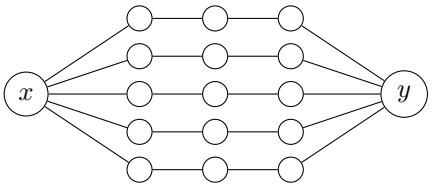


Figure 4.1 – The watermelon graph $W_{6,3}$.

We now extend Theorem 1 to a similar result on potential maximal cliques. Let us first focus on potential maximal cliques having an active separator (see Definition 2 and Proposition 12). We give a result similar to Lemma 1, showing that such a potential maximal clique can be determined by a certain partition of the vertex cover W of G.

Lemma 2. *Let $G = (V, E)$ be a graph and W be a vertex cover of G. Consider a potential maximal clique Ω of G having an active separator $S \subseteq \Omega$ and an active pair $x, y \in S$. Let C be the unique connected component of $G - S$ intersecting Ω and let D_S be the union of all other connected components of $G - S$. Denote by D_x the union of components of $G - \Omega$ contained in C, seeing x, by D_y the union of components of $G - \Omega$ contained in C not seeing x.*

Now let $D_S^W = D_S \cap W$, $D_x^W = D_x \cap W$ and $D_y^W = D_y \cap W$.

Then one of the following holds:

1. *There is a vertex $t \in \Omega$ such that $\Omega \setminus S = N(t) \cap C$.*
2. *There is a vertex $t \in \Omega$ such that $\Omega = N[t]$.*
3. *A vertex $z \notin W$ is in Ω if and only if*
   (a) *z sees D_S^W and $D_x^W \cup D_y^W$, or*

(b) z does not see D_S^W but is sees $D_x^W \cup \{x\}$, $D_y^W \cup \{y\}$ and $D_x^W \cup D_y^W$.

Proof. Note that D_x, D_y, D_S and Ω form a partition of the vertex set V.

We first prove that any vertex $z \notin W$ satisfying conditions 3a or 3b must be in Ω.

Consider first the case 3a when z sees D_S^W and $D_x^W \cup D_y^W$. So z sees D_S and C; we can apply Lemma 1 to partition (D_S, S, C) thus $z \in S$. Consider now the case 3b when z sees $D_x^W \cup D_y^W$, $D_x \cup \{x\}$ and $D_y \cup \{y\}$ but not D_S^W. Again by Lemma 1 applied to partition (D_S, S, C), vertex z cannot be in S. Since z has a neighbor in $D_x \cup D_y$, we have $z \in C$. Let $H = G[C \cup \{x, y\}]$ and $T = \Omega \cap C$ (thus we also have $T = \Omega \setminus S$). Recall that T is an x, y-minimal separator in H by Proposition 12. By definition of set D_x, we have that $D_x \cup \{x\}$ is exactly the component of $H - T$ containing x. Note that $D_y \cup \{y\}$ is the union of the component of $H - T$ containing y and of all other components of $H - T$ (that no not see x nor y). By applying Lemma 1 on graph H, with vertex cover $(W \cap C) \cup \{x, y\}$ and with partition $(D_x \cup \{x\}, T, D_y \cup \{y\})$ we deduce that $z \in T$.

Conversely, let $z \in \Omega \setminus W$. We must prove that either z satisfies conditions 3a or 3b, or we are in one of the first two cases of the Lemma. We distinguish the cases $z \in S$ and $z \in T$. When $z \in S$, by Lemma 1 applied to partition (D_S, S, C), z must see D_S and C. If z sees some vertex in $C \setminus \Omega$, we are done because z sees $D_x^W \cup D_y^W$ so we are in case 3a. Assume now that $N(z) \cap C \subseteq \Omega$, we prove that actually $N(z) \cap C = \Omega \cap C = T$, so we are in case 1. Assume there is $u \in T \setminus N(z)$. By Proposition 10, there must be a connected component D of $G - \Omega$ such that $z, u \in N(D)$. Since $u \in C$, this component D must be a subset of C, so $D \subseteq C \setminus \Omega$. Together with $z \in N(D)$, this contradicts the assumption $N(z) \cap C \subseteq \Omega$.

It remains to treat the case $z \in T$. Clearly $z \in C$ cannot see D_S because S separates C from D_S. We again take graph H, with vertex cover $(W \cap C) \cup \{x, y\}$, and apply Lemma 1 with partition $(D_x \cup \{x\}, T, D_y \cup \{y\})$. We deduce that z sees both $D_x^W \cup \{x\}$ and $D_y^W \cup \{y\}$. Assume that z does not see $D_x^W \cup D_y^W$. So $N(z) \cap C \setminus \Omega = \emptyset$ thus $N[z] \subseteq \Omega$. If Ω contains some vertex $u \notin N[z]$, no component of $G - \Omega$ can see both z and u (because $N(z) \subseteq \Omega$), contradicting Proposition 10. We conclude that either z sees $D_x^W \cup D_y^W$ (so satisfies condition 3b) or $\Omega = N[z]$ (thus we are in the second case of the Lemma). □

Theorem 2. *Every graph G containss $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ potential maximal cliques with active separators. Moreover the set of its potential maximal cliques with active separators can be listed in $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ time.*

Proof. The number of potential maximal cliques with active separators satisfying the second condition of Lemma 2 is at most n, and they can all be listed in polynomial time by checking, for each vertex t, if $N[t]$ is a potential maximal clique.

For enumerating the potential maximal cliques with active separators satisfying the first condition of Lemma 2, we enumerate all minimal separators S using Theorem 1, then for each $t \in S$ and each of the at most n components C of $G - S$ we check if $S \cup (C \cap N(t))$ is a potential maximal clique. Recall that testing if a vertex set is a potential maximal clique can be done in polynomial time by Corollary 2. Thus the whole process takes $\mathcal{O}^*(3^{\mathrm{vc}(G)})$ time, and this is also an upper bound on the number of listed objects.

It remains to enumerate the potential maximal cliques with active separators satisfying the third condition of Lemma 2. For this purpose, we "guess" the sets D_S^W D_x^W, D_y^W as in the Lemma and then we compute Ω. More formally, for each four-partition $(D_S^W, D_x^W, D_y^W, \Omega^W)$ of W, we let $\Omega^{\overline{W}}$ be the set of vertices $z \notin W$ satisfying conditions 3a or 3b of Lemma 2, and we test using Corollary 2 if $\Omega = \Omega^W \cup \Omega^{\overline{W}}$ is indeed a potential maximal clique. By Lemma 2, this enumerates in $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ all potential maximal cliques of this type. □

For counting and enumerating all potential maximal cliques of graph $G = (V, E)$, including the ones with no active separators, we apply the same ideas as in [33], based on Proposition 13.

Theorem 3. *Any graph G has $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ potential maximal cliques. Moreover the set of its potential maximal cliques can be listed in $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ time.*

Proof. Let $(v_1, \ldots, v_n)$ be an arbitrary ordering of the vertices of V. Denote by G_i the graph $G[\{v_1, \ldots, v_i\}]$ induced by the first i vertices, for all $i, 1 \leq i \leq n$. Let $k = \mathrm{vc}(G)$. Note that for all i we have $\mathrm{vc}(G_i) \leq k$. Actually, if W is a vertex cover of G, then $W_i = W \cap \{v_1, \ldots, v_i\}$ is a vertex cover of G_i. In particular, by Theorems 1 and 2, each G_i has at most 3^k minimal separators and $\mathcal{O}^*(4^k)$ potential maximal cliques with active separators.

For $i = 1$, graph G_1 has a unique potential maximal clique equal to $\{v_1\}$.

For each i from 2 to n, in increasing order, we compute the potential maximal cliques of G_i from those of G_{i-1} using Proposition 13. Observe that $G_{i-1} = G_i - v_i$. We initialize the set of potential maximal cliques of G_i with the ones having active separators. This can be done in $\mathcal{O}^*(4^k)$ time by Theorem 2. Then for each minimal separator S of G_i we check if $\Omega = S \cup \{v_i\}$ is a potential maximal clique of G_i and if so we add it to the set. This takes $\mathcal{O}^*(3^k)$ time by Theorem 1 and Corollary 2. Eventually, for each potential maximal clique Ω' of G_{i-1}, we test using Corollary 2 if Ω' (resp. $\Omega' \cup \{v_i\}$) is a potential maximal clique of G_i. If so, we add it to the set of potential maximal cliques of G_i. The running time of this last part is the number of potential maximal cliques of G_{i-1} times nm. Altogether, it takes $\mathcal{O}^*(4^k)$ time.

By Proposition 13, this algorithm covers alls cases and thus lists all potential maximal cliques of G_i. Hence for $i = n$ we obtain all potential maximal cliques of G, and they have been enumerated in $\mathcal{O}^*(4^k)$ time. □

4.3 Relations to modular width

Recall that a *module* of graph $G = (V, E)$ is a set of vertices W such that, for any vertex $x \in V \setminus W$, either $W \subseteq N(x)$ or W does not intersect $N(x)$. For the reader familiar with the modular decompositions of graphs, the modular width $\mathrm{mw}(G)$ of a graph G is the maximum size of a prime node in the modular decomposition tree. Equivalently,

graph G is of modular width at most k if:

1. G has at most one vertex (the base case).
2. G is a disjoint union of graphs of modular width at most k.
3. G is a *join* of graphs of modular width at most k. I.e., G is obtained from a family of disjoint graphs of modular width at most k by taking the disjoint union and then adding all possible edges between these graphs.
4. The vertex set of G can be partitioned into $p \leq k$ modules $V_1, \ldots, V_p$ such that $G[V_i]$ is of modular width at most k, for all $i, 1 \leq i \leq p$.

The modular width of a graph can be computed in linear time, using e.g., [134]. Moreover, this algorithm outputs the algebraic expression of G corresponding to the grammar above.

Let $G = (V, E)$ be a graph with vertex set $V = \{v_1, \ldots, v_k\}$ and let $M_i = (V_i, E_i)$ be a family of pairwise disjoint graphs, for all i, $1 \leq i \leq k$. Denote by H the graph obtained from G by replacing each vertex v_i by the module M_i. I.e., $H = (V_1 \cup \cdots \cup V_k, E_1 \cup \cdots \cup E_k \cup \{ab \mid a \in V_i, b \in V_j \text{ s.t. } v_iv_j \in E\})$. We say that graph H has been obtained from G by *expanding* each vertex v_i by the module M_i.

A vertex subset W of H is an *expansion* of vertex subset W_G of G if $W = \cup_{v_i \in W_G} V_i$. Given a vertex subset W of H, the *contraction* of W is $\{v_i \mid V_i \text{ intersects } W\}$.

Lemma 3. *Let S be a minimal y, z-separator of H, for $y, z \in V_i$. Then $S \cap V_i$ is a minimal separator of M_i and $S \setminus V_i = N_H(V_i)$.*

Proof. Note that all vertices of $N_H(V_i)$ are in $N_H(y) \cap N_H(z)$, by construction of graph H and the fact that y and z are in the same module V_i. Therefore $N_H(V_i)$ must be contained in S. Let $S_i = S \cap V_i$. Since $H[V_i] = M_i$, we have that S_i separates z and y in graph M_i. Assume that S_i is not a minimal y, z-separator of M_i, so let $S_i' \subsetneq S_i$ be a minimal y, z-separator in graph M_i. We claim that $S_i' \cup N_H(V_i)$ is a y, z-separator in H. Indeed each y, z-path of H is either contained in V_i (in which case it intersects S_i') or intersects $N_H(V_i)$. In both cases, it passes through $S_i' \cup N_H(V_i)$, which proves the claim. Since $S_i' \cup N_H(V_i)$ is a subset of S and S is a y, z-minimal separator of H, the only possibility is that $S = S_i' \cup N_H(V_i)$. This proves that $S \cap V_i$ is a minimal separator of M_i and $S \setminus V_i = N_H(V_i)$. □

Lemma 4. *Let S be a minimal separator of H. Assume that some V_i intersects S, but is not contained in S.*

Then V_i intersects all full components of $H - S$ associated to S. In particular $S \cap V_i$ is a minimal separator in M_i and $S \setminus V_i = N_H(V_i)$.

Proof. Let $x \in V_i \cap S$ and $t \in V_i \setminus S$. By Proposition 6, there are at least two full components of $H - S$, associated to S. Let C be one of them, not containing t. Let z be a neighbor of x in C, we prove that $z \in V_i$. If $z \notin V_i$, then $z \in N_H(V_i)$, and since V_i is a module in H we also have $z \in N_H(t)$. This contradicts the fact that t and z are in different components of $H - S$. It remains that $z \in V_i$. By applying the same argument for z instead of t, it follows that V_i intersects each full component D of $H - S$ and moreover x has a neighbor in $D \cap V_i$.

By Proposition 6, S is a minimal y, z-separator in H, for some $y, z \in N_H(x) \cap V_i$. The rest follows by Lemma 3. □

Lemma 5. *Let S be a minimal separator of H. Then one of the following holds*

1. *S is the expansion of a minimal separator S_G of G.*
2. *There is $i \in \{1, \ldots, k\}$ such that $S \cap V_i$ is a minimal separator of M_i and $S \setminus V_i = N_H(V_i)$.*

Proof. Assume there is a set V_i intersecting S but not contained in it. By Lemma 4, $S \cap V_i$ is a minimal separator of M_i and $S \setminus V_i = N_H(V_i)$. Hence we are in the second case of the Lemma.

Otherwise, for any V_i intersecting S, we have $V_i \subseteq S$. Thus S is the expansion of a vertex subset S_G of G, formed exactly by the vertices v_i of G such that V_i intersects S. Let C and D be two full components of $H - S$ associated to S and let $a \in C$, $b \in D$. Recall that, by Proposition 6, S is a minimal a, b-separator of H. Let V_k be the set containing a and V_l the set containing b. Consider first the possibility that $k = l$. Then, by Lemma 3, S satisfies the second condition of this lemma, for $i = k = l$. (This case may occur when M_k is disconnected and $S = N_H(V_k)$.)

Finally, we consider the case $k \neq l$. We prove that S_G is a minimal v_k, v_l-separator of G. Consider a v_k, v_l path of G. If this path does not intersect S_G in G, then there is a path from a to b in $H - S$, obtained by replacing each vertex v_j of the path by some vertex of V_j (v_k and v_l are replaced by a and b respectively). This would contradict the fact that S separates a and b in H. Therefore S_G is indeed a v_k, v_l-separator in G. Assume that S_G is not minimal among the v_k, v_l-separators of G, and let $v_j \in S_G$ such that $S_G \setminus \{v_j\}$ separates v_k and v_l in G. We claim that $S \setminus V_j$ also separates a from b in H. By contradiction, assume there is a path from $a \in C \cap V_k$ to $b \in D \cap V_l$ in H, avoiding $S \setminus V_j$. By contracting, on this path, all vertices belonging to a same V_i into vertex v_i, we obtain a path (or a connected subgraph) joining v_k to v_l in G. This contradicts the fact that all such paths should intersect $S_G \setminus \{v_j\}$. Therefore S_G is a minimal separator of G. □

Lemma 5 provides an injective mapping from the set of minimal separators of H to the union of the sets of minimal separators of G and of the graphs M_i. Therefore we have:

Corollary 3. *The number of minimal separators of H is at most the number of minimal separators of G plus the number of minimal separators of each M_i.*

We now aim to prove a statement equivalent of Corollary 3, for potential maximal cliques instead of minimal separators.

Lemma 6. *Let Ω be a potential maximal clique of H, and let $\Omega_G = \{v_i \mid V_i \text{ intersects } \Omega\}$. Assume that Ω is the expansion of Ω_G, i.e., $\Omega = \cup_{v_i \in \Omega_G} V_i$. Then Ω_G is a potential maximal clique of G.*

Proof. We prove that Ω_G satisfies, in graph G, the conditions of Proposition 10. For the first condition, let C_G be a component of $G - \Omega_G$ and let $S_G = N_G(C_G)$. Assume that S_G is not strictly contained in Ω_G, hence $S_G = \Omega_G$. Let C be the expansion of C_G in H and note that $N_H(C)$ is the expansion of $N_G(C_G)$, thus $N_H(C) = \Omega$. If C_G is formed by at least two vertices, since $G[C_G]$ is connected then so is $H[C]$. Therefore, in graph H, we have $N_H(C) = \Omega$ and C is a component of $H - \Omega$. But this contradicts the first condition of Proposition 10 applied to the potential maximal clique Ω of H. In the case that C_G is formed by a unique vertex v_k, its expansion $C = V_k$ might not induce a connected subset in H (if M_k is disconnected). But it is sufficient to consider a connected component V'_k of $H[V_k]$, and again this is also a component of $H - \Omega$ with the property that its neighborhood in H is the whole set Ω, contradicting Proposition 10 applied to Ω.

For the second condition of Proposition 10, let $v_j, v_k \in \Omega_G$ such that $v_j v_k$ is not an edge of G. Let $a \in V_j$ and $b \in V_k$. These vertices are non-adjacent in H, so by Proposition 10 applied to the potential maximal clique Ω of H there must be a component C of $H - \Omega$ seeing both a and b. Consider an a, b-path

in $H[C \cup \{a, b\}]$. The contraction of this path contains a v_j, v_k-path in G, whose internal vertices are not in Ω_G. This proves that v_j and v_k are in the neighborhood of a same component of $G - \Omega_G$, thus Ω_G satisfies the second condition of Proposition 10. □

Lemma 7. *Let Ω be a potential maximal clique of H, and assume that there is some set V_i that intersects Ω but is not contained in Ω. Then $\Omega \cap V_i$ is a potential maximal clique of M_i and $\Omega \setminus V_i = N_H(V_i)$.*

Proof. Let V_i be a vertex set that intersects Ω, but is not contained in Ω. We distinguish two cases.

Case 1. There is a minimal separator $S \subseteq \Omega$ of H, such that S intersects V_i.

By Lemma 4, $S \cap V_i$ is a minimal separator of M_i and V_i intersects all full components of $H - S$ associated to S. Let C be the unique component of $H - S$ intersecting Ω; recall that it exists and moreover it is full w.r.t. S, by Proposition 11. Then, by Lemma 4, C also intersects V_i. Also by Lemma 4, $S \setminus V_i = N_H(V_i)$. We claim that actually $C \subseteq V_i$ and C is also a full component of $M_i - S_i$. Recall that $S \setminus V_i = N_H(V_i)$ separates in graph H the vertices of V_i from the rest of the graph. Since C intersects V_i, $H[C]$ is connected and $N_H(V_i)$ separates V_i from all other vertices, we must have $C \subseteq V_i$. Since $H[C]$ is connected, so is $M_i[C]$, thus C is contained in some component of $M_i - S_i$. But each such component is also a component of $H - S$, hence C is both a component of $H - S$ and of $M_i - S_i$. In particular $\Omega \cap C \subseteq V_i$.

It remains to prove that $\Omega_i = \Omega \cap V_i$ is a potential maximal clique of M_i. By the above observations, we also have $\Omega_i = \Omega \setminus N_H(V_i)$. We show that Ω_i satisfies, in graph M_i, the conditions of Proposition 10. Let D be a component of $M_i - \Omega_i$. Observe that D is also a component of $H - \Omega$ and let $T = N_{M_i}(D)$. Either D is a component of $M_i - \Omega_i$ disjoint from C, or it is contained in C. In the former case, T is a subset of S_i, hence T is a strict subset of Ω_i (since S_i is itself a strict subset of Ω_i by Proposition 10 applied to potential maximal clique Ω of H). In the later case, if $T = \Omega_i$, note that $N_H(D) = \Omega$ because $\Omega \setminus V_i = N_H(V_i)$ is also contained in the neighborhood of D in H. This contradicts Proposition 10 applied to potential maximal clique Ω of H.

For the second condition, let $x, y \in \Omega_i$, non-adjacent in M_i. Then there is a component F of $H - \Omega$ seeing, in graph H, both x and y (by Proposition 10 applied to Ω). Since this component sees V_i, it must be contained in V_i. So F is also a component of $M_i - \Omega_i$ seeing both x and y in M_i, which concludes our proof for this case.

Case 2. There is no minimal separator $S \subseteq \Omega$ of H, intersecting V_i.

Let us prove that, in this case, for any $x \in \Omega_i$ we have that $\Omega = N_H[x]$. By Proposition 11, if x has a neighbor outside Ω, hence in some component D of $H - \Omega$, then $N_H(D)$ is a minimal separator of H containing x – a contradiction. Therefore $N_H[x] \subseteq \Omega$. If there is $y \in \Omega \setminus N_H[x]$, then by the same proposition, x and y must see a same component of $H - \Omega$, contradicting the fact that $N_H(x) \subseteq \Omega$. We deduce that $\Omega = N_H[x]$.

Since this holds for each vertex of Ω_i, we have that Ω_i is a clique in H (thus in M_i), and the vertices of Ω_i cannot have neighbors in $V_i \setminus \Omega_i$. Therefore Ω_i is a clique and a connected component of M_i. By Proposition 11, it is a potential maximal clique of M_i. The fact that $\Omega = N_H[x]$ also implies that $\Omega \setminus V_i = N_H(V_i)$, which concludes our proof. □

From Lemmata 6 and 7, we directly deduce:

Lemma 8. *Let Ω be a potential maximal clique of H. One of the following holds*

1. *Ω is the expansion of a potential maximal clique Ω_G of G.*
2. *There is some $i \in \{1, \ldots, k\}$ such that $\Omega \cap V_i$ is a potential maximal clique of M_i and $\Omega \setminus V_i = N_H(V_i)$.*

The previous lemma provides an injective mapping from the set of potential maximal cliques of H to the union of the sets of potential maximal cliques of G and of the graphs M_i. Therefore we have:

Corollary 4. *The number of potential maximal cliques of H is at most the number of potential maximal cliques of G plus the number of potential maximal cliques of each M_i.*

We use the bounds the number of minimal separators and potential maximal cliques of arbitrary graphs with respect to n, given in Proposition 20, to prove the main result of this section.

Theorem 4. *For any graph $G = (V, E)$, the number of its minimal separators is $\mathcal{O}(n \cdot \rho^{\mathrm{mw}(G)})$ where $\rho < 1.6181$ is the golden ratio. The number of its potential maximal cliques is $\mathcal{O}(n \cdot 1.7347^{\mathrm{mw}(G)})$. Moreover, the minimal separators and the potential maximal cliques can be enumerated in time $\mathcal{O}^*(1.6181^{\mathrm{mw}(G)})$ and $\mathcal{O}^*(1.7347^{\mathrm{mw}(G)})$ time respectively.*

Proof. Let $k = \mathrm{mw}(G)$. By definition of modular width, there is a decomposition tree of graph G, each node corresponding to a leaf, a disjoint union, a join or a decomposition into at most k modules. The leaves of the decomposition tree are disjoint graphs with at single vertex, thus these vertices form a partition of V. In particular, there are at most n leaves and, since each internal node is of degree at least two, there are $O(n)$ nodes in the decomposition tree. For each node N, let $G(N)$ be the graph associated to the subtree rooted in N. We prove that $G(N)$ has $\mathcal{O}(n(N) \cdot \rho^k)$ minimal separators and $\mathcal{O}(n(N) \cdot 1.7347^k)$ potential maximal clique, where $n(N)$ is the number of nodes of the subtree rooted in N. We proceed by induction from bottom to top. The statement is clear when N is a leaf.

Let N be an internal node $N_1, N_2, \ldots, N_p$ be its sons in the tree.

Graph $G(N)$ is the expansion of some graph $G'(N)$ by replacing the i-th vertex with module $G(N_i)$. If N is a *join* node, then $G'(N)$ is a clique. When N is a *disjoint union* node, graph $G'(N)$ is an independent set, and in the last case $G'(N)$ is a graph of at most k vertices. In all cases, by Proposition 20 graph $G'(N)$ has $\mathcal{O}(\rho^k)$ minimal separators. Thus $G(N)$ has at most $\mathcal{O}(\rho^k)$ more minimal separators than all its sons taken together, which completes our proof for minimal separators.

Concerning potential maximal cliques, when $G'(N)$ is a clique it has exactly one potential maximal clique, and when $G'(N)$ is of size at most k is has $\mathcal{O}(1.7347^k)$ potential maximal cliques. We must be more careful in the case when $G'(N)$ is an independent set (i.e., N is a disjoint union node), since in this case it has p potential maximal cliques, one for each vertex, and p can be as large as n. Consider a potential maximal clique Ω of $G(N)$ corresponding to an expansion of vertices of $G'(N)$ (see Lemma 8). It follows that this potential maximal clique is exactly the vertex set of some $G(N_i)$, for a child N_i of N. By construction this vertex set is disconnected from the rest of $G(N)$, and by Proposition 10 the only possibility is that this vertex set induces a clique in $G(N)$. But in this case Ω is also a potential maximal clique of $G(N_i)$. This proves that, when N is of type disjoint union, $G(N)$ has no more potential maximal cliques than the sum of the numbers of potential maximal cliques of all its sons. We conclude that the whole graph G has $\mathcal{O}(n \cdot 1.7347^k)$ potential maximal cliques. All our arguments are constructive and can be turned directly into enumeration algorithms for these objects. □

4.4 Applications

Recall that from Proposition 19 the following problems are solvable in $\mathcal{O}^*(|\Pi_G|)$ time, when Π_G is given in the input: (Weighted) Treewidth, (Weighted) Minimum Fill-In, Treelength. Moreover, from Proposition 22, for a fixed integer t and a fixed CMSO_2 formula φ, Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ is solvable in $\mathcal{O}(|\Pi_G| \cdot n^{t+4})$ time. Pipelined with Theorems 3 and 4, we deduce:

Theorem 5. *For an input graph G, the problems* Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ (Weighted) Treewidth, (Weighted) Minimum Fill-In, Treelength *are solvable in time $\mathcal{O}^*(4^{\mathrm{vc}(G)})$ and in time $\mathcal{O}^*(1.7347^{\mathrm{mw}(G)})$.*

We re-emphasis that problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ generalizes many classical problems, e.g., Maximum Independent Set, Maximum Induced Forest, Longest Induced Path, Maximum Induced Matching, Independent Cycle Packing, k-in-a-Path, k-in-a-Tree, Maximum Induced Subgraph With a Forbidden Planar Minor. More examples of particular cases are given in [67].

4.5 Treedepth

In [52], Deogun *et al.* give the following formula for computing the treedepth of a graph.

Proposition 29 ([52])**.** *Let G be a graph and Δ_G^+ a set of vertex subsets, containing all minimal separators of G.*

$$\mathrm{td}(G) = \min_{S \in \Delta_G^+} \Big(|S| + \max_C \mathrm{td}(G[C]) \Big)$$

where the maximum is taken over all components C of $G - S$.

Note that if G is a complete graph, then its treedepth is the number of vertices of G.

Following [52], we recursively define a *piece* D of $G = (V, E)$ as a vertex subset such that $D = V$, or there exists a larger piece C of G and a minimal separator S of $G[C]$ such that D is a connected component of $G[C] - S$.

Let us say that (C, S) is a *piece-separator pair* of G if C is a piece and S is a minimal separator of $G[C]$. Proposition 29 leads in [52] to a natural dynamic programming algorithm for computing the treedepth. Let us restate it for our convenience.

Lemma 9. *Given as input a graph G together with the set PS of all its piece-separator pairs (or a superset of this set), the treedepth of G can be computing in $\mathcal{O}^*(|PS|)$ time.*

Proof. Let *Pieces* be the set of all sets C such that $(C, S) \in PS$ (hence a superset of all pieces). Assume that the elements of *Pieces* are ordered by inclusion, or simply by size, using the bucket sort. The goal is to compute, in this order, a quantity that corresponds to the treedepth of $G[C]$, if C is really a piece. For the minimal sets $C \in$ *Pieces*, we set $\mathrm{td}[C] \leftarrow |C|$. Then for each $C \in$ *Pieces*, sorted by inclusion, for each S such that $(C, S) \in PS$, we check that all components D of $G[C] - S$ belong to *Pieces*. If this is not the case, then C is not a piece. Otherwise, we set compute $|S| + \sum_D \mathrm{td}[D]$ (the sum is taken over all components D), and we set $\mathrm{td}[C]$ to the minimum among these values, over all such S. (If no such S exist, we can set $\mathrm{td}[C] \leftarrow \infty$). By Proposition 29, at the end of the algorithm, $\mathrm{td}(G) = \mathrm{td}[V]$. □

Following the same ideas as in Theorem 1, we can list the piece-separator pairs of G in a running time which is single-exponential in its vertex cover.

Lemma 10. *Let W be a vertex cover of graph $G = (V, E)$.*

For each piece C of G, either C is a singleton or there is a three-partition (C^W, T^W, R^W) of W such that $C \cap W = C^W$ and $C \setminus W$ is formed by the vertices $x \in V \setminus W$ such that $N(x)$ is contained in $C^W \cup T^W$ and intersects C^W.

Proof. The proof goes by induction, from larger to smaller pieces. The base case corresponds to piece $C = V$. The first condition holds, for the partition $(W, \emptyset, \emptyset)$.

Let now C be a piece obtained as a component of $G[F] - Q$, for some larger piece F and some minimal separator Q of $G[F]$. We may assume w.l.o.g. that C is a full component associated to Q in $G[F]$. Indeed, if this is not the case, then $Q' = N_{G[F]}(C)$ is also a minimal separator of $G[F]$ (C is one of the full components associated to Q' in $G[F]$; another one can be obtained from a full component D associated to Q, by taking the component of $G[F] - Q'$ containing D, see Proposition 6). Thus we can replace Q by Q'.

By induction hypothesis, there is a partition (F^W, T_F^W, R_F^W) such that $F \cap W = F^W$ and for any vertex $y \in F \setminus W$, y is in F if and only if $N_G(y)$ intersects F^W and is contained in $F^W \cup T_F^W$. Consider the partition (C, S, D_2) of F and apply Lemma 1 to $G[F]$, this partition and the vertex cover $W' = W \cap F$ of $G[F]$. Let $C^W = C \cap W'$, $S^W = S \cap W'$ and $D_2^W = D_2 \cap W'$ (we obtain the same intersections if we replace W' by W). Denote $T^W = T_F^W \cup S^W$, and $R^W = R_F^W \cup D_2^W$. Observe that (C^W, T^W, R^W) is a three-partition of W. It remains to prove that, if C has more than one vertex, then this three-partition of W satisfies the condition of the lemma.

Let x be a vertex of $C \setminus W$, in particular $x \in F$. Since C is not a singleton, x has at least one neighbor in $C \cap W = C^W$, in graph G. Since $x \in F$, its neighborhood $N_G(x)$ is contained in $F \cup T_F^W$ by induction hypothesis. Now if $N_G(x)$ intersects D_2^W, we would have that $N_{G[F]}(x)$ intersects both C^W and D_2^W, contradicting (by Lemma 1) the fact that $x \in C$. It remains that $N_G(x)$ is contained in $C^W \cup T^W$.

Conversely, let $x \in V \setminus W$ such that $N_G(x)$ intersects C^W and is contained in $C^W \cup T^F$. Note that $x \in F$, by induction hypothesis. We cannot have $x \in S$ (its neighborhood would intersect D_2^W, hence R^W, by Lemma 1 applied on (C, S, D_2) in $G[F]$), nor $x \in D_2$ (because S separates C from D_2 in $G[F]$, so $N_G(D_2)$ cannot intersect C). It remains that $x \in C$, concluding the proof of the lemma. □

Theorem 6. *A graph G has at most $n + 3^{\mathrm{vc}(G)}$ pieces and at most $n + 5^{\mathrm{vc}(G)}$ piece-separator pairs. Moreover, the piece-separator pairs of G can be listed in $\mathcal{O}^*(5^{\mathrm{vc}(G)})$ time.*

Proof. The fact that G has at most $n+3^{\mathrm{vc}(G)}$ pieces is a straightforward consequence of Lemma 10: each piece C is either a single vertex, or is completely determined by some three-partition (C^W, T^W, R^W) of a minimum vertex cover W. We point out that the singleton condition of Lemma 10 is crucial. Indeed, if G is a star, its minimum vertex cover is of size one, and each of the other $n-1$ vertices is a piece formed by a single vertex.

Let us enumerate and upper bound the number of piece-separator pairs (C, S). There are at most n such pairs where C is a singleton, so assume that C has at least two vertices. For each such pair, let (C^W, T^W, R^W) be a partition of a minimum vertex cover W, like in Lemma 10. Since S is a minimal separator of $G[C]$, there is (by Lemma 1) a partition (D_1^W, S^W, D_2^W) of C^W such that $S \setminus W$ corresponds to vertices of $x \in C \setminus W$ whose neighborhood in $G[C]$ intersects both D_1^W and D_2^W. By Lemmata 10 and 1, the pair (C, S) is completely determined by the five-partition $(D_1^W, S^W, D_2^W, T^W, R^W)$. Hence there are at most $5^{\mathrm{vc}(G)}$ such piece-separator pairs, which proves the upper bound. The enumeration algorithm simply enumerates all pairs of type $(C, \emptyset)$ where $|C| = 1$, and all other pairs (C, S), from five-partitions of the vertex cover W, therefore enumerating a superset of the piece-separator pairs of G within the required running time. □

A similar result can be obtained using parameter modular width. As in Section 4.3, let $G = (V, E)$ be a graph with vertex set $V = \{v_1, \ldots, v_k\}$ and let $M_i = (V_i, E_i)$ be a family of pairwise disjoint graphs, for all i, $1 \leq i \leq k$. We denote by H the graph obtained from G by expanding each vertex v_i by the module M_i.

Lemma 11. *Let S be a minimal separator of H and C be a connected component of $H - S$. Then one of the following holds:*

1. *C is the expansion of a vertex subset C_G of G.*
2. *There is $i \in \{1, \ldots, k\}$ and a minimal separator S_i of $M_i = H[V_i]$ such that C is a connected component of $M_i - S_i$.*

In particular, each piece of H is an expansion of a vertex subset of G, or a piece of some module M_i.

Proof. The two conditions are a straightforward consequence of Lemma 5. Recall that, by this Lemma, S is the expansion of a minimal separator S_G of S (in which case the first condition holds), or there is some i and a minimal separator S_i of M_i such that $S = S_i \cup N_H(V_i)$. In the latter case, the component C is either contained in V_i (and the second condition holds), or does not intersect V_i, so it is the expansion of some component of $G - N_G(v_i)$ (implying the first condition of the lemma).

For the last part, we use the recursive definition of pieces. Consider a piece of $H[C]$, for some component C of $H - S$. If C is a component of $M_i - S_i$ for some i, then pieces of $H[C]$ are also pieces of $M_i[C]$ and hence of M_i. If C is an expansion of a vertex subset C_G of G, then $H[C]$ is an expansion of $H' = G[C_G]$ and we recursively apply the same argument on H'. □

Theorem 7. *For any graph G, the number of its pieces is $\mathcal{O}^*(2^{\mathrm{mw}(G)})$, and a superset of its piece-separator pairs can be listed in time $\mathcal{O}^*(3.2361^{\mathrm{mw}(G)})$.*

Proof. Let us count the pieces of G. Consider the modular-decomposition tree of G, like in the proof of Theorem 4. For each node N of the decomposition (corresponding to a leaf, a disjoint union, a join or a decomposition into at most $\mathrm{mw}(G)$ modules), let $G(N)$ be the graph whose corresponding decomposition tree is the subtree rooted in N, and $G'(N)$ be the prime, complete or independent graph corresponding to node N. By Lemma 11, the pieces of $G(N)$ are either pieces of $G(N_i)$ for some son N_i of N, or correspond to an expansion of $G'(N)$. If $G'(N)$ is prime, there are at most $2^{\mathrm{mw}(G)}$ such expansions. Also observe that if $G'(N)$ is independent then the only pieces of $G(N)$ that are expansions of $G'(N)$ correspond to single vertices of $G'(N)$. Hence, there are at most n such pieces. In the case when $G'(N)$ is a complete graph, the only possible piece of $G(N)$ that is an expansion of $G'(N)$ is the whole graph $G(N)$, hence this piece is unique. Altogether, G has $\mathcal{O}^*(2^{\mathrm{mw}(G)})$ pieces.

For each piece C of G, there are $\mathcal{O}^*(\rho^{\mathrm{mw}(G)})$ minimal separators in $G[C]$, where ρ is the golden ratio, by Theorem 4 and the fact that the modular width of $G[C]$ is at most $\mathrm{mw}(G)$. Hence the number of piece-separator pairs of G is $\mathcal{O}^*((2 \cdot \rho)^{\mathrm{mw}(G)})$, thus $\mathcal{O}^*(3.2361^{\mathrm{mw}(G)})$. The counting arguments can be transformed into an enumeration algorithm for a superset of all possible such pairs, within the same running time bounds. □

From Lemma 9 and Theorems 6 and 7 we deduce:

Theorem 8. *The treedepth of a graph G is computable in time $\mathcal{O}^*(5^{\mathrm{vc}(G)})$ and in time $\mathcal{O}^*(3.2361^{\mathrm{mw}(G)})$.*

4.6 Discussion

We have proved single exponential upper bounds for the number of minimal separators and the number of potential maximal cliques of graphs, with respect to parameters vertex cover and modular width.

We point out that our bounds on the number of potential maximal cliques w.r.t. vertex cover and to modular width do not seem to be tight. Any improvement on these bounds and of the enumeration algorithm of potential maximal cliques would imply faster algorithms for the problems mentioned in Section 4.4.

A natural question is whether these results can be extended to other natural graph parameters. We observe that for parameters like feedback vertex set, clique-width or maximum leaf spanning tree, one cannot obtain upper bounds of type $\mathcal{O}^*(f(k))$ for any function f. A counterexample is provided by the graph $W_{p,q}$, formed by p disjoint paths of q vertices plus two vertices u and v seeing the left, respectively right ends of the paths (similar to the watermelon graph of Figure 4.1). Indeed this graph has feedback vertex set 1, a maximum leaf spanning tree with p leaves and a clique-width of no more than $2p+1$, but it has roughly $p^{n/p}$ minimal u,v-separators. Skodinis [131] observes that, if we chose as parameter d the maximum degree of the complement graph, then the number of minimal separators is bounded by $\mathcal{O}^*(2^{\mathcal{O}(d)})$. The argument is that each a,b-minimal separator contains all common neighbors of a and b, hence all but $2d$ vertices of the graph. A similar bound holds for the number of potential maximal cliques.

A different extension of this technique will be considered in the next chapter, with different parameters including feedback vertex set. We show that we can solve the problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ in FPT time. The results of this chapter and the next one are complementary, in the sense that in the results of this chapter we considered a more restricted family of graphs and obtained single exponential algorithms. The results of next chapter consider a larger family of graphs, but the algortihms are not single exponential anymore, and the results does not extend to problems like Treewidth of Treedepth.

Chapter 5

Beyond classes of graphs with *few* minimal separators: FPT results through potential maximal cliques

Consider the class $\mathcal{G}_{\text{poly}} + kv$, formed by graphs of $\mathcal{G}_{\text{poly}}$ to which we may add a set of at most k vertices with arbitrary adjacencies, called *modulator*. In this chapter, we prove that problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ is fixed parameter tractable on $\mathcal{G}_{\text{poly}} + kv$, with parameter k, if the modulator is also part of the input. The running time is of type $\mathcal{O}\left(f(k+t,\mathcal{P}) \cdot n^{t+5} \cdot (\text{poly}(n)^2)\right)$, for some function f.

Most of the results of this chapter were published in the 41st International Workshop of Graph-Theoretic Concepts in Computer Science (WG 2015) and were obtained in collaboration with Mathieu Liedloff and Ioan Todinca [102]. The hardness result was obtained also in collaboration with Iyad Kanj.

5.1 Introduction

As we remarked in the last chapter, the combinatorial bounds on the number of minimal separators and potential maximal cliques with respect to vertex cover and modular width, can not be extended to other natural graph parameters like feedback vertex set, clique-width or maximum leaf spanning tree. The counterexample is the watermelon graph, where those parameters are bounded (constant), while the number of minimal separators is exponential. However, there is another interesting property of the watermelon graph: the bounded feedback vertex set (equals 1) implies that the deletion of one vertex leads to a graph with polynomially many minimal separators ($\mathcal{O}(n)$ since it is a tree). In other words, the watermelon graph is *one vertex far from having polynomially many minimal separators*.

In this chapter our goal is to study the algorithmic applications of potential maximal cliques in classes of graphs that are *close* to have polynomially many minimal separators. From a parameterized perspective, these are classes of graphs with "few" minimal separators, to which we are allowed to add k vertices with arbitrary adjacencies. Let poly be some polynomial, and let $\mathcal{G}_{\text{poly}} + kv$ denote the class of graphs $G = (V, E)$ containing a vertex subset $M \subseteq V$ of size at most k, such that $G - M \in \mathcal{G}_{\text{poly}}$. The set M is called the *modulator* of G.

Let Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ on $\mathcal{G}_{\text{poly}}+kv$ be the problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ on the graph class $\mathcal{G}_{\text{poly}}+kv$, with parameter k. *Moreover, we assume that the input graph G is given together with a modulator M* (see also Section 5.6 for discussions on this point). Our main result is that problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ on $\mathcal{G}_{\text{poly}}+kv$ is fixed-parameter tractable (FPT), i.e., there is an algorithm solving the problem in time $f(k) \cdot n^{O(1)}$ for some function f. More specifically, the running time is of type $\mathcal{O}\left(f(k+t,\mathcal{P}) \cdot n^{t+5} \cdot (\text{poly}(n)^2)\right)$, where function f depends on property $\mathcal{P}$ and on $k+t$ (see also the Conclusion section for further discussions).

Theorem 9. *Problem* Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ on $\mathcal{G}_{\text{poly}}+kv$ *with parameter k is fixed-parameter tractable, when a modulator is also part of the input.*

A natural idea is to "guess" how the optimal solution intersects with the modulator M. But we still need to carefully express how the rest of the solution intersects with the graph $G - M$. Think, e.g., of the LONGEST INDUCED PATH problem: we need to make sure that the solution restricted to $G - M$ forms indeed a connected path with the selected vertices of the modulator. Our algorithm extends, in a non-trivial way, the one of [67] in order to handle this situation.

We also point out that several authors considered classes of graphs of similar flavor, e.g., *Chordal* + kv [110] and *Split*+kv [109], providing both FPT and hardness results for various problems, parameterized by k.

5.2 The algorithm

Our goal is to provide an FPT algorithm for the problem MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ ON $\mathcal{G}_{\text{poly}} + kv$, thus proving our Main Theorem 9. Recall that in this problem, the input is a graph $G \in \mathcal{G}_{\text{poly}} + kv$, together with a modulator M of size at most k.

We may assume w.l.o.g. that we also have as input the set $\Pi_{G'}$ of potential maximal cliques of graph $G' = G - M$. Indeed these objects can be computed in polynomial time by [33] (Proposition 15).

The following easy observation is crucial for the correctness of our algorithm.

Lemma 12 (compatibility lemma)**.** *Let G be the input graph, M be a vertex subset, and (F, X) be an optimal solution for $\mathcal{P}$ and t. Let $G' = G - M$ and $F' = F \setminus M$. There is a minimal triangulation $T_{F'}$ of $G'[F']$ of width at most t, and a minimal triangulation $T_{G'}$ of G' respecting $T_{F'}$, i.e., such that $T_{F'}$ is the subgraph induced by F' in $T_{G'}$.*

Proof. Since $G[F]$ is of treewidth at most t, so is its subgraph $G'[F']$. Therefore it exists a minimal triangulation $T_{F'}$ of $G'[F']$ of width at most t. By Proposition 23 applied to graph G' and to $T_{F'}$, there is a minimal triangulation $T_{G'}$ of G', respecting the minimal triangulation $T_{F'}$. □

We will "guess", by brute force, the intersections of F and X with the modulator M. Let us fix $F^M = F \cap M$ and $X^M = X \cap M$, with $X^M \subseteq F^M$. For each such pair (F^M, X^M), we need to construct $F' = F \setminus M$ and $X' = X \setminus M$ such that the pair $(F, X) = (F' \cup F^M, X' \cup X^M)$ satisfies the constraints and $X' \cup X^M$ is of maximum size under these conditions. (Eventually, the global solution is obtained by trying all the 3^k possible combinations for subsets $X^M \subseteq F^M \subseteq M$.)

Our algorithm will construct X' and F' by dynamic programming on minimal separators and potential maximal cliques. The graph $G[F' \cup F^M]$ has to be of treewidth at most t, and while we construct this graph we also need to check property $\mathcal{P}$ on it. Unfortunately we will not be able to handle $G[F' \cup F^M]$ as a $(t+1)$-terminal recursive graph. Instead, we will see it as a $(t+k^M+1)$-terminal recursive graph, where $k^M = |F^M|$ (and we will explicitly check that $\text{tw}(G[F]) \leq t$). Informally, while we construct F', we also maintain some information on a tree-decomposition of $G'[F']$, of width at most t. To this decomposition, we simply add the whole set F^M in each bag, hence obtaining a tree-decomposition of width at most $t + k^M$ of $G[F \cup F^M]$.

Now, on the $(t + k^M + 1)$-terminal recursive graph $G[F' \cup F^M]$ having $W \cup F^M$ as set of terminals for some W (W will be memorized during the dynamic programming), we need to check the property $\mathcal{P}$ but also the fact that the graph is of treewidth at most t. Therefore, let $\mathcal{Q}(H, Y)$ be the property $\mathcal{P}(H, Y) \wedge (\text{tw}(H) \leq t)$. Property $\mathcal{Q}$ is also regular (see Section 5.3). Of course our approach, keeping the class $h_{\mathcal{Q}}(G[F], X)$ for property $\mathcal{Q}$, ensures that if two partial solutions are of the same class, they are equivalent w.r.t. extensions.

Our algorithm is an extension of the dynamic programming scheme proposed by Fomin *et al.* [67] (Proposition 22, see Section 3.3.2). For a better understanding we completely describe the new algorithm, trying to follow the same notations as in [67] and Section 3.3.2, and we emphasize the points that differ from the result of Fomin *et al.*

Recall that sets X^M and F^M are fixed, and $X^M \subseteq F^M \subseteq M$. We consider a total order $(v_1, \ldots, v_n)$ on the vertices of $G = (V, E)$. When we speak of a subset T of vertices as a set of terminals, T is considered as the ordered set, with the ordered induced by $(v_1, \ldots, v_n)$ on its vertices.

Definition 4 (partial compatible solution)**.** *Consider a block (S, C) and a good triple (S, C, Ω) of graph G'. Let $W \subseteq S$ (resp. $W \subseteq \Omega$) a vertex set of size at most $t + 1$. Let c be a homomorphism class for property $\mathcal{Q}$ on $(t + k^M + 1)$-terminal recursive graphs. We say that a pair (F, X) is a* partial solution compatible with (S, C, W, c) *(resp. with (S, C, Ω, W, c)) if the following conditions hold :*

1. $X \cap M = X^M$, $F \cap M = F^M$, *and* $X \subseteq F$.
2. $F \setminus M \subseteq S \cup C$ *and* $F \cap S = W$ *(resp.* $F \cap \Omega = W$*).*
3. $H = (F, W \cup F^M, E(G[F]))$ *is a* $(t + k^M + 1)$*-terminal recursive graph, and the homomorphism class* $h_Q(H, X)$ *for property* Q *is exactly* c*.*
4. *There is a minimal triangulation* $T_{F'}$ *of* $G'[F']$ *(here* $F' = F \setminus M$*) and a minimal triangulation* $T_{G'}$ *of* G' *respecting* $T_{F'}$*, such that* S *is a minimal separator (resp.* Ω *is a maximal clique) of* $T_{G'}$*.*

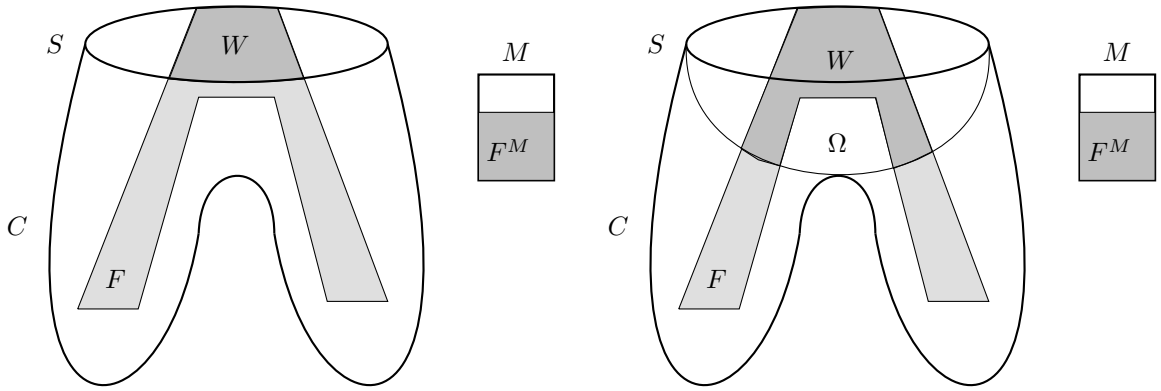


Figure 5.1 – (a) Partial solutions compatible with (S, C, W, c) (left), and (b) with (S, C, Ω, W, c) (right). Set F is depicted in grey. Note that set W corresponds to $F \cap S$ in the first case, and to $F \cap \Omega$ in the second case.

With the same notations as in Section 3.3.2, let $\alpha(S, C, W, c)$ (resp. $\beta(S, C, \Omega, W, c)$) be the maximum size of X over all partial solutions (F, X) compatible with (S, C, W, c) (resp. (S, C, Ω, W, c)). The situation is depicted in Figure 5.1. For simplicity, we did not represent set X. The algorithm orders the blocks (S, C) by the size of $S \cup C$. It proceeds by dynamic programming on blocks (S, C) in this order, and on good triples (S, C, Ω), computing all possible values $\alpha(S, C, W, c)$ and $\beta(S, C, \Omega, W, c)$. The outline of Algorithm 2 is the same as in Section 3.3.2 (and [67]), the differences appear in the details of the computations of α and β values.

Base case: the good triple (S, C, Ω) is such that $\Omega = S \cup C$. In this case the only possible partial solutions (F, X) compatible with (S, C, Ω, W, c) correspond to base graphs $G[F]$, where all vertices are terminals (see [67]). Hence $F = W \cup F^M$ is also the set of terminals. Thus set X is unique (or might not

Algorithm 2: MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ on $\mathcal{G}_{\text{poly}} + kv$

Input: graph $G = (V, E)$ and a modulator M of size at most k s.t. $G' = G - M$ is in $\mathcal{G}_{\text{poly}}$; the potential maximal cliques of G'; sets $X^M \subseteq F^M \subseteq M$

Output: sets $X \subseteq F \subseteq V(G)$ such that $G[F]$ has treewidth at most t, $\mathcal{P}(G[F], X)$ is true, $X \cap M = X^M$, $F \cap M = F^M$ and, subject to these constrains, X is of maximum size

```
1 Order all blocks (S, C) of G' by inclusion on S ∪ C;
2 for all blocks (S, C) in this order do
3     for all good triples (S, C, Ω) of G', all W ⊆ Ω of size ≤ t + 1 and all c ∈ C do
4         if Ω = S ∪ C then Compute β(S, C, Ω, W, c) using Equation 5.1 ;
5         else Compute β(S, C, Ω, W, c) using Equations 5.3, 5.4, 5.5, and 5.6 ;
6     for all W ⊆ S of size ≤ t + 1 and all c ∈ C do
7         Compute α(S, C, W, c) using Equation 5.2;
8 Compute an optimal solution using Equation 5.7;
```

exist), because we must have $X = trm(c, W \cup F^M)$. For simplicity, we denote by $G[W \cup F^M]$ the base graph $(W \cup F^M, W \cup F^M, E(G[F \cup F^M]))$. We have:

$$\beta(S, C, \Omega, W, c) = \begin{cases} |X| & \text{if there is } X \subseteq W \text{ such that } h(G[W \cup F^M], X) = c \\ -\infty & \text{otherwise} \end{cases} \tag{5.1}$$

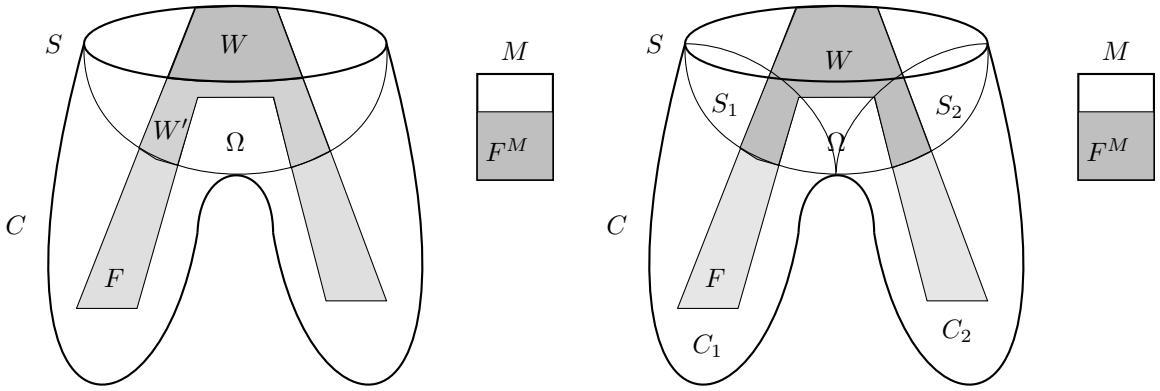


Figure 5.2 – Computing α form β (left), and β from α (right). Set M is not depicted for simplicity.

Computing α from β. We aim to compute $\alpha(S, C, W, c)$ from β values on good triples of type (S, C, Ω) (see also Figure 5.2(a)).

Let (F, X) be an optimal solution compatible with (S, C, W, c) and $F' = F \setminus M$. We denote by H the $(t + k^M + 1)$-terminal recursive graph $G[F]$ with $W \cup F^M$ as set of terminals. By Definition 4, there is a minimal triangulation $T_{F'}$ of F' and a minimal triangulation $T_{G'}$ of G' respecting $T_{F'}$, such that S is a minimal separator of $T_{G'}$. By [32], there is a potential maximal clique Ω of G', inducing a maximal clique in $T_{G'}$, and such that (S, C, Ω) form a good triple of G'. Let $W' = F' \cap \Omega$. The graph H', corresponding to $G[F]$ with set of terminals $W' \cup F^M$, is also a $(t + k^M + 1)$-terminal recursive graph (see Proposition 3, or [67] for full details). Let $c' = h(H', X)$. Note that $H = forget_{W' \cup F^M \to W \cup F^M}(H)$, where the $forget$ operation corresponds to the fact that the set of terminals $W' \cup F^M$ is reduced to $W \cup F^M$. Therefore, we have (see [67] for full details):

$$\alpha(S, C, W, c) = \max \beta(S, C, \Omega, W', c'), \tag{5.2}$$

where the maximum is taken over potential maximal cliques Ω such that (S, C, Ω) is a good triple, all subsets $W' \subseteq \Omega$ of size at most $t + 1$ such that $W' \cap S = W$ and all classes $c' \in \mathcal{C}$ such that $\odot_{forget_{W' \cup F^M \to W \cup F^M}}(c') = c$.

Computing β from α. Let (S, C, Ω) be a good triple of G. Denote by $C_1, \ldots, C_p$ the components of $G' - \Omega$ contained in C, and let S_i be the neighborhood of C_i in G'. The pairs (S_i, C_i) are also blocks (see [32] for more details) and they have been processed by the algorithm before (S, C). Our goal is to compute $\beta(S, C, \Omega, W, c)$. Let $W_i = W \cap S_i$, for all $i \in \{1, \ldots p\}$. We will use, as in [67], two intermediate functions γ_i and δ_i.

Let $\delta_i(S, C, \Omega, W, c_i^+)$ denote $\max |X_i|$ over the partial solutions (F_i^+, X_i) that are compatible with (S, C, Ω, W, c_i^+) and such that $F_i^+ \setminus F^M \subseteq \Omega \cap C_i$. Let H_i^+ be the graph $G[F_i^+]$ with set of terminals $W \cup F^M$. Let also H_i be the graph $H_i^+[S_i \cup C_i \cup M]$, with set of terminals $W_i \cup F^M$. Note that H_i^+ is obtained by gluing H_i with the base graph $G[W \cup F^M]$, with the "canonical" gluing, obtained by identifying the vertices of S_i from both sides. Let $glue_{W_i \cup F^M; W \cup F^M}$ denote this gluing operation.

$$\delta_i(S, C, \Omega, W, c_i^+) = \max \alpha(S_i, C_i, W_i, c_i) + |trm(c_W, W \cup F^M) \setminus trm(c_i, W_i \cup F^M)|, \tag{5.3}$$

where the maximum is taken over all $c_i, c_W \in \mathcal{C}$ s.t. $\odot_{glue_{W_i \cup F^M; W \cup F^M}}(c_i, c_W) = c_i^+$ and $c_W = h(G[W \cup F^M], X_W \cup X^M)$ for some $X_W \subseteq W$. Here $G[W \cup F^M]$ denotes the base graph with terminals $W \cup F^M$.

Notice the part $|trm(c_W, W \cup F^M) \setminus trm(c_i, W_i \cup F^M)|$ in the formula, which avoids the overcounting of the vertices of $X_i \cap S_i$.

These partial solutions (F_i^+, X_i^+) corresponding to $\delta_i(S, C, \Omega, W, c_i^+)$ cannot be glued together in one step, since we are only allowed to glue two graphs at a time. Hence the need of the γ function which allows to add, one by one, the partial solutions to the gluing. Now let $\gamma_i(S, C, \Omega, W, c)$ denote the size of the optimal partial solution compatible with (S, C, Ω, W, c) and contained in $M \cup \Omega \cup C_1 \cdots \cup C_i$. So we only consider the first i components, the partial solution is the union of (F_1^+, X_1^+) to (F_i^+, X_i^+). By definition,

$$\gamma_1(S, C, \Omega, W, c) = \delta_1(S, \Omega, C, W, c) \tag{5.4}$$

We then compute γ_i, for i from 2 to p as follows.

$$\gamma_i(S, C, \Omega, W, c) = \max \gamma_{i-1}(S, C, \Omega, W, c') + \delta_i(S, \Omega, C, W, c'') - |trm(c', W \cup F^M)|, \tag{5.5}$$

where the maximum is taken over all $c', c'' \in \mathcal{C}$ s. t. $\odot_{glue_{W \cup F^M; W \cup F^M}}(c', c'') = c$, where the gluing operation is the canonical gluing, the set of terminals for both arguments being $W \cup F^M$.

By definition of γ_p, we have

$$\beta(S, C, \Omega, W, c) = \gamma_p(S, C, \Omega, W, c). \tag{5.6}$$

It remains to retrieve the optimal solution for the algorithm. The maximum is taken over all accepting classes c, i.e., classes such that $h(G, X) = c$ implies that $\mathcal{P}(G, X)$:

$$\max \alpha(\emptyset, V, \emptyset, c), \tag{5.7}$$

We refer to Section 3.3.2 and [67] for detailed proofs of correctness and for complexity issues. Altogether, the algorithm takes time $\mathcal{O}(f(t + k^M, \mathcal{P}) \cdot n^{t+4} \cdot |\Pi_{G'}|)$. The function $f(t + k^M, \mathcal{P})$ comes from the application of Proposition 4 on $t + k^M + 1$-recursive graphs. By applying Algorithm 2 on all possible subsets $X^M \subseteq F^M \subseteq F$, we have proved Theorem 9.

We point out that our algorithm really needs to keep track of the homomorphism classes of partial solutions (F, X) for property $\mathcal{Q}(G[F], X) = \mathcal{P}(G[F], X) \wedge (\text{tw}(G[F]) \leq t)$. A naive approach would be to only keep the class $h_{\mathcal{P}}(G[F], X)$ (for property $\mathcal{P}$) and to reject partial solutions that do not satisfy $\text{tw}(G[F]) \leq t$. We could have two different partial solutions (F, X) and (F', X) for the same part of the graph (e.g., corresponding to the same parameters for function α), such that $h_{\mathcal{P}}(G[F], X) = h_{\mathcal{P}}(G[F'], X')$, and both $\text{tw}(G[F])$ and $\text{tw}(G[F'])$ are at most t. Or, it may happen that one of the solution, say (F, X), can be extended into a better one of type $(F \cup F'', X \cup X'')$, while the other cannot because such an extension$(F' \cup F'', X' \cup X'')$ would violate the condition $\text{tw}(G[F' \cup F'']) \leq t$.

Of course our approach, keeping the class $h_{\mathcal{Q}}(G[F], X)$ for property $\mathcal{Q}$, ensures that if two partial solutions are of the same class, they are equivalent w.r.t. extensions.

5.3 On the regularity of $\mathcal{Q}$

In this section, we explain why property $\mathcal{Q}$ is regular. Let us define this slightly more general lemma, that will be useful also in Chapter 6 . Let $\mathcal{Q}(G, X, Y)$ be a property assigning to each graph G and vetex subsets X and Y a Boolean value. We extend the gluing and forget operations to triplets (G, X, Y) in the natural way, as we made with pairs (G, X). In particular if $(G, X, Y) = glue_m((G_1, X_1, Y_1), (G_2, X_2, Y_2))$ then $(G, X) = glue_m((G_1, X_1), (G_2, X_2))$ and $(G, Y) = glue_m((G_1, Y_1), (G_2, Y_2))$, and if $(G, X, Y) = forget_m(G', X', Y')$ then $(G, X) = forget_m(G', X')$ and $(G, Y) = forget_m(G', Y')$.

Lemma 13. *Let $\mathcal{P}_1(G, X)$, $\mathcal{P}_2(G, Y)$ regular properties on graphs and vertex sets, and let $\mathcal{Q}(G, X, Y)$ be the property $\mathcal{P}_1(G, X) \wedge \mathcal{P}_2(G, Y)$. Then property $\mathcal{Q}$ is regular.*

Proof. Let $\mathcal{C}_{\mathcal{P}}$, $h_{\mathcal{P}}$, $\odot^{\mathcal{P}}_{glue_m}$ and $\odot^{\mathcal{P}}_{forget_m}$ be the set of classes, homomorphism function, gluing function and forget function for a property $\mathcal{P}$. We show that if two properties $\mathcal{P}_1(G, X)$ and $\mathcal{P}_2(G, Y)$ are regular, then property $\mathcal{Q}(G, X, Y) = \mathcal{P}_1(G, X) \wedge \mathcal{P}_2(G, Y)$ is also regular. In order to prove this, observe that

1) We may define the set of classes of $\mathcal{Q}$ as $\mathcal{C}_{\mathcal{Q}} = \mathcal{C}_{\mathcal{P}_1} \times \mathcal{C}_{\mathcal{P}_2}$.

2) The homomorphism function $h_{\mathcal{Q}}$ for property $\mathcal{Q}$ is defined as $h_{\mathcal{Q}}(G, X, Y) = (h_{\mathcal{P}_1}(G, X), h_{\mathcal{P}_2}(G, Y))$.

3) For $c_1, c_2 \in \mathcal{C}_{\mathcal{Q}}$, such that $c_1 = (c_1^{\mathcal{P}_1}, c_1^{\mathcal{P}_2})$ and $c_2 = (c_2^{\mathcal{P}_1}, c_2^{\mathcal{P}_2})$, then the gluing function of $\mathcal{Q}$ is

$$\odot^{\mathcal{Q}}_{glue_m}(c_1, c_2) = \left(\odot^{\mathcal{P}_1}_{glue_m}(c_1^{\mathcal{P}_1}, c_2^{\mathcal{P}_1}), \odot^{\mathcal{P}_2}_{glue_m}(c_1^{\mathcal{P}_2}, c_2^{\mathcal{P}_2})\right)$$

4) For $c = (c^{\mathcal{P}_1}, c^{\mathcal{P}_2}) \in \mathcal{C}_{\mathcal{Q}}$, the forget function of $\mathcal{Q}$ is

$$\odot^{\mathcal{Q}}_{forget_m}(c) = \left(\odot^{\mathcal{P}_1}_{forget_m}(c^{\mathcal{P}_1}), \odot^{\mathcal{P}_2}_{forget_m}(c^{\mathcal{P}_2})\right)$$

We must show that $\mathcal{C}_{\mathcal{Q}}$, $h_{\mathcal{Q}}$, $\odot^{\mathcal{Q}}_{glue_m}$ and $\odot^{\mathcal{Q}}_{forget_m}$ satisfy the properties asked in Definition 1.

Suppose first that $h_{\mathcal{Q}}(G_1, X_1, Y_1) = h_{\mathcal{Q}}(G_2, X_2, Y_2)$. From observation 1) we have that $h_{\mathcal{P}_1}(G_1, X_1) = h_{\mathcal{P}_1}(G_2, X_2)$ and $h_{\mathcal{P}_2}(G_1, Y_1) = h_{\mathcal{P}_2}(G_2, Y_2)$. Since $\mathcal{P}_1$ and $\mathcal{P}_2$ are regular, $\mathcal{P}_1(G_1, X_1) = \mathcal{P}_1(G_2, X_2)$ and $\mathcal{P}_2(G_1, Y_1) = \mathcal{P}_2(G_2, Y_2)$. Hence $h_{\mathcal{P}_1}(G_1, X_1) = h_{\mathcal{P}_1}(G_2, X_2)$ implies $\mathcal{Q}(G_1, X_1, Y_1) = \mathcal{Q}(G_2, X_2, Y_2)$.

Let $(G, X, Y) = glue_m((G_1, X_1, Y_1), (G_2, X_2, Y_2))$. Then

$$\begin{aligned}
h_{\mathcal{Q}}(G, X, Y) &= (h_{\mathcal{P}_1}(G, X), h_{\mathcal{P}_2}(G, Y)) \\
&= (h_{\mathcal{P}_1}(glue_m((G_1, X_1), (G_2, X_2))), h_{\mathcal{P}_2}(glue_m((G_1, Y_1), (G_2, Y_2)))) \\
&= (\odot^{\mathcal{P}_1}_{glue_m}(h_{\mathcal{P}_1}(G_1, X_1), h_{\mathcal{P}_1}(G_2, X_2)), \odot^{\mathcal{P}_2}_{glue_m}(h_{\mathcal{P}_2}(G_1, Y_1), h_{\mathcal{P}_2}(G_2, Y_2))) \\
&= \odot^{\mathcal{Q}}_{glue_m}((h_{\mathcal{P}_1}(G_1, X_1), h_{\mathcal{P}_2}(G_1, Y_1)), (h_{\mathcal{P}_1}(G_2, X_2), h_{\mathcal{P}_2}(G_2, Y_2))) \\
&= \odot^{\mathcal{Q}}_{glue_m}(h_{\mathcal{Q}}(G_1, X_1, Y_1), h_{\mathcal{Q}}(G_2, X_2, Y_2))
\end{aligned}$$

Finally, if $(G', X', Y') = forget_m(G, X, Y)$ then

$$\begin{aligned}
h_{\mathcal{Q}}(G', X', Y') &= (h_{\mathcal{P}_1}(G', X'), h_{\mathcal{P}_2}(G', Y')) \\
&= (h_{\mathcal{P}_1}(forget_m(G, X)), h_{\mathcal{P}_2}(forget_m(G, Y))) \\
&= (\odot^{\mathcal{P}_1}_{forget_m}(h_{\mathcal{P}_1}(G, X)), \odot^{\mathcal{P}_2}_{forget_m}(h_{\mathcal{P}_2}(G, Y))) \\
&= \odot^{\mathcal{Q}}_{forget_m}(h_{\mathcal{P}_1}(G, X), h_{\mathcal{P}_2}(G, Y)) \\
&= \odot^{\mathcal{Q}}_{forget_m}(h_{\mathcal{Q}}(G, X, Y))
\end{aligned}$$

Therefore $\mathcal{C}_{\mathcal{Q}}$, $h_{\mathcal{Q}}$, $\odot^{\mathcal{Q}}_{glue_m}$ and $\odot^{\mathcal{Q}}_{forget_m}$ satisfy the properties asked in the definition of a regular property (Definition 1), so $\mathcal{Q}$ is regular. □

Lemma 14. *Let $\mathcal{P}(G, X)$ be a regular property on graphs and vertex sets, and let $\mathcal{Q}(G, X)$ be the property $\mathcal{P}(G, X) \wedge (\mathrm{tw}(G) \leq t)$. Property $\mathcal{Q}$ is regular.*

Proof. Suppose that property $\mathcal{P}'(G)$ defined by "$\mathrm{tw}(G) \leq t$" is regular, then from Lemma 13, $\mathcal{Q}$ is regular. It remains to argue that property $\mathcal{P}'(G)$ is regular. One classical argument is that the class of graphs of treewidth at most t is minor-closed (see, e.g., [22] for a similar discussion). Hence, by the Graph Minors theorem [127], the class is defined by a finite set of forbidden minors, denoted $Obs(t)$. Therefore, $\mathrm{tw}(G) \leq t$ if and only if G has no minor among the graphs of $Obs(t)$. The property that a fixed graph is a minor of G is expressible in CMSO (see, e.g., [31]), hence the property "$\mathrm{tw}(G) \leq t$" is regular by Proposition 4. In order to turn this argument into a completely constructive one, we must build the obstruction set $Obs(t)$ for graphs of treewidth at most t. This can be done by brute force, thanks to the result of Lagergren [98] showing that such obstructions are of size (number of vertices) at most $f(t)$, for some function doubly exponential in t[5]. Hence, one could enumerate all the graphs with number of vertices bounded by this function, test the ones of treewidth strictly larger than t and extract the minimal ones w.r.t. the minor relation. The output is precisely $Obs(t)$.

For our purpose, a better alternative is provided by the celebrated algorithm of Bodlaender and Kloks [24]. This algorithm takes as input a graph of treewidth at most w, for some constant w, and decides if this graph has treewidth at most t. Its running time is $O(n)$, and the hidden constant is single exponential in a polynomial in w. Moreover, the algorithm precisely provides an effective way for constructing the homomorphism class of property $\mathcal{P}'(G) = (\mathrm{tw}(G) \leq t)$ for $(w+1)$-terminal recursive graphs. Such a class is addressed in [24] as a *full set of characteristics*. Recall that, in our case, we need to construct these class for $(w+1)$-terminal recursive graphs where $w \leq t + k$, k being the size of the modulator M. Therefore, for computing the homomorphism classes for property $\mathcal{Q}(G, X) = \mathcal{P}(G, X) \wedge (\mathrm{tw}(G) \leq t)$ one can combine the Borie, Parker, and Tovey approach (to obtain $h_{\mathcal{P}}(G, X)$) and the Bodlaender and Kloks approach (to obtain $h_{\mathcal{P}'}(G)$ for $\mathcal{P}'(G) = (\mathrm{tw}(G) \leq t)$), and the couple $(h_{\mathcal{P}}(G, X), h_{\mathcal{P}'}(G))$ is the homomorphism class $h_{\mathcal{Q}}(G, X)$. □

5.4 Vertex and edge deletion/addition

For a polynomial poly, we denote $\mathcal{G}_{\text{poly}} + ke$ (resp. $\mathcal{G}_{\text{poly}} - ke$) the class of graphs obtained from graphs of $\mathcal{G}_{\text{poly}}$ by adding (resp. removing) at most k edges, and by $\mathcal{G}_{\text{poly}} - kv$ the class obtained by removing at most k vertices. For each of these classes, the set of at most k edges/vertices that has to be added/removed to/from the graph of $\mathcal{G}_{\text{poly}}$ is called a *modulator* for the respective class.

The following observation allows us to relate these classes.

Proposition 30 (Lemma 3 in [33]). *Let $G' = (V', E')$ be an induced subgraph of $G = (V, E)$. For any minimal separator S' of G', there is a minimal separator S of G such that $S' = S \cap V'$. In particular, G' has no more minimal separators than G.*

We deduce that, for small values of k w.r.t. n, class $\mathcal{G}_{\text{poly}} - kv$ is a subclass of $\mathcal{G}_{\text{poly}'}$ for some polynomial poly$'$, and classes $\mathcal{G}_{\text{poly}} - ke$ and $\mathcal{G}_{\text{poly}} + ke$ are subclasses of $\mathcal{G}_{\text{poly}'} + 2kv$:

Lemma 15. *Let poly be a polynomial. There is a polynomial* poly$'$ *such that, for each n-vertex graph $G' \in \mathcal{G}_{\text{poly}}$:*

1. *for any graph $G \in \mathcal{G}_{\text{poly}} - kv$ obtained by removing at most $k \leq n/2$ vertices from G', we have $G \in \mathcal{G}_{\text{poly}'}$;*
2. *for any graph $G \in \mathcal{G}_{\text{poly}} - ke$ (resp. $G \in \mathcal{G}_{\text{poly}} + ke$) obtained by removing (resp. adding) at most $k \leq n/4$ edges from (resp. to) G', we have $G \in \mathcal{G}_{\text{poly}'} + 2kv$.*

   Moreover, if G is given together with a modulator for class $\mathcal{G}_{\text{poly}} - ke$ or $\mathcal{G}_{\text{poly}} + ke$, the modulator for class $\mathcal{G}_{\text{poly}'} + 2kv$ can be found in linear time.

Proof. Choose a polynomial poly$'$ such that, for any $n' \geq n/2$, poly$'(n') \geq$ poly(n).

Let $G \in \mathcal{G}_{\text{poly}} - kv$, obtained by removing a set M of at most $k \leq n/2$ vertices from G'. By Proposition 30, G has at most poly(n) minimal separators. It also has n' vertices, with $n' \geq n - k \geq n/2$ vertices. Since poly$'(n') \geq$ poly(n), graph G belongs to class $\mathcal{G}_{\text{poly}'}$. This proves the first part of the lemma.

For the second part, let F be the set of edges removed from (resp. added to) G' in order to obtain graph G. Let M denote the set of vertices of G' incident to at least one edge of F. Clearly, $|M| \leq 2k$. By the same arguments as above, graph $G'' = G' - M$ is in $\mathcal{G}_{\text{poly}'}$. Since G is obtained from G'' by adding the vertices of M together with a suitable set of edges incident to M, the conclusion follows. □

We deduce that Theorem 9 can be extended to the classes of graphs $\mathcal{G}_{\text{poly}} - ke$ and $\mathcal{G}_{\text{poly}} + ke$. On the other hand, the generic problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ is polynomial on $\mathcal{G}_{\text{poly}} - kv$, if k is small.

5.5 Deletion to $\mathcal{G}_{\text{poly}}$

One of the requirements of using our FPT algorithm is that the modulator of size at most k must be explicitly given in the input. In this section we show that for any polynomial poly there is no FPT algorithm computing a modulator to $\mathcal{G}_{\text{poly}}$, unless $W[2]$ is contained in FPT.

Let us consider the following problem, for some fixed polynomial poly.

Deletion to $\mathcal{G}_{\text{poly}}$
Input: A graph $G = (V, E)$ and a polynomial poly
Parameter: k
Question: A vertex subset M of size at most k, such that $G - M$ is in $\mathcal{G}_{\text{poly}}$

We will show that Deletion to $\mathcal{G}_{\text{poly}}$ is $W[2]$-Hard, reducing to it the problem Hitting set parameterized by the size of the solution. Our reduction is inspired in a similar reduction for Deletion to Weakly Chordal [84]. Intuitively, we represent an instance of hitting set $I = (U, \mathcal{A}, k)$ as a graph $G_I = (V, E)$, where $U \subset V$, and each $A \in \mathcal{A}$ will be represented (by non-necessarily disjoint) cycle C_A. The idea is that for two sets $A_1, A_2 \in \mathcal{A}$, cycles $C_{A_1} \cap C_{A_2} = A_1 \cap A_2$. Then, a solution X of instance I will intersect every cycle of G_I representing the sets of $\mathcal{A}$. The reduction on the number of minimal

separators follows, since a cycle of length l provide $\mathcal{O}(l^2)$ minimal separators, while a path provide only $\mathcal{O}(l)$ minimal separators. To obtain a proof that is valid for every polynomial $\text{poly}(n) \geq n$, we replace the gadgets of each edge in a cycle C_A by a watermelon graph, obtaining that $G_I \notin \mathcal{G}_{\text{poly}}$ and $G_I - X \in \mathcal{G}_{\text{poly}}$, where X is a solution of HITTING SET on instance I.

Let $I = (U, \mathcal{A}, k)$ be an instance of the problem HITTING SET, i.e., $\mathcal{A} = \{A_1, \ldots, A_m\}$ is a family of subsets of an universe U of size n, and k is a positive integer. The question is whether exists a set $X \subset U$ such that $X \cap A_i \neq \emptyset$ for all $i \in [m]$. Without loss of generality we assume that $\forall u \in U$ there exists $i \in [m]$ such that $u \in A_i$.

Let poly be a polynomial such that there exists $t_0 > 0$ for which it becomes an increasing function, i.e., $\text{poly}(t_2) > \text{poly}(t_1)$ for every $t_2 > t_1 > t_0$. Call g the degree of poly, and $p = \lfloor \log(\text{poly}((nm)^2 - n)/(n^2 m)) \rfloor$. We suppose that m and n are big enough to satisfy the following inequalities:

$$(mn)^2 - n > t_0, \tag{5.8}$$

$$\frac{(nm)^2 - n}{nm} > 4p + 1 \tag{5.9}$$

and

$$\left(1 - \frac{1}{nm^2}\right)^g > 2/3, \quad \text{i.e.,} \quad nm^2 > \frac{1}{1 - (2/3)^{1/g}}. \tag{5.10}$$

Let $I = (U, \mathcal{A})$ be an instance of hitting set. We define G_I to be the graph of size $(nm)^2$ defined as follows (see Figure 5.4).

- The vertex set of G_I is divided in $m + 2$ parts: a copy of U, a set called R and $\tilde{C}_i$, for $i \in [m]$.
- For each $i \in [m]$, let s_i be the size of A_i. Let C_i be a cycle of length $2s_i$ such that if $C_i = \{c_1, \ldots, c_{2s_i}\}$ and $A_i = \{a_1, \ldots, a_{s_i}\}$, then c_{2j} equals a_j, for each $j \in \{1, \ldots, s_i\}$.
- Call $\tilde{C}_i$ the graph obtained replacing each edge of C_i with a watermelon graph $W_{2,p}$ depicted in Figure 5.3. The replacement means that if $\{x, y\}$ is an edge of C_i, we remove that edge, make a copy of the watermelon graph $W_{2,p}$, and identify vertices x and y with the one shown in Figure 5.3. We say that C_i is the cycle associated to A_i and $\tilde{C}_i$ is the super-cycle associated to A_i.

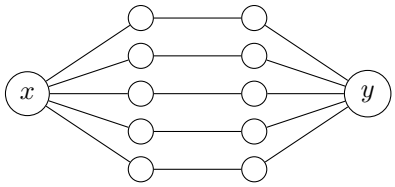


Figure 5.3 – Watermelon graph $W_{2,p}$, when $p = 5$. The watermelon graph consists in a set of p disjoint paths of length 4, plus two other vertices connected to different endpoints of each path

Note that $|\tilde{C}_i| = (4p + 2)|A_i|$, U is contained in $\bigcup_{i \in [m]} \tilde{C}_i$, and that

$$\left| \bigcup_{i \in [m]} \tilde{C}_i \right| = |U| + \sum_{i \in [m]} (4p + 1)|A_i| \leq n + nm(4p + 1) < (nm)^2$$

.

- For each pair $i, j \in [m]$ with $i \neq j$, add an edge incident to each pair of vertices $\{x, y\}$ such that $x \in \tilde{C}_i \setminus U$ and $y \in \tilde{C}_j \setminus U$.
- Add a set R of $(mn)^2 - (n + \sum_{i \in [m]} (4p+1)|A_i|)$ vertices, adjacent to every other vertex in $R \cup U \cup \bigcup_{i \in m} \tilde{C}_i$. These vertices are added to G_I in order to obtain a graph of size $(nm)^2$ and simplify the rest of the arguments. Note that $(mn)^2 > (n + \sum_{i \in [m]} (4p + 1)|A_i|)$ thanks to Equation 5.9.

In the following two lemmas, we show that G_I has a modulator M of size k to $\mathcal{G}_{\text{poly}}$ if and only if I is a *yes*-Instance of HITTING SET.

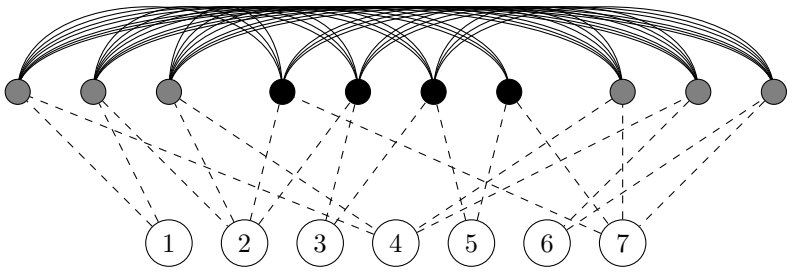


Figure 5.4 – Graph G_I on input $I = (\{1,2,3,4,5,6,7\}, \{\{1,2,4\},\{2,3,5,7\},\{4,6,7\}\}, k)$. Note that I is a *Yes*-instance if $k = 2$ (a solution is e.g., $X = \{2,7\}$) but a *No*-Instance if $k = 1$. Set U is represented in the white numbered nodes, and the other nodes in C_1, C_2 and C_3 in black and gray (the different colors are to help the recognition of the cycles in the figure). In this figure, only the cycles (and not the super-cycles) associated to the sets are represented. To represent the super-cycles, dashed edges must be replaced with a watermelon graph, with $p = \lfloor \log(poly((nm)^2 - n)/(nm)) \rfloor$. The nodes and edges incident to R, and the edges incident to the nodes in the dashed edges were omitted.

Lemma 16. *For each $i \in [m]$, graph $G_I[\tilde{C}_i]$ has more than* $\text{poly}((nm)^2)$ *minimal separators. Therefore, G_I does not belong to $\mathcal{G}_{\text{poly}}$.*

Proof. For each pair of edges of the cycle associated to A_i there are $(2^p)^2$ possible choices for a minimal separator contained in the watermelon graph corresponding to those edges in the super-cycle $\tilde{C}_i$. Hence the number of minimal separators contained in $G_I[\tilde{C}_i]$ is at least

$$|\Delta_{G_I[\tilde{C}_i]}| \geq \frac{|A_i|(|A_i|-1)}{2} \cdot (2^p)^2 \geq \frac{|A_i|^2}{4} \cdot \frac{\text{poly}((nm)^2 - n)^2}{(nm)^2}.$$

We claim that $\text{poly}((nm)^2 - n) \geq (2/3) \cdot \text{poly}((nm)^2)$. Indeed $(nm)^2 - n = (nm) \cdot (1 - 1/(nm^2))$. Therefore $\text{poly}((nm)^2 - n) \geq (1 - 1/(nm^2))^g \cdot \text{poly}((nm)^2) > (2/3) \cdot poly((nm)^2)$. Where the last inequality follows from Equation 5.10. Then

$$|\Delta_{G_I[\tilde{C}_i]}| \geq \frac{|A_i|^2}{4(nm)^2} \cdot \frac{4}{9} \cdot \text{poly}((nm)^2)^2 \geq \frac{|A_i|^2}{9} \cdot \frac{\text{poly}((nm)^2)^2}{(nm)^2} \geq poly((nm)^2).$$

Where the last inequality follows from the fact that $|A_i| \geq 3$ and $\text{poly}((nm)^2) \geq (nm)^2$. Since a graph has at least the number of minimal separators of any of its induced subgraphs, we deduce that G_I is not in $\mathcal{G}_{\text{poly}}$. □

Lemma 17. *An instance $I = (U, \mathcal{A}, k)$ is a* Yes*-Instance of* Hitting Set *if and only if there exists a set M of k vertices of G_I such that $G_I - M$ belongs to $\mathcal{G}_{\text{poly}}$.*

Proof. Suppose that there exists a set M of k vertices of G_I such that $G_I - M$ is in $\mathcal{G}_{\text{poly}}$. From Lemma 16, the number of minimal separators of $G_I[\tilde{C}_i]$ is greater than $\text{poly}((nm)^2)$, for each $i \in [m]$. This implies that necessarily M must intersect $\tilde{C}_i$, for each $i \in [m]$. Without loss of generality, we may assume that for each $i \in [m]$, the intersection $M \cap C_i$ is contained in U. Indeed, for each $i \neq j$, $(\tilde{C}_i \cap \tilde{C}_j) \setminus U = \emptyset$, so it is always more convenient to pick elements in U to break the cycles. We conclude that $M \cap A_i \neq \emptyset$ for all $i \in [m]$, so $X = M \cap U$ is a solution of instance $(U, \mathcal{A}, k)$ of Hitting Set.

Conversely, let $X \subseteq U$, $|X| \leq k$, be such that for each $i \in [m]$, $X \cap A_i \neq \emptyset$. Consider the graph $H = G_I - X$. Let x, y be two vertices of H and let S be a x,y-minimal separator of H. We claim that either (1) S is the neighborhood of x or y, i.e $S = N_H(x)$ or $S = N_H(y)$, or (2) there exists $i \in [m]$ such that $S = \tilde{S} \cup R \cup (\bigcup_{j \neq i} \tilde{C}_j \setminus U)$, where $\tilde{S}$ is a minimal separator of $H[\tilde{C}_i \setminus X]$.

There are two kind of nonadjacent vertices x, y in H:

- For any $i \in [m]$ the pair $\{x, y\}$ is not contained in $\tilde{C}_i$. In this case either $S = N(x)$ or $S = N(y)$. Note that $x \in U$ or $y \in U$, otherwise they are adjacent. If x belongs to U and y belongs to $V_H \setminus (U \cup R)$ then $S = N(x)$. Otherwise x, w, y is a x,y-path in $H - S$, for any $w \in N(x) \setminus S$. If x and y belong to U, and neither $S = N(x)$ or $S = N(y)$, then x, w_1, w_2, y is an x,y-path in $H - S$, for any $w_1 \in N(x) \setminus S$ and any $w_2 \in N(y) \setminus S$ (possibly $w_1 = w_2$).

- There exits $i \in [m]$ such that x, y belong to $\tilde{C}_i$. Suppose that neither $S = N(x)$ or $S = N(y)$. Let D_x and D_y be the components of $H[\tilde{C}_i \backslash X]$ containing x and y, respectively. Note that $|D_x| > 1$ and $|D_y| > 1$, because otherwise S contain (and for minimality is) the neighborhood of either x or y. Hence there exists $w_x \in D_x$ and $w_y \in Dy$ such that one of $\{x, w_x\}$ and one of $\{y, w_y\}$ does not belong to U. This implies that S must contain R and $\bigcup_{i \neq j} \tilde{C}_j$. Since $\tilde{S} = S \cap C_i$ is a minimal separator of $H[\tilde{C}_i \setminus X]$, we conclude that $S = \tilde{S} \cup R \cup (\bigcup_{i \neq j} \tilde{C}_j)$. This proves the claim.

We obtain that the number of minimal separators of H is at most

$$\sum_{i \in [m]} |A_i|(2^p + 1) < nm \cdot \frac{\text{poly}((nm)^2 - n)}{nm} \leq \text{poly}((nm)^2 - n) \leq \text{poly}((nm)^2 - k).$$

Since H is of size $(nm)^2 - k$, we conclude that H belongs to $\mathcal{G}_{\text{poly}}$. □ □

The following theorem is a direct consequence of Lemma 17 and the fact that Hitting Set parameterized by the size of the solution is $W[2]$-Hard [56].

Theorem 10. *Let* poly *be a polynomial such that* $\text{poly}(n) \geq n$ *for every big enough* n. *Then* Deletion to $\mathcal{G}_{\text{poly}}$ *is* $W[2]$*-Hard.*

5.6 Discussion

We gave an FPT algorithm for the problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ on $\mathcal{G}_{\text{poly}} + kv$, under the assumption that the modulator is part of the input. The problem encompasses many classical ones [67]. Moreover, the result can be extended to the classes of graphs $\mathcal{G}_{\text{poly}} - ke$ and $\mathcal{G}_{\text{poly}} + ke$ (i.e., graphs of $\mathcal{G}_{\text{poly}}$ minus or plus ar most k edges), and the generic problem is polynomial on $\mathcal{G}_{\text{poly}} - kv$, if k is small.

Later, we have shown that the problem consisting in the computing of the modulator, that we called Deletion to $\mathcal{G}_{\text{poly}}$, is $W[2]$-Hard. This result inhibits us to conclude that the problem Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ is FPT on the class $\mathcal{G}_{\text{poly}} + kv$, without needing to require to have the modulator M as part of the input. However, we would like to mention that for some particular classes with polynomially many minimal separators, the deletion problem is FPT. For example Deletion to Chordal is FPT [111]. However for other classes, this is not the case, e.g., problem Deletion to Weakly chordal is $W[2]$-hard [84].

One way that we would may extend this research is related with the complexity of our FPT algorithm, which is

$$\mathcal{O}\left(f(k+t, \mathcal{P}) \cdot n^{t+5} \cdot (\text{poly}(n)^2)\right).$$

The dependency on $\mathcal{P}$ and $t + k$ comes from Courcelle's theorem (Proposition 4), applied for deciding property $\mathcal{P}$ on graphs of treewidth $t+k$. As shown by Frick and Grohe [72], function f can be very huge, typically a tower of exponentials in $t+k$, the height of the tower depending on the property to be checked. Our algorithm actually constructs an induced subgraph of treewidth t, although we were only able to build a decomposition of width $t + k$. In particular, if we do not need to check a particular property $\mathcal{P}$ on the induced graph, but we only ask this graph to be of treewidth at most t, then function f becomes $(k+t)^{O((k+t)^3)}$ — coming from the algorithm of Bodlaender and Kloks [24] that, given a graph and tree decomposition of width $k + t$, checks whether the treewidth of the graph is at most t.

For easier cases, when $t = 0$ to $t = 1$, the function f becomes 2^k and $(k+t)^{O(k+t)}$ respectively. Also, for natural properties $\mathcal{P}$, like "being connected" or "being a path" one can perform the property checking using standard (ad-hoc) dynamic programming tools which avoid the heavy machinery of Proposition 4. Again, the extra-cost becomes much more reasonable.

A natural and challenging question would be to separate the dependency on t and k, typically to obtain a complexity of type $f(t, \mathcal{P}) \cdot g(k) \cdot n^{t+\mathcal{O}(1)}$, where g would be a "more reasonable" function. For that purpose we would need to construct the partial solutions as a $(t+1)$-terminal recursive graph, maybe by a more clever way of dealing with the intersection between this solution and the modulator.

We finish this chapter with the open question whether Treewidth on $\mathcal{G}_{\text{poly}} + kv$ is FPT, when the modulator is a part of the input. This question was the initial motivation of this research, and was posed by Bart Jansen in a private communication, in the particular case when the input graph belongs to Chordal $+ kv$.

Chapter 6

Distance-d Independent Set and extensions

This chapter is composed of two parts. In the first, we give a result of combinatorial nature, showing that the odd powers of a graph G have at most as many minimal separators as G. Consequently, Distance-d Independent Set, which consists in finding maximum set of vertices at pairwise distance at least d, is polynomial on $\mathcal{G}_{\text{poly}}$, for any even d. This problem was posed by Eto *et al.* in [59], where the authors show that problem is NP-hard on chordal graphs for any odd $d \geq 3$ [59], and ask whether this problem is polynomial in chordal-bipartite graphs for even d.

Later, we study extensions of Proposition 22 into a wider set of problems. In [67] the authors asked if Proposition 22 can be extended in order to solve problems like Connected Vertex Cover or Connected Feedback Vertex Set. The this part of the chapter are of exploratory nature, and some of the results are not very conclusive. We show that the mentioned problems are indeed polynomial on chordal and circular-arc graphs. We finalize discussing variants of independent domination problems.

The results of this chapter were published in the 42st International Workshop of Graph-Theoretic Concepts in Computer Science (WG 2016)and were obtained in collaboration with Ioan Todinca [113].

6.1 Introduction

This chapter explores extensions of the dynamic programming over minimal separators and potential maximal cliques into a wider family of problems. The goal is to establish the limits of this technique, exploring more general problems or new properties of graphs with a few minimal separators.

Recall that Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$, roughly consists in find the maximum induced subgraph of treewidth t satisfying a property $\mathcal{P}$. In [67] the authors asked whether Proposition 22 can be extended in order to warrant some property of *the rest of the graph* $G - F$. For example, Connected Vertex Cover consists in finding a maximum independent set F, whose *complement* $G - F$ is connected. We explore this question providing polynomial algorithms for some specific classes of graphs with a few minimal separators.

Let $\mathcal{Q}(G, F)$ be a property assigning to each graph G and vertex subset F of G a Boolean value. We ask, for which kinds of properties $\mathcal{Q}$ we can solve the following problem when the input graph has a few minimal separators:

Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ and $\mathcal{Q}$
Input: A graph G.
Task: Compute

max	$\lvert F\rvert$
subject to	The treewidth of $G[F]$ is at most t;
	$G[F] \models \varphi$;
	$\mathcal{Q}(G, F) = 1$.

Clearly, we will have to impose some restrictions in order to accomplish our task, otherwise the problem quickly becomes NP-Hard, even if $\mathcal{Q}$ is regular. Notice that if $\mathcal{Q}$ is a regular property, so from Lemma 13, property $\mathcal{P}'(G, F, X) = \mathcal{P}(G, X) \wedge \mathcal{Q}(G, F)$ is regular. However, we have no bounds on the treewidth of graph $G - F$, so the set of characteristics may be exponential in the size of the graph. Moreover, if we choose $t = 0$, we do not ask any formula φ, and $\mathcal{Q}$ is defined as the property "F is a dominating set of G", then our problem becomes Independent Dominating Set, which is NP-hard on chordal graphs [50].

We obtained some positive results, either for particular properties $\mathcal{Q}$ and/or by restricting ourselves to specific classes of graphs with a few minimal separators.

First, we consider the problem Distance-d Independent Set on $\mathcal{G}_{\text{poly}}$, where the goal is to find a maximum independent set F of the input graph G, such that the vertices of F are at pairwise distance at least d in G (in the literature this problem is also known as Scattered-Set). In this case, the second property $\mathcal{Q}$ could be stated as "the connected components of $G[F]$ are at distance at least d in G". This problem is equivalent to finding a maximum independent set in graph G^{d-1}, the $(d-1)$-th power of G. Eto *et al.* [59] already studied the problem on chordal graphs, and proved that it is polynomial for every even d, and NP-hard for any odd $d \geq 3$ (it is even $W[1]$-hard when parameterized by the solution size). Their positive result is based on the observation that for any even d, if G is chordal then so is G^{d-1}. Eto *et al.* [59] ask if Distance-d Independent Set is polynomial on chordal bipartite graphs (which are *not* chordal but weakly chordal, see Section 3.2), a subclass of $\mathcal{G}_{\text{poly}}$. We bring a positive answer to their question for even values d, by a result of combinatorial nature: for any graph G and any odd k, the graph G^k has no more minimal separators than G (see Section 6.2). Consequently, Distance-d Independent Set is polynomial on $\mathcal{G}_{\text{poly}}$, for any even value d and any polynomial poly, and NP-hard for any odd $d \geq 3$ and any poly(n) asymptotically larger than n. Such a dichotomy between odd and even values also appears when computing large d-clubs, that are induced subgraphs of diameter at most d [78], and for quite similar reasons.

Second, we consider Connected Vertex Cover, Connected Feedback Vertex Set and more generally the problem of finding a maximum induced subgraph $G[F]$ of treewidth at most t, such that $G - F$ is connected (i.e., in this case $\mathcal{Q}$ is "$G - F$ is connected"). We show (Section 6.3) that the problems are polynomially solvable for subclasses of $\mathcal{G}_{\text{poly}}$, like chordal and circular-arc graphs. This does not settle the complexity of these problems on $\mathcal{G}_{\text{poly}}$. As we shall discuss in Section 6.4, when restricted to bipartite graphs in $\mathcal{G}_{\text{poly}}$, Connected Vertex Cover can be reduced from Red-Blue Dominating Set (see [55]). It might be that this latter problem is NP-hard on bipartite graphs of $\mathcal{G}_{\text{poly}}$; that was our hope, since as we said the very related problem Independent Dominating Set is NP-hard on chordal bipartite graphs [50], and on circle graphs [34]. This question is still open, however we will observe that the Red-Blue Dominating Set is polynomial on the two natural classes of bipartite graphs with polynomially many minimal separators: chordal bipartite and circle bipartite graphs.

6.2 Powers of graphs with polynomially many minimal separators

Let us prove that for any odd k, graph G^k has no more minimal separators than G.

Theorem 11. *Consider a graph G, an odd number $k = 2l + 1$ with $l \geq 0$, and a minimal separator $\overline{S}$ of G^k. Then there exists a minimal separator S of G such that $\overline{S} = N_G^l[S]$.*

Proof. The lemma is trivially true if $\overline{S} = \emptyset$. Let $a, b \in V$ such that $\overline{S} \neq \emptyset$ is an a, b-minimal separator in G^k, and call C_a, C_b the components of $G^k - \overline{S}$ that contain a and b, respectively. Let us call $D_a = N_G^l[C_a]$ and $D_b = N_G^l[C_b]$.

Claim 1: $dist_G(D_a, D_b) \geq 2$.

Suppose that $dist_G(D_a, D_b) < 2$, and pick $x \in D_a, y \in D_b$ with $dist_G(x, y) \leq 1$ (notice that possibly $x = y$). Let $x_a \in C_a$ and $x_b \in C_b$ be such that there exists an x_a, x-path and a y, x_b -path in G, each one of length at most l, called P_a and P_b, respectively. This implies that there must be a x_a, x_b-path of length at most $2l + 1 = k$ in G, which means that $\{x_a, x_b\} \in E(G^k)$, a contradiction with the fact that $\overline{S}$ separates C_a from C_b in G^k.

Claim 2: $\tilde{S} = \overline{S} \setminus (D_a \cup D_b)$ separates a and b in G.

Notice first that $N_G(D_a) \subseteq N_G^{l+1}(C_a) \subseteq \overline{S}$. Suppose that $\tilde{S}$ does not separate a and b, and let P be an a, b-path in G that does not pass through $\tilde{S}$. Let $x_1, \ldots, x_{s-2}$ the internal nodes of P, where $s = |P|$,

and consider $i = \max\{j \mid x_j \in D_a \cap P\}$. Since $P \cap \tilde{S} = \emptyset$, necessarily $x_{i+1} \in D_b$, a contradiction with Claim 1.

Claim 3: D_a and D_b are connected subsets of G.

This is straightforward from the definition of the sets, $D_a = N_G^l[C_a]$ and $D_b = N_G^l[C_b]$, and the fact that C_a and C_b are connected in G.

Let $\tilde{C}_b$ be the connected component of $G - N_G[D_a]$ that contains b, and denote $S = N_G(\tilde{C}_b)$. Note that $S \subseteq \tilde{S} \subset \overline{S}$. By applying Proposition 9, we have that S is a minimal a,b-separator in G. Call $\tilde{C}_a$ the component of $G - S$ that contains a. Since $S \subseteq \tilde{S}$, we have that $D_b \subseteq \tilde{C}_b$ and $D_a \subseteq \tilde{C}_a$.

Claim 4: $N_G^l[S] = \overline{S}$.

We first prove that $N_G^l[S] \subseteq \overline{S}$. By construction, $S \subseteq N_G(D_a)$. Consequently $S \subseteq N_G^{l+1}(C_a) \backslash N_G^l(C_a)$, therefore $N_G^l[S] \subseteq N_G^{2l+1}(C_a) = N_{G^k}(C_a) = \overline{S}$.

Conversely, we must show that every vertex x of $\overline{S}$ is in $N_G^l[S]$.

By contradiction, let $x \in \overline{S} \setminus N_G^l[S]$. We distinguish two cases : $x \in \tilde{C}_a$, and $x \in \overline{S} \setminus \tilde{C}_a$. In the first case, since $N_{G^k}(C_b) = \overline{S}$, there exists a path from some vertex $y \in C_b$ to x of length at most k, in graph G. Let us call P one of those y,x-paths. Observe that the first $l+1$ vertices of the path belong to $D_b \subseteq \tilde{C}_b$, and none of the last $l+1$ vertices of the path belongs to S (otherwise $x \in N_G^l[S]$). Then P is a path that connects $\tilde{C}_b$ with $\tilde{C}_a$ without passing through S, a contradiction with the fact that S separates a and b in graph G.

It remains to prove the last case, when $x \in \overline{S} \setminus \tilde{C}_a$. Since $N_{G^k}(C_a) = \overline{S}$, there exists a node $y \in C_a$ such that there is a y,x-path P of length at most k in G. Since the first $l+1$ vertices of the path belong to D_a, and the last $l+1$ vertices of the path do not belong to S, we deduce that P is an y,x-path in G that does not intersect S. The path can be extended (through C_a) into an a,x-path that does not intersect S, a contradiction with the fact that x does not belong to $\tilde{C}_a$. This concludes the proof of our theorem. □

Recall that Distance-d Independent set on G is equivalent to Maximum Independent Set on G^{d-1}. Since the latter problem is polynomial on $\mathcal{G}_{\text{poly}}$ by Proposition 21, we deduce:

Theorem 12. *For any even value d, and any polynomial* poly*, problem* Distance-d Independent set *is polynomially solvable on $\mathcal{G}_{\text{poly}}$.*

We remind that for any odd value d, problem Distance-d Independent set is NP-hard on chordal graphs [59], thus on $\mathcal{G}_{\text{poly}}$ for any polynomial poly asymptotically larger than n. The construction of [59] also shows that even powers of chordal graphs may contain exponentially many minimal separators.

Let $G = (V, E)$ be an arbitrary graph of size n and call $H(G)$ the graph on $|V| + |E|$ vertices defined as follows. We add to $H(G)$ one vertex for each vertex and each edge in G, i.e., the vertex set of $H(G)$ is $V \cup E$. Then we add an edge between every pair of vertices u, v of $H(G)$ such that u represent a vertex of G and v represents an edge of G which has u as an endpoint. Finally we add and edge between every pair of vertices representing edges. An example of this construction is depicted in Figure 6.1.

Note that $H(G)$ is a split graph. Note also that $u, v \in V$ are adjacent in G if and only if u and v are at distance 2 in $H(G)$. Therefore $H(G)^2[V] = G$, so $H(G)^2$ contain at least the same number of minimal separators than G. Therefore, if we pick G that not contained in $\mathcal{G}_{\text{poly}}$ for some polynomial poly, we obtain that $H(G)$ has a linear number of minimal separators, while $H(G)^2$ does not belong to $\mathcal{G}_{\text{poly}}$.

6.3 On Connected Vertex Cover and Connected Feedback Vertex Set

Let us consider the problem of finding a maximum induced subgraph $G[F]$ such that $\text{tw}(G[F]) \leq t$ and $G - F$ is connected. One can easily observe that, for $t = 0$ (resp. $t = 1$), this problem is equivalent to Connected Vertex Cover (resp. Connected Feedback Vertex Set), in the sense that if F is an optimal solution for the former, than $V(G) - F$ is an optimal solution for the latter.

Our goal is to enrich the dynamic programming scheme described in Section 3.3.2 in order to ensure the connectivity of $G - F$. One should think of this dynamic programming scheme as similar to dynamic programming algorithms for bounded treewidth. The difference is that the bags (here, the potential

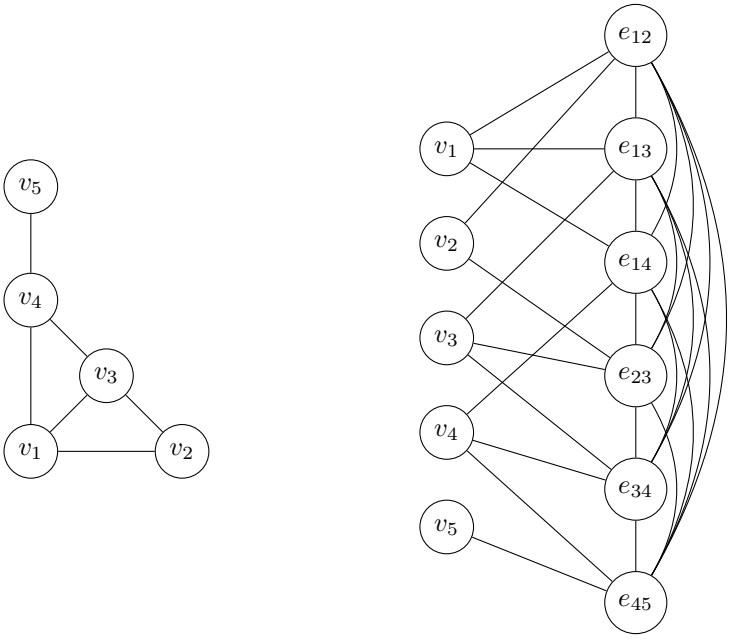


Figure 6.1 – Example of the construction of graph $H(G)$ (in the right) when G is the graph in the left.

maximal cliques) are not small but polynomially many, and we parse simultaneously through a set of decompositions. Nevertheless, we can borrow several classical ideas from treewidth-based algorithms.

In general, for checking some property for the solution F, we considered the notion of *homomorphism class* of partial solutions. In literature, the homomorphism classes are also known as *characteristics*. To distinguish the homomorphism classes of property $\mathcal{P}$ form the ones of $\mathcal{Q}$, in the following we call characteristics the homomorphism classes of $\mathcal{Q}$

As usual in dynamic programming, the characteristics must satisfy several properties: (1) we must be able to compute the characteristic for the base case, (2) the characteristic of a partial solution F obtained from gluing smaller partial solutions F_i must only depend on the characteristics of F_i, and (3) the characteristic of a global solution should indicate whether it is acceptable or not. Moreover, for a polynomial algorithm, we need the set of possible characteristics to be polynomially bounded.

For simplicity, in the following we suppose that we are solving the problem only for the connectivity property, ignoring property $\mathcal{P}$, i.e., we are solving Max. Induced Connected Subgraph of tw $\leq t$. It is easy to extend the result asking also for a solution satisfying property $\mathcal{P}$ following Section 3.3.2.

For checking connectivity conditions on $G - F$, we define the characteristics of partial solutions in a natural way[1]. Consider a block (S, C) (resp. a good triple (S, C, Ω)) and a subset W of S (resp. of Ω). Let F be a partial solution compatible with (S, C, W) (resp. (S, C, Ω, W)), see Definition 3. The characteristic c of F and property $\mathcal{Q}$ (connectivity) for (S, C, W) (resp. for (S, C, Ω, W)) is defined as the partition induced on $S \setminus W$ (resp. on $\Omega \setminus W$) by the connected components of $G[S \cup C] - F$. More formally, let $D_1, \ldots, D_q$ denote the connected components of $G[S \cup C] - F$, and let $P_j = D_j \cap S$ (resp. $P_j = D_j \cap \Omega$), for all $1 \leq j \leq q$. Then $c' = \{P_1, \ldots, P_q\}$. We decide that if $S \neq \emptyset$, partial solutions F having some component D_j that does not intersect S (resp. Ω) are immediately rejected; indeed, for any extension F' of F, the graph $G - F'$ remains disconnected. Hence we may assume that all sets P_j are non-empty.

We say that a partial solution F is *compatible with* (S, C, W, c) (resp. *with* (S, C, Ω, W, c)) if it satisfies the conditions of Defintion 3, and c is the characteristic of F for (S, C, W) (resp. for (S, C, Ω, W)).

We also define functions $\alpha(S, C, W, c)$, $\beta(S, C, \Omega, W, c)$ and $\gamma_i(S, C, \Omega, W, c)$ like in Subsection 3.3.2, as the maximum size of partial solutions F compatible with the parameters. We can update Equations 3.1to 3.7 as follows.

[1]We assume, in this section, that the reader is familiar with the dynamic programming schemes over bounded treewidth graphs, which consider the connectivity property of the solution (e.g., connected vertex cover).

Base case. For the good triples (S, C, Ω) such that (S, C) is inclusion-minimal (hence $\Omega = S \cup C$),

$$\beta(S, C, \Omega, W, c, c') = |W| \text{ if } c' \text{ corresponds to the connected components of } G[\Omega \setminus W]. \tag{6.1}$$

Otherwise we set $\beta(S, C, \Omega, W, c, c') = -\infty$.

Computing α from β.

$$\alpha(S, C, W, c) = \max_{\Omega, W', c'} \beta(S, C, \Omega, W', c'), \tag{6.2}$$

where the maximum is taken over all potential maximal cliques Ω such that (S, C, Ω) is a good triple, and all subsets W' of Ω, of size at most $t+1$, such that $W = W' \cap S$, and all characteristics c' such that each part of c corresponds to the intersection between S and a part of c'. If $S \neq \emptyset$ we also request that each part of c' intersects S. This condition allows to reject partial solutions F for which a component of $G[S \cup C] - F$ is strictly contained in C. Indeed, such partial solutions cannot extend to global solutions F' such that $G - F'$ is connected.

For the particular case $S = \emptyset$ (hence $C = V$ and $W = \emptyset$) we only consider characteristics c' with a single part. This ensures that the global solution F satisfies that $G - F$ is connected.

If there is no such triple (Ω, W', c'), then we set $\alpha(S, C, W, c) = -\infty$ (we can assume that, when it has no parameters, function max returns $-\infty$).

Computing β from α.

$$\gamma_1(S, C, \Omega, W, c) = \max_{c'}(\alpha(S_1, C_1, \Omega, W \cap S_1, c') + |W| - |W \cap S_1|), \tag{6.3}$$

over all characteristics if c' that *map correctly* on c, in the following sense. Consider a characteristic c' and let $G_{c'}[\Omega \setminus W]$ be the graph obtained from $G[\Omega \setminus W]$ by completing each part $D \in c'$ into a clique. We say that a characteristic c' maps correctly on c if c is the partition of $\Omega \setminus W$ defined by the connected components of $G_{c'}[\Omega \setminus W]$.

The notion of mapping transforms the characteristic of the partial solution F_1 w.r.t. $(S_1, C_1, W \cap S_1)$ into the characteristic of $F_1 \cup W$ w.r.t. the quadruple (S, C, Ω, W).

For all i, $2 \leq i \leq p$,

$$\gamma_i(S, C, \Omega, W, c) = \max_{c_{i-1}, c_i} (\gamma_{i-1}(S, C, \Omega, W, c_{i-1}) + \alpha(S, C, \Omega, W \cap S_i, c_i) - |W \cap S_i|), \tag{6.4}$$

over all pairs of characteristics c_{i-1}, c_i that *map correctly* on c. That is, c must correspond to the connected components of $G_{c_{i-1}, c_i}[\Omega \setminus W]$, obtained from $G[\Omega \setminus W]$ by completing each part of c_{i-1} and each part c_i of into a clique.

Finally

$$\beta(S, C, \Omega, W, c) = \gamma_p(S, C, \Omega, W, c). \tag{6.5}$$

The optimal solution size is, as before, $\alpha(\emptyset, V, \emptyset, \{\emptyset\})$.

In general, the number of characteristics may be exponential. Nevertheless, there are classes of graphs with the property that each minimal separator S and each potential maximal clique Ω can be partitioned into at most a constant number of cliques. With this constraint, the number of characteristics is polynomial (even constant, for any given triple (S, C, W) or quadruple (S, C, Ω, W)).

This is the case for chordal graphs, where each minimal separator and each potential maximal clique induces a clique in G.

It is also the case for circular-arc graphs. Recall that each minimal separator corresponds to the set of arcs intersecting a pair of scanpoints [92]. Moreover, by [92, 32], each potential maximal clique corresponds to the set of arcs intersecting a triple of scanpoints. Since arcs intersecting a given scanpoint form a clique, we have that each minimal separator can be partitioned into two cliques, and each potential maximal clique can be partitioned into three cliques.

We deduce:

Theorem 13. *On chordal and circular-arc graphs, problems* Connected Vertex Cover *and* Connected Feedback Vertex Set *are solvable in polynomial time. More generally, one can compute in polynomial time a maximum vertex subset F such that $G[F]$ is of treewidth at most t and $G - F$ is connected.*

Note that Escoffier *et al.* [58] already observed that Connected Vertex Cover is polynomial for chordal graphs.

Theorem 13 does not settle the complexity of the connectivity condition of the rest of the graph for any class of graphs with a few minimal separators.

Consider the problem Red-Blue Dominating Set. In this problem we are given a bipartite graph $G = (R, B, E)$ with red and blue vertices, and an integer k, and the goal is to find a set of at most k red vertices dominating all the blue ones. Red-Blue Dominating Set can be reduced to Connected Vertex Cover as follows [55]. Let G' be the graph obtained from $G = (R, B, E)$ by adding a new vertex u adjacent to all vertices of R and then, for each $v \in B \cup \{u\}$, a pendant vertex v' adjacent only to v. Then G has a red-blue dominating set of size at most k if and only if G' has a connected vertex cover of size at most $k + |B| + 1$. Indeed any minimum connected vertex cover of G' must contain u, B, and a subset of R dominating B (see Figure 6.2).

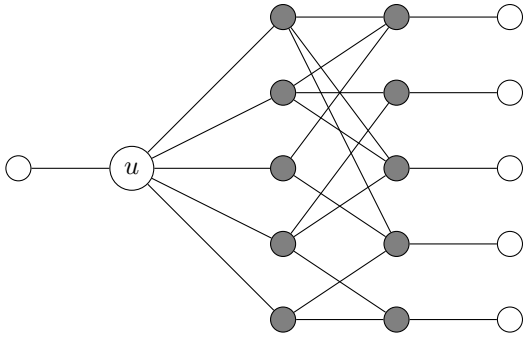


Figure 6.2 – Graph G' obtained in the reduction from Red-Blue Dominating Set on input G (represented in gray vertices) to Connected Vertex Cover

Lemma 18. *Let G' be the graph obtained from the bipartite graph $G = (R, B, E)$ by adding a new vertex u adjacent to all vertices of R and then, for each $v \in B \cup \{u\}$, a pendant vertex v' adjacent only to v.*

G' has at most $|V(G')|$ more minimal separators than G.

Proof. Let S be a minimal separator of G' and let C, D be two components of $G' - S$ such that $N_{G'}(C) = N_{G'}(D) = S$.

Assume first that there are two blue vertices $c \in C$ and $d \in D$. Then vertex u (seeing the whole set R) must be in S. No pendant vertex can be in S because vertices of S have neighbors in both C and D. Therefore $S \setminus \{u\}$ also separates c and d in G, it is even a c, d-minimal separator. Indeed, by removing the pendant vertices from C and D, we obtain two components of $G - S$ whose neighborhood in graph G is $S \setminus \{u\}$. Therefore, the number of minimal separators S of G' of this type is at most the number of minimal separators of G.

It remains to consider the case when one of the components, say C, does not contain any red vertex. Then C contains at most two vertices. Actually, C either contains a unique pendant vertex, or a vertex $v \in B \cup \{u\}$ and its pendant vertex (C cannot contain a unique vertex $v \in B \cup \{u\}$, because its pendant neighbor should be in S, or S cannot contain pendent vertices). Hence the number of minimal separators of this type is at most $|V(G')|$. □

Therefore, if Red-Blue Dominating Set is NP-hard on bipartite graphs in $\mathcal{G}_{\text{poly}}$ for some *poly*, so is Connected Vertex Cover.

We did not succeed to establish the complexity of the problem on bipartite $\mathcal{G}_{\text{poly}}$. There are two natural, well-studied classes of bipartite graphs with polynomial number of minimal separators, and it turns out that Red-Blue Dominating Set is polynomial for both. One is the class of chordal bipartite graphs (which are actually defined as the bipartite, *weakly* chordal graphs). For this class, Red-Blue Dominating Set is polynomial by [50]. Reference [50] considers the total domination problem for the class, but the approach is based on red-blue domination.

The second natural class is the class of circle bipartite graphs, i.e., bipartite graphs that are also circle graphs. They have an elegant characterization established by de Fraysseix [51]. Let $H = (V, E)$ be a planar multigraph, and partition its edge set into two parts E_R and E_B such that $T = (V, E_R)$ is

a spanning tree of H. Let $B(H, E_R) = (E_R, E_B, E')$ be the bipartite graph defined as follows: E_R is the set of red vertices, E_B is the set of blue vertices, and $e_R \in E_R$ is adjacent to $e_B \in E_B$ if the unique cycle obtained from the spanning tree T by adding e_B contains the edge e_R. We say that $B(H, E_R)$ is a fundamental graph of H. By [51], a graph is circle bipartite if and only if it is the fundamental graph $B(H, E_R)$ of a planar multigraph H.

Consider now the TREE AUGMENTATION problem that consists in finding, on input G and a spanning tree T of G, a minimum set of edges $D \subseteq E(G) - E(T)$ such that each edge in $E(T)$ is contained in at least one cycle of $G' = (V, E(T) \cup D)$. In [120] is shown that TREE AUGMENTATION is polynomial when the input graph is planar. Is direct to see that a set $S \subseteq E_B$ is a solution of the TREE AUGMENTATION problem on input $H = (V, E_R \cup E_B)$ and $T = (V, E_R)$, if and only if S is a solution of RED-BLUE DOMINATING SET on input $B(H) = (E_R, E_B, E')$. This observation, together with [51] and [120], impliy that RED-BLUE DOMINATING SET is polynomial in circle bipartite graphs.

6.4 INDEPENDENT DOMINATING SET and variants

The INDEPENDENT DOMINATING SET problem consists in finding a *minimum* independent set F of G such that F dominates G. Hence the solution F induces a graph of treewidth 0 and it is natural to ask if similar techniques work in this case. The fact that we have a minimization problem is not a difficulty: the general dynamic programming scheme applies in this case, and for any weighted problem with polynomially bounded weights, including negative ones [68, 67].

INDEPENDENT DOMINATING SET is known to be NP-complete in chordal bipartite graphs [50] and in circle graphs [34]. Therefore, it is NP-hard on $\mathcal{G}_{\text{poly}}$ for some polynomials poly. But, again, we can use our scheme in the case of circular-arc graphs, for this problem or any problem of the type MAX INDUCED SUBGRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$ and $\mathcal{Q}$, where $\mathcal{Q}$ is the property "F is a dominating set of G". We call this problem MINIMUM DOMINATING INDUCED GRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$.

The natural way for defining the homomorphism class of $\mathcal{Q}(G, F)$, when F is a partial solution compatible with (S, C, W, c) is to specify which vertices of S are dominated by F and which are not (we already know that $F \cap S = W$). It is thus enough to memorize which vertices of S are dominated by $F \cap C$. In circular-arc graphs, this information can be encoded using a polynomial number of classes. Indeed, a minimal separator S corresponds to arcs intersecting a scanline, between two scanpoints p_1 and p_2 of some intersection model of G. Moreover (see [92]), the vertices of component C correspond to the arcs situated on one of the sides of the scanline. Let $s^1_1, s^1_2, \ldots, s^1_{l_1}$ be the arcs of the model containing scanpoint p_1, ordered by increasing intersection with the side of p_1p_2 corresponding to C. Simply observe that if $F \cap C$ dominates vertex s^1_i, it also dominates all vertices s^1_j with $j > i$. Therefore we only have to store the vertex $s^1_{min_1}$ dominated by $F \cap C$ which has a minimum intersection with the side of the scanline corresponding to component C, and proceed similarly for the arcs of S containing scanpoint p_2. These two vertices of S will define the characteristic of F, and they suffice to identify all vertices of S dominated by $F \cap C$. These characteristics can be used to compute a MINIMUM DOMINATING INDUCED GRAPH OF tw $\leq t$ SATISFYING $\mathcal{P}$, for circular-arc graphs, in polynomial time.

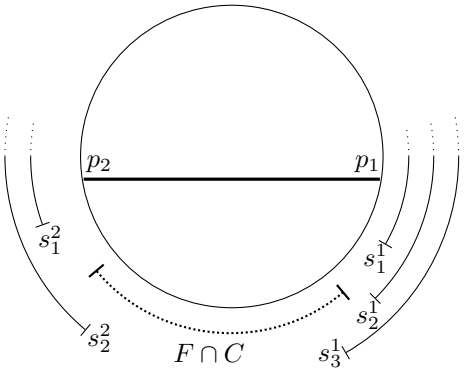


Figure 6.3 – Domination in circular-arc graphs. The characteristic of F w.r.t. (S, C) is (s^1_3, s^2_2).

Problem INDEPENDENT DOMINATING SET was already known to be polynomial for this class [41, 135].

The algorithm of Vatshelle [135] is more general, based on parameters called *Boolean-width* and *MIM-width*, which are small ($\mathcal{O}(\log n)$ for the former, constant for the latter) on circular-arc graphs and also other graph classes.

6.5 Discussion

We showed how the dynamic programming scheme of [68, 67] can be extended for other optimization problems, on subclasses of $\mathcal{G}_{\text{poly}}$. Note that the algorithm of [67] (Proposition 22) allows to find in polynomial time, on $\mathcal{G}_{\text{poly}}$, a maximum (weight) subgraph $G[F]$ of treewidth at most t, satisfying some property expressible in CMSO_2. It also handles annotated versions, where the vertices/edges of $G[F]$ must be selected from a prescribed set.

We have seen that Distance-d Independent Set can be solved in polynomial time on $\mathcal{G}_{\text{poly}}$ for any even d. This also holds for the more general problem of finding an induced subgraph $G[F]$ whose components are at pairwise distance at least d, and such that each component is isomorphic to a graph in a fixed family. E.g., each component could be an edge, to have a variant of Maximum Induced Matching where edges should be at pairwise distance at least d. For this we need to solve the corresponding problem on G^{d-1}, using only edges from G, as in [67].

When seeking for maximum (resp. minimum) induced subgraphs $G[F]$ of treewidth at most t such that $G-F$ is connected (resp. F dominates G) on particular subclasses of $\mathcal{G}_{\text{poly}}$, we can add any CMSO_2 condition on $G[F]$. It is not unlikely that the techniques can be extended to other classes than circular-arc graphs (and chordal graphs, for connectivity constraints).

An interesting open problem is the complexity of Connected Vertex Cover and Connected Feedback Vertex set in weakly chordal graphs, and on $\mathcal{G}_{\text{poly}}$.

Chapter 7

Conclusion and perspectives of Part I

This part of the thesis was devoted to the study of structural and algorithmic aspects of potential maximal cliques and minimal separators, specially focused on graphs with a *few* minimal separators and parameterized complexity.

We succeed to describe new relations between structural parameters of graphs and their number of minimal separators and potential maximal cliques. We showed that for any graph the number of minimal separators and potential maximal cliques is upper bounded by a single exponential function its vertex cover number or a single exponential function of its the modular width. We also showed that odd powers of a graph G have at most the number of minimal separators of G.

These new structural properties, pipelined with the algorithm of Fomin *et al.* (Proposition 22), imply new complexity results for a large family of problems. The bound on the number of minimal separators of a graph with respect its vertex cover number (respectively its modular width) implies FPT algorithms with single-exponential function of the parameter for Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ parameterized by the vertex cover number (respectively the modular width of the input graph). The result involving odd powers of minimal separators implies polynomial time algorithms for problems like d-independent set, for even d.

There are several open questions which may motivate further research, the following where already posed (though not stated in the same way) by Fomin *et al.* in [67]. It is still open which kind of properties can we ask to *the rest of the graph* when we look for a maximum induced subgraph of bounded treewidth. We proved in Chapter 6 that a set F such that $G - F$ is connected and $G[F]$ is of treewidth at most t, can be computed for graph classes like chordal graphs and circular-arc graphs. An interesting further research is whether this problem is polynomial time solvable in graph classes with polynomially many minimal separators. A good starting point would be to try to solve Connected Vertex Cover is the class of weakly-chordal graphs.

The running time of the algorithm Max Induced Subgraph of tw $\leq t$ satisfying $\mathcal{P}$ given in Proposition 22 is $\mathcal{O}(|\Pi_G| \cdot f(\mathcal{P}, t) \cdot n^{t+3})$. We may ask is whether this can be improved, maybe for some particular property $\mathcal{P}$, into $\mathcal{O}(|\Pi_G| \cdot g(\mathcal{P}, t) \cdot n^{\mathcal{O}(1)})$, i.e., an FPT algorithm parameterized by t. A more concrete question is whether problem Clique parameterized by the size of the solution is FPT in graphs with polynomially many minimal separators. We remark that for every structured class of graphs with few minimal separators presented in this thesis (weakly-chordal, polygon-circle, d-trapezoid, and their subclasses) problem Clique is polynomial time solvable. However, Clique is NP-Hard restricted to classes of graphs with a linear number of minimal separators [131]. It is unknown if Clique parameterized by the size of the solution is FPT in graphs with polynomially many minimal separators.

Note that all the algorithms presented in this part of the thesis utilize all the minimal separators and potential maximal cliques. Recently Lokshtanov, Vatshelle and Villanger [104] and Lokshtanov, Pilipczuk and van Leeuwen [106] proposed polynomial and quasi-polynomial time algorithms for solving independent set in P_5-free and P_6-free graphs. Their protocols are based on potential maximal cliques (among other techniques and structures), however P_5-free and P_6-free graphs may contain exponentially many minimal separators. The idea of designing algorithms that use some (not all) minimal separators and potential maximal cliques is not new. Broersma *et al.* showed in [36] that Independent Set can be solved in AT-Free graphs (which is also a class that does not have polynomially many minimal separators) carefully selecting relevant minimal separators (the ones that are in the neighborhood of a

vertex). A future research may explore the combination of the structural results presented in this thesis with algorithms that use some (and not all) minimal separators and potential maximal cliques.

Finally, we believe that the interplay between graphs of bounded MIM-width [135] and $\mathcal{G}_{\text{poly}}$ deserves to be studied. None of the classes contains the other, but several natural graph classes are in their intersection, and they are both somehow related to induced matchings.

Part II

A study of the Broadcast Congested Clique model

Chapter 8

Description of the model and tools

In this chapter, we will give a formal definition of the distributed computing model that we will consider in this thesis: the Broadcast Congested Clique. Later, we will survey the main techniques that are going to be used in the next chapters.

8.1 The Broadcast Congested Clique model

The *Broadcast Congested Clique* model is a message-passing model of distributed computation where a set of nodes communicate with each other in synchronous rounds over a *complete network* [1, 2, 4, 13, 14, 15, 57, 79, 85, 87].

Let $\mathcal{G}$ be the set of all undirected graphs of size n, let Y be an arbitrary set, and let $f : \mathcal{G} \to Y$ be an arbitrary function. In the Broadcast Congested Clique model, that we denote BCLIQUE, there is a set of n players, called *nodes*, which are numbered with unique identifiers in $[n]$. The nodes wish to compute $f(G)$ for some input graph $G \in \mathcal{G}$. The difficulty is that only node i knows the i-th row of the adjacency matrix of G (which is called the *input of node* i). In order to evaluate function f, the nodes communicate according to a fixed *protocol* which depends only on f. A protocol consists in a certain number of *communication rounds* (or simply *rounds*) where each node simultaneously transmit a single message visible to every other node. We say that nodes *broadcast* their messages. At each communication round, the nodes must decide which message they will broadcast based only in their inputs and all the messages received so far. The protocol finalizes when the value of $f(G)$ can be determined by every node, so every node *terminates*. The maximum size of a message sent by a node is called the *bandwidth* of the protocol, and the number of rounds needed until every node terminates is called the *time* of a protocol. The product of the bandwidth and the time is called *cost* of a protocol.

This model can be seen a multi-player generalization of the model of *simultaneous messages* in communication complexity. In such a model, there are two players, Alice and Bob, which communicate in simultaneous rounds in order to compute some function. To generalize this to several players, there exist in the literature (e.g, [97]) two main schemes. In the NUMBER-IN-HAND model, each player has its own private input, not known to other players. This model has some interest, though its often more restricted two the two players case. The other extreme is the NUMBER-ON-THE-FOREHEAD model, where each player knows everyone's input, except it own. In this presentation, we will focus only in *graph problems*, which lies somehow in between of the NUMBER-ON-THE-FOREHEAD and the NUMBER-IN-HAND models. In this scenario, the joint input to the n nodes is an undirected graph G whose set of vertices coincides with the set of nodes. Note that the adjacency matrix of a graph is symmetric, so every bit of the input (edge of the graph) is known by two players.

Note that we do not bound the computational resources that the nodes require to compute the messages. Since the input of each node belongs to $\{0,1\}^n$, any node can *reconstruct* the whole input graph simply communicating x_i. Therefore, we will be interested only in protocols with sub-linear cost, i.e., where the product of the number of rounds R and the bandwidth b satisfies $Rb = o(n)$.

For many natural functions, the cost of a protocol computing it can be drastically reduced when randomization is allowed. A *deterministic protocol* for a function f is a protocol where the algorithm performed by the nodes have no access to any randomness and correctly evaluate function f on any input. An ϵ-*error randomized protocol* is a protocol where the nodes receive, together with the corresponding

inputs, a random Boolean vector, which they may use to produce their messages. In this case, we do not ask that the protocol outputs the correct answer for every input. Instead we ask that for every input the protocol gives the correct answer with probability at least $1 - \epsilon$. We also say that a protocol gives the correct answer *with high probability* if the probability of error smaller than $1/n^c$, where $c > 0$ and n is the number of nodes (or the size of the input).

We distinguish two types of randomized protocols: the *public coins protocols* and the *private coins protocols*. In the public coins case, the random Boolean vector (that can be viewed as the result a certain number of *coin* tosses) is of *public knowledge*, which means that every node knows it. In other words, every node receives the same random vector. In private coins protocol, the random vector received to each node is *private*, so every node receives a potentially different random vector, and they have no knowledge about the ones received by other nodes. We will discuss the power of deterministic, public and private coins protocols in Chapter 9. When we do not specify, a *randomized protocol* will be a public-coin protocol.

The *deterministic cost* of a function f, denoted $C^{\text{det}}(f)$, is the minimum cost of any deterministic protocol in the BCLIQUE model computing f. Analogously, we denote $C_\epsilon^{\text{priv}}(f)$, $C_\epsilon^{\text{pub}}(f)$, as the costs for ϵ-error public and private protocols, respectively.

Sometimes we consider protocols with a fixed number of rounds. In a such case the cost of the protocol equals its bandwidth. Moreover, we are interested in one-round protocols, which can be characterized in a slightly different but equivalent way. We assume that there is a global memory, represented in a *whiteboard*, where every node can write a single message of the size of the bandwidth. Then, we consider another player called the *referee*. This player has no access to the input of the problem. Instead it reads the whiteboard, i.e., receives the messages produced by each node in the first (and unique) communication round, and writes on the whiteboard the evaluation of f in the corresponding input.

We emphasize that the model with the whiteboard and the referee is equivalent (up to one bit) to the one-round BCLIQUE model. Indeed, suppose that there is a one-round protocol $\mathcal{P}$ in the BCLIQUE model computing a function f. As we said in the definition of the broadcast congested clique model, after the first round every node i must be able to evaluate $f(x_1, \ldots, x_n)$ only knowing x_i and the messages sent in the communication round. We simulate this with a referee with no access to any input as follows. Consider the protocol $\mathcal{P}'$, where each node i writes on the whiteboard the message that it would send in protocol $\mathcal{P}$, together with $x_{i,1}$, i.e., the first coordinate of x_i. Since the input is a graph, necessarily the first coordinate of the inputs of the n nodes must be equal to the input of node 1, in other words $x_1 = (x_{1,1}, \ldots, x_{n,1})$. Then, the referee can use the n messages on the whiteboard to simulate the behavior of node 1 in protocol $\mathcal{P}$. Note that this equivalence may be not possible if the joint input of the n nodes is not a graph.

In [14] Becker *et al.* proposed the first study of one-round protocols Broadcast Congested Clique. The authors showed that any d-degenerate graph can be reconstructed by a one-round deterministic protocol in the BCLIQUE model with cost $\mathcal{O}(d^2 \cdot \log n)$.

Proposition 31 ([14]). *There is a one-round deterministic protocol in the* BCLIQUE *model with cost* $\mathcal{O}(d^2 \log n)$ *which reconstructs and recognize d-degenerate graphs.*

The family of graphs of bounded degeneracy contains many relevant classes of graphs, including graphs of bounded tree-width, planar graphs, and graphs that not contain a particular graph as a minor. Another result involving recognition of classes of graphs is due to Kari et. al [87], where the authors show that the class of cographs can be recognized by a one-round public-coin protocol in the BCLIQUE model with high probabilty and cost $\mathcal{O}(\log n)$.

Proposition 32 ([87]). *There is a one-round public-coins protocol in the* BCLIQUE *model with cost* $\mathcal{O}(\log n)$ *which with high probability determines if the input graph is a cograph, and if it is, the referee reconstructs the graph.*

In [14], the authors also give some lower bounds for one-round deterministic protocols, i.e., problems that require each node to communicate $\Omega(n)$ bits in any one-round deterministic protocol. Some examples of hard functions are the problem consisting in computing the diameter of the input graph, of determining if the graph contains a triangle (cycle of length three). Other examples of lower bounds for one-round protocols appear in [87], where is studied the problem of recognizing whether a fixed graph H appears in the input graph as a subgraph or an induced subgraph. The authors show that for any non-bipartite

graph H, any one-round ϵ-error public-coin protocol recognizing graphs where H is a subgraph (or an induced subgraph) has cost $\Omega(n)$.

Druker *et al.* [57] were the first giving lower bounds in the Broadcast Congested Clique model, without restricting to one-round protocols. The authors showed that for any fixed $l \geq 4$, any ϵ-error public coin protocol recognizing graphs containing a clique K_l as a subgraph has cost $\Omega(n)$, recognizing graphs containing a cycle C_l as a subgraph has cost $\Omega(n)$ if l is odd and $\Omega(n^{2/l})$ if l is even, and for $l, m \geq 1$ any protocol recognizing graphs containing a biclique $K_{l,m}$ has cost $\Omega(\sqrt{n})$. The authors also give a lower bound in the detection of triangles, using a reduction from the disjointness problem in the three-players NUMBER-ON-THE-FOREHEAD model in communication complexity (see Section 8.2.1 for more details).

One way to increase the computing power of the model is to lift the broadcast restriction and to allow the nodes the possibility of sending *different* messages through different links. This model, that we will call *unicast congested clique* [57], is often called the *congested clique* model in the literature, because it corresponds to the $\mathcal{CONGEST}$ model when the communication network is a clique. Sending different messages through different communication channels gives the possibility to perform a load balancing procedure efficiently. Such enormous intrinsic power has allowed some authors to provide fast protocols for solving natural problems: a $\mathcal{O}(\log\log\log n)$-round protocol for finding a 3-ruling set [83], $\mathcal{O}(n^{0.158})$-round protocols for counting triangles, for counting 4-cycles and for computing the girth [38], a $\mathcal{O}(1)$-round protocol for detecting a 4-cycle [38], a $\mathcal{O}(\log^* n)$-round protocol for constructing a minimum spanning tree [76]. We denote UCLIQUE the *Unicast Congested Clique*.

The *unicast* and *broadcast* congested clique models in graphs are equivalent when they are restricted to one-round protocols. The argument is similar to the equivalence between one round protocols in the BCLIQUE model and the model defined with the whiteboard and the referee. Indeed, if $\mathcal{P}$ is a protocol in the UCLIQUE model, then all nodes must accept in one round, each one of them sending potentially a different message to every other player. Consider now the following protocol $\mathcal{P}'$ in the BCLIQUE model defined from $\mathcal{P}$ as follows: each node i writes the message that it would send to node 1 in protocol $\mathcal{P}$, together with bit $x_{i,1}$. The referee then reconstructs the input of player 1 and collects all the messages that node 1 would receive in protocol $\mathcal{P}$, and outputs the answer that node 1 would output in protocol $\mathcal{P}$. We reemphasize that this equivalence may be not possible if the joint input of the n nodes is not a graph.

In [12] the authors show that the space between the unicast and broadcast congested clique models is very rich and interesting. For instance, they show the existence of a problem that takes $\Omega(n/\log n)$ rounds in the broadcast model while it can be solved in two rounds if only *two different messages* can be sent by each node.

Another very natural, much more limited and less dramatic way to increase the computing power of the broadcast congested clique model, *is to expand the local knowledge the nodes initially have about* G. The idea of a constant-radius neighborhood independent of the size of the network is present in the research on local algorithms pioneered by Angluin [3], Linial [103] and Naor and Stockmeyer [114]. We call BCLIQUE_r the extension of the broadcast congested clique model where each node u sees (receives as input) the set of all edges lying on a path of length at most r, starting in u, and each message is of size at most b. Note that BCLIQUE_1 corresponds to the classical BCLIQUE model.

8.1.1 Problems

We present here some of the problems that will appear in the next chapters.

The first function is RECONSTRUCTION(G), whose output is G itself, i.e., the adjacency matrix of G. Note that if a protocol computes RECONSTRUCTION, then cost of the protocol must be $\Omega(n)$ [14]. Indeed, there are $2^{\binom{n}{2}}$ labeled graphs of size n, which can be coded in $\binom{n}{2}$ bits. Since any protocol with cost C communicates nC bits, we obtain that $C = \Omega(n)$.

For a class of graphs $\mathcal{C}$, the function RECOGNITION OF $\mathcal{C}$ on an input graph G outputs 1 if and only if G belongs to the class $\mathcal{C}$. We also define RECONSTRUCTION ON $\mathcal{C}$, which is the same problem than RECONSTRUCTION, but with the promise that the input graph belongs to $\mathcal{C}$.

The problem TRIANGLE(G) equals 1 if and only if G contain a triangle and 0 otherwise. More in general, we consider functions INDUCED-CYCLE$_{=k}$, INDUCED-CYCLE$_{\leq k}$, INDUCED-CYCLE$_{>k}$, which output 1 if and only if, respectively, the input graph has an induced cycle of length exactly k, at most k, strictly larger than k. Problems CYCLE$_{=k}$, CYCLE$_{\leq k}$, CYCLE$_{>k}$ are defined in a similar way, but here

we ask whether the input graph has a cycle (induced or not) of length k, at most k, strictly larger than k. The function $\textsc{Diam}_k(G)$, for $k > 0$ asks if the input graph G has diameter at most k.

We call $\textsc{Connectivity}(G)$ to the function that outputs 1 if and only if G is connected. Similarly, Connected Components (respectively Maximum Spanning tree) is the problem consisting on computing the connected components (respectively a maximum spanning tree) of the input graph G. At the end of the protocol each node must know the connected components (respectively the same spanning tree).

The problem $\textsc{Bipartiteness}(G)$ is the function that outputs 1 if and only if G is bipartite. We point a remark made in [1], where the authors reduce the problem Bipartiteness to the problem of counting the number of connected components. Indeed, for a graph G of size n call $D(G)$ the graph of size $2n$, where for each vertex v of G, we construct verices v_1, v_2 in $D(G)$, and for each edge $\{u, v\}$ of G, we create edges $\{u, v_2\}$ and $\{u_2, v_1\}$ in G.

Proposition 33 ([1])**.** *Let k be the number of connected components in G. Then $D(G)$ has $2k$ connected components if and only if G is bipartite.*

Proof. The proof of this proposition appears in [1]. Let $G_1, \ldots, G_k$ be the connected components of G. By construction of $D(G)$, each $D(G_i)$ is disconnected of $D(G_j)$ for $i \neq j$. Then, is suffices to count the number of connected components of $D(G)$, for G connected.

Suppose that there exists an odd cycle in G which contains a vertex u. The odd cycle corresponds to a u_1, u_2-path in $D(G)$. Each path from a u to v in G corresponds to a path from u_1 to w and a path from u_2 to w' in $D(G)$, where $w \in \{v_1, v_2\}$ and $w' \in \{v_1, v_2\} \setminus \{w\}$. Since G is connected, there exists a path from u to any v in $V(G)$. Therefore, there is a path from u_1 to any vertex of $D(G)$, so $D(G)$ is connected. On the other hand, if G is connected then $D(G)$ is not connected because any path from u_1 to u_2 in $D(G)$ corresponds to a odd cycle in G. □

8.2 Tools

In this section we give an overview of the main tools used in this part of the thesis: *Communication complexity lower bounds*, *fingerprints*, *linear sketches*, *deterministic sparse linear sketches*, and finally *error correcting codes*.

8.2.1 Communication Complexity lower bounds

For many functions, we are going to be able to separate the information in roughly two (or a constant) independent parts, even if every bit of the input is shared by two nodes. In such cases, the lowerbounds of communication complexity translate to the broadcast congested clique. The *simultaneous messages* communication model is simply the Broadcast Congested Clique model with two nodes, which we name Alice and Bob. Note that we do not ask in this case that the join input of the two players is a graph, but any pair of n-bit input. All definitions are the same, including the definitions of randomized and deterministic protocols.

Newmann [117] showed that in the two players case private and public coin multi-round randomized protocols are equivalent, up to a logarithmic factor in the message size complexity. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function, and call $R_\epsilon^{\text{priv}}(f)$ (respectively $R_\epsilon^{\text{pub}}(f)$) the product of the bandwidth times number of rounds of the best ϵ-error private coins (resp. public coins) randomized protocol computing f.

Proposition 34 (Newman [117], see also Kushilevitz and Nissan [97])**.** *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function. For every $\delta > 0$ and every $\epsilon > 0$, $R_{\epsilon+\delta}^{priv}(f) \leq R_\epsilon^{pub}(f) + \mathcal{O}(\log n + \log(\delta^{-1}))$.*

In Chapter 9 we will show that this is true in general for any number of players.

Clear separations have been proved between one-round deterministic, private coins and public coins protocols. For instance, consider the EQ_n function, which simply tests whether the n-bit inputs of Alice and Bob are equal. I can be shown, e.g., using the *fooling set* technique [97], that for any deterministic protocol the bandwidth b times the number of rounds r satisfy $rb = \Theta(n)$ [97]. Using the *fingerprint* technique (see next subsection) it was shown that a bandwidth of $\mathcal{O}(1)$ is enough for one round randomized protocols with public coins with constant one-sided error [9], and $\Theta(\sqrt{n})$ for randomized protocols with private coins and constant one-sided error [9].

Proposition 35 ([9]). *Let $\epsilon > 0$. The function EQ_n has the following costs: $C^{\det}(\mathrm{EQ}) = n$, $C^{\mathrm{priv}}_{\epsilon}(\mathrm{EQ}_n) = \Theta(\sqrt{n})$ and $C^{\mathrm{pub}}_{\epsilon}(\mathrm{EQ}_n) = \mathcal{O}(1)$.*

More generally, Babai and Kimmel [9] proved that for any function f its randomized message size complexity, for private coins protocols, is at least the square root of its deterministic message size complexity.

Proposition 36 ([9]). *Consider the number-in-hand model with two players and a constant $\epsilon > 0$. For any two input Boolean function f, $C^{\mathrm{priv}}_{\epsilon}(f) = \Omega(\sqrt{C^{\det}(f)})$.*

Chakrabarti *et al.* [39] proved that, for some family of functions, the gap between deterministic and randomized message size complexity with private coins is smaller than the square root. They consider the function OREQ_n that takes as input two Boolean $n \times n$ matrices, and the output is 1 if and only if there is some $1 \leq i \leq n$ such that the i-th lines of the two matrices are equal.

Proposition 37 ([39]). *For any $\epsilon < 1/3$, $C^{\mathrm{priv}}_{\epsilon}(\mathrm{OREQ}_n) = \Omega(n\sqrt{n})$. Also, $C^{\det}(\mathrm{OREQ}_n) = \Theta(n^2)$.*

Naturally, there are functions which are hard for deterministic and randomized protocols. For one-round protocols a *canonical* example is the function $\textsc{Index}_n$, where Alice receives a n-bit input x and Bob receives an index $i \in [n]$, and the output is the value of x_i. Evidently this function can be computed in two rounds and bandwidth $\mathcal{O}(\log n)$: in the first round Bob sends i ($\mathcal{O}(\log n)$ bits) and in the second sound Alice communicates x_i (one bit). However, restricted to one round protocols this function is hard, even in the randomized setting. Intuitively, Alice has no way to guess which coordinate will be asked by Bob, so she has no option but communicate all the coordinates of her input x.

Proposition 38 ([97]). *For any $\epsilon < 1/3$, $C^{\det}(\textsc{Index}_n) = C^{\mathrm{pub}}_{\epsilon}(\textsc{Index}_n) = \Omega(n)$.*

There are also examples of hard functions for randomized multiround protocols. Consider the function DISJ_n tests whether the n-bit inputs of Alice and Bob are disjoint. In other words, if we call x and y the inputs of Alice and Bob respectively, $\mathrm{DISJ}_n(x, y) = 1$ if there is a coordinate $i \in [n]$ such that $x_i = y_i = 1$. It can be shown, using techniques from *information complexity*, that any r round protocol (deterministic or randomized) of bandwidth b satisfy $rb = \Theta(n)$ [97].

Proposition 39 ([97]). *For any $\epsilon < 1/3$, $C^{\det}(\mathrm{DISJ}_n) = C^{\mathrm{pub}}_{\epsilon}(\mathrm{DISJ}_n) = \Omega(n)$.*

In Chapter 11 we will give a reduction, based on a result of Druker *et al.* [57] that uses the disjointness problem DISJ_n in the three-player Number-on-the-forehead model. Consider three players Alice, Bob and Charlie, each one has a subset X_A (resp. X_B, X_C) of $\{1, \ldots m\}$, and each player sees the vector of the two others. Their common goal is to decide whether the three sets have a non-empty intersection. It is known that this problem requires a communication of $\Omega(n)$ in the deterministic case, and $\Omega(\sqrt{m})$ for any ϵ-error probabilistic protocol with $\epsilon < 1/3$. Moreover, Pătraşcu and Williams [121] showed that the linear lower bound extends to the probabilistic case under the Strong Exponential Time Hypothesis (stating that, for any $\delta < 1$, there is no algorithm solving n-variable instances of Sat in time $\mathcal{O}(2^{\delta n})$, see Section 2.4).

Proposition 40 (Pătraşcu and Williams [121]). *Let $\epsilon < 1/3$. An ϵ-error randomized protocol for DISJ_m in the three-player Number-on-the-forehead model with communication complexity $o(m)$, and local $2^{o(m)}$ running time, implies that the Strong Exponential time hypothesis fails.*

In other words, under the Strong Exponential Time hypothesis and the assumption that local computations must be performed in time $2^{o(n)}$, there is no nontrivial randomized algorithm for disjointness in the 3-party Number-on-the-forehead model communicating only $o(n)$ bits.

8.2.2 Fingerprints

The following technique, that we call *fingerprints*, is based on a result known as the Schwartz–Zippel Lemma, used in verification of polynomial identities. Let n be a positive integer and p be a prime number. In the following, we denote by $\mathbb{F}_p$ the finite field of size p (we refer to the book of Lidl and Niederreiter [101] for further details and definitions around finite fields). A polynomial $P \in \mathbb{F}_p[X]$ of *degree* d is an expression of the form $P(x) = \sum_{i=0}^{d} a_i x^i$, where $a_i \in \mathbb{F}_p$ and $a_i \neq 0$ for each $i \in d$. We denote by $\mathbb{F}_p[X]$ the polynomial ring on $\mathbb{F}_p$. An element $b \in \mathbb{F}_p$ is called a *root* of a polynomial $P \in \mathbb{F}_p[X]$ if $P(b) = 0$.

Proposition 41. *[101] A polynomial of degree d in $\mathbb{F}_p[X]$ has at most d roots in $\mathbb{F}_p$.*

Let n be a positive integer, p and q be two prime numbers such that $q < p$ and $p > n$. For each $a \in \mathbb{F}_q^n$ and $t \in \mathbb{F}_p$, consider the polynomial in $P(a, \cdot) \in \mathbb{F}_p[X]$ defined as

$$P(a,t) = \sum_{i \in [n]} a_i t^{i-1}.$$

For $t \in \mathbb{F}_p$, we call $P(a,t)$ the *fingerprint* of a and t. Note in the last expression that the coordinates of a are interpreted as elements of $\mathbb{F}_p$.

Lemma 19. *Let n be a positive integer, p and q be two prime numbers such that $q < p$ and $p > n$, and let $a, b \in \mathbb{F}_q^n$ be such that $a \neq b$. Then*

$$|\{t \in \mathbb{F}_p : P(a,t) = P(b,t)\}| \leq n.$$

Proof. Note that $P(a,t) = P(b,t)$ implies that $P(a-b,t) = P(a,t) - P(b,t) = 0$. Since $P(a-b,t)$ is a polynomial of degree at most n in $\mathbb{F}_p[X]$, from Proposition 41 it has at most n roots in $\mathbb{F}_p$. Therefore $|\{t \in \mathbb{F}_p : P(a,t) = P(b,t)\}| \leq n$. □

Recall the problem EQ_n, where two players Alice and Bob receive each an n-bit vector, called a and b respectively, and the question is to determine if $a = b$. Consider the following one-round public coins randomized protocol for solving EQ_n [97]. First, fix a prime number p in $(n^{c+1}, 2n^{c+1})$, for some constant $c > 0$ (p exists thanks to the Prime Number Theorem). Then using the public coins, Alice and Bob pick (the same) $t \in \mathbb{F}_p$ uniformly at random. Alice sends to the referee $P(a,t)$ and Bob sends $P(b,t)$. Note that both messages are in $\mathbb{F}_p$, so they can be encoded in $\mathcal{O}(\log p) = \mathcal{O}(\log n)$ bits. The referee then checks if the messages of Alice equals the message of Bob.

Let us show that the referee fails with a very low probability. Clearly two equal vectors have equal fingerprints for every $t \in \mathbb{F}_p$, i.e., if $P(a,t) \neq P(b,t)$ then $a \neq b$, so the referee does not obtain false negatives. More important, if $a \neq b$, the probability that $P(a,t) = P(b,t)$ is at most $1/n^c$. Indeed, because t was chosen uniformly at random, and from Lemma 19, we obtain that the probability that t is a root of $P_a - P_b$ is at most $n/p < 1/n^c$.

8.2.3 Linear sketches and graph connectivity

Connectivity and maximum spanning tree (MST) are one of the most relevant and foundational problems in the Congested Clique model. In fact, the Unicast version of the model was defined in [108, 107] where the authors studied the maximum spanning tree in the $\mathcal{CONGEST}$ model, when the communication graph has constant diameter. The authors showed that a MST can be computed in the deterministic UClique model in $\mathcal{O}(\log \log n)$ rounds and bandwidth $\mathcal{O}(\log n)$. More recently, Ghaffari *et al.* [76] showed that a MST can be computed by a randomized protocol in the UClique model in $\mathcal{O}(\log^* n)$ rounds with bandwidth $\mathcal{O}(\log n)$.

These two protocols are based on a quite simple algorithm published in 1926 by Borůvka [115]. Let G be a graph, and denote by $\hat{V}$ the set of *supernodes*, which initially are the n singletons $\{\{u\} | u \in V\}$. The protocol runs int $t = \lceil \log n \rceil$ phases. At phase $0 \leq i < t$, each supernode picks an incident edge to each set $\hat{v} \in \hat{V}$. Then, we merge the obtained connected components of supernodes into a single supernode. The procedure finishes when all supernodes are isolated, in which case the supernodes represent the connected components of G. The procedure finishes before $t = \lceil \log n \rceil$ steps since the number of supernodes at least halves at each step.

Borůvka's algorithm directly implies a $\mathcal{O}(\log n)$ round deterministic protocol for computing a MST in the deterministic BClique model with bandwidth $\mathcal{O}(\log n)$. Despite its simplicity, to our knowledge there is no randomized or deterministic protocol in the Broadcast Congested Clique that computes the connected components of the input graph in a number of rounds r and bandwidth b that satisfy $rb = o(\log^2 n)$. Moreover, no nontrivial lower-bounds exist, even in the one-round case.

In their SODA 2012 paper [1], Ahn, Guha and McGregor provided the first one-round (randomized) protocol for graph connectivity in the Broadcast Congested Clique, with polylogarithmic bandwidth. To be precise, their protocol was designed for the *dynamic graph streams* model, however their techniques can be directly applied to our model.

Their algorithm is based on a extremely beautiful technique called *linear sketches*. This technique uses randomization to compress any vector into a another with an exponential reduction in the coordinates. This compression is such that one can recover, with high probability, a nonzero component of the original vector with high probability. Moreover, the compression operation is a linear function, therefore adding to compressed vectors we obtain the compression of the sum of the two vectors. Formally,

Definition 5. *Let $n, k, \delta > 0$. A δ-linear sketch of size k is a function $S : \{0,1\}^{\mathcal{O}(\log n)} \times \{-1,0,1\}^n \to \{0,1\}^k$, such that, if we call $S_r = S(r, \cdot)$, then*

- *S_r is linear, for each $r \in \{0,1\}^{\mathcal{O}(\log n)}$;*
- *If r is chosen uniformly at random, then there is an algorithm that on input $S_r(x)$ returns* ERROR *with probability at most δ, and otherwise returns a pair (i, x_i) such that $x_i \neq 0$ and coordinate i is picked uniformly at random between the non-zero coordinates of x. The probabilities are taken over the random choices of r.*

In [86] it is shown a construction of δ-linear sketches that compress input vectors of size n into vectors with polylog (n) coordinates. Their construction is performed in two steps: the first step is a *sampling* procedure, the input vector is sampled $\log(n)$ times, where the i-th time each coordinate is picked with probability 2^{-i}. This way we obtain that one of the samples is roughly $\mathcal{O}(\log n)$-sparse (has at most that number of nonzero coordinates) with high probability. The second step is a *reduction* procedure, where using $\mathcal{O}(\log n)$-wise independent hashing functions of range $\mathcal{O}(n^3)$, each sample of the input vector is reduced into a vector of dimension $\mathcal{O}(\log n)$. The reduction procedure satisfies that if the sample is $\mathcal{O}(\log n)$-sparse then each nonzero coordinate is hashed into a different value with high probability.

Proposition 42 ([86])**.** *For each $n, \delta > 0$, there exists a δ-linear sketch of size $\mathcal{O}(\log^2 n \log \delta^{-1})$.*

In the CONNECTIVITY protocol of [1], each node constructs $\mathcal{O}(\log n)$ linear sketches based on its neighborhood and on a sequence of public random coins, and broadcasts them to all other nodes. Using all these messages, the referee can simulate the phases of Borůvka's algorithm by (1) finding an outgoing edge of each *supernode* (2) summing the linear sketches of the clusters to obtain the linear sketches of the union of adjacent supernodes. The crucial point is that everything is done in one round of communication.

Proposition 43 (Ahn *et al.* [1])**.** *Let $n, \delta > 0$ and $t = \lceil \log n \rceil$. There exists an algorithm that receives t independent δ-connectivity sketches of a graph G, produced with $r_1, \ldots, r_t \in \{0,1\}^{\mathcal{O}(\log n)}$ random strings picked uniformly at random, and outputs a spanning forest of G with probability $1 - \delta$.*

Let $G = (V, E)$ be a graph of size n, and $x \in V$. We call *connectivity vector of x in G* the vector of dimension $\binom{n}{2}$ defined as follows:

$$a^x_{\{u,v\}} = \begin{cases} 1 & \text{if } \{u,v\} \in E, x = u \text{ and } u < v, \\ -1 & \text{if } \{u,v\} \in E, x = v \text{ and } u < v, \\ 0 & \text{otherwise.} \end{cases}$$

For $r \in \{0,1\}^{\mathcal{O}(\log n)}$, we say that $S_r(G) = \{S_r(a^x)\}_{x \in V(G)}$ is a δ-*connectivity sketch* of G, where S is a δ-linear sketch. Note that for any $x \in V$, each non-zero coordinate of a^x represents an edge of $N(x)$, and for any $U \subseteq V$ the non zero coordinates of $\sum_{x \in U} a_x$ are exactly the edges in the cut between U and its complement $V \setminus U$.

Let $G = (V, E)$ be the input graph. The one-round protocol in the BCLIQUE model devised by Ahn *et al.* for computing a spanning forest of G with cost $\mathcal{O}(\log^3 n)$works as follows. Let $t = \lceil \log n \rceil$. Each node computes and sends t independent δ-linear sketches of its connectivity vector, using t random strings $r_1, \ldots, r_t$ picked uniformly at random. Using these messages, any node can compute t independent δ-connectivity sketches of G and therefore, it can compute a spanning tree using the following t steps procedure. First, as in Borůvka's algorithm, let us denote by $\hat{V}$ the set of *supernodes*, which initially are the n singletons $\{\{u\} | u \in V\}$. At step $0 \leq i < t$, each node samples an incident edge to each set $\hat{v} \in \hat{V}$ using the ith collection of linear sketches $\sum_{x \in \hat{v}} S_{r_i}(a^x)$, and merge the obtained connected components into a single supernode.

8.2.4 Deterministic sparse linear sketches

The following technique is a *deterministic* version of linear sketches. The idea is to design a function that compress a vector into a smaller one via a linear function, with the promise that we can recover some information of the original vector if we only know the compression. This task might be impossible to accomplish if we want to deterministically compress any vector into a much smaller one (e.g., of sub-linear size). Indeed, if such function existed, we might solve the Equality problem in communication complexity with sub-linear communication cost. Instead, we will build deterministic *sparse* linear sketches.

Let d be a positive integer. A n-dimensional Boolean vector x is called *d-sparse* if the sum of its coordinates is at most d. We will show a technique that permits to compress any n-dimensional Boolean vector via a linear function ℓ. This function has the property of being injective when restricted to d-sparse Boolean vectors. For example, if $x, y \in \{0,1\}^n$ are such that $x - y$ is d-sparse, then we can recover $x - y$ knowing $\ell(x)$ and $\ell(y)$.

Note that, in contrast with the setting of (randomized) linear sketches, the following lemma *does not use randomization*. In fact, our technique is a form of derandomization of the *fingerprint technique*.

Lemma 20. *Let $n, d > 0$. There exists a function $\ell : \mathbb{Z}^n \to \mathbb{F}_p$, for some prime number $p = 2^{\mathcal{O}(d \cdot \log n)}$, such that:*

- *ℓ is linear, and*
- *ℓ is injective when restricted to d-sparse Boolean inputs.*

Proof. Let $\mathcal{B} = \{b \in \{0,1\}^n : \sum_{i=1}^n b_i \leq d\}$ be the family of d-sparse Boolean vectors of dimension n, and call $\mathcal{H} = \{h \in \{-1,0,1\}^n : \exists \text{ distinct } b, b' \in \mathcal{B}, h = b - b'\}$. Note that $|\mathcal{B}| \leq (1+n)^d$ and then $|\mathcal{H}| \leq (1+n)^{2d}$.

Call $p = p(n,d)$ the smallest prime number greater than $(1+n)^{2d} \cdot n$. From the Prime Number Theorem, we know that p lies in $[(1+n)^{2d} \cdot n, 2 \cdot (1+n)^{2d} \cdot n]$. Let now $P(\mathcal{H})$ be the family of polynomials over the field $\mathbb{F}_p$ associating to each $h \in \mathcal{T}$ the polynomial $P(h,x) = \sum_{i=1}^n h_i x^{i-1}$ (values are taken modulo p). In other words, $P(h,x)$ is the fingerprint of h and x.

Let $\overline{x} = \overline{x}(n,d)$ be the minimum integer in $\mathbb{F}_p$ which is not a root of any polynomial in $P(\mathcal{H})$. We claim that $\overline{x}$ exists because there are at most $|\mathcal{H}|$ polynomials in $P(\mathcal{H})$, each polynomial has at most n roots in $\mathbb{F}_p$, and $p > |\mathcal{H}| \cdot n$.

We define then, for each $v \in \mathbb{F}^n$, the function $\ell(v) = P(v, \overline{x})$. Clearly, ℓ is linear, and, by definition of $\overline{x}$, for any distinct $b, b' \in \mathcal{B}$, we have $\ell(b) = P(b, \overline{x}) \neq P(b', \overline{x}) = \ell(b')$. □

This technique will be used in Chapter 10 to improve the d-degenerate graph reconstruction of Proposition 31.

8.2.5 Error correcting codes

The last technique that we present in this chapter are the *error correcting codes*. These techniques were originally used to obtain reliable communication in a noisy communication channel. An error correcting code with parameters (n, m, d) is a mapping $C : \{0,1\}^n \to \{0,1\}^m$ such that for any $x, y \in \{0,1\}^n$, $x \neq y$ implies $d_H(C(x), C(y)) \geq d$, where $d_H(\cdot,\cdot)$ is the Hamming distance (the number of coordinates where two vectors differ). The idea of an error correcting code is that x is the unique n-dimensional Boolean vector that can produce a value in the ball of center $C(x)$ with radius d measured with the Hamming distance.

Consider the following error correcting code with parameters (n, m, d), where $m = n + d\lceil \log(2(n+d)) \rceil$, introduced by Reed and Solomon [123]. Let $n + d < q < 2(n+d)$ be a prime number, and fix a set $\{a_1, \ldots, a_{n+d}\} \in \mathbb{F}_q$, where $\mathbb{F}_q$ is the finite field of size q. Let p_x be the unique polynomial in $\mathbb{F}_q$ such that $p_x(a_i) = x_i$ for each $i \in [n]$. Consider now the function $C : \{0,1\}^n \to \mathbb{F}_q^m$ defined as $C(x) = (p_x(a_1), \ldots, p_x(a_{n+d}))$. Clearly if $x, y \in \{0,1\}^n$ such that $x \neq y$, then $d_H(C(x), C(y)) \geq d - 1$, since two different polynomials of degree n coincide in at most $n - 1$ points. Note that the first n coordinates of $C(x)$ equal x, and we can encode the other d coordinates as elements of $\{0,1\}^{\lceil \log(2(n+d)) \rceil}$. Therefore C, as a function with range $\{0,1\}^m$, is a (n, m, d) error correcting code. In such case, we say that C is a Reed-Solomon code with parameters n and d.

We will use error correcting codes in Chapter 10 to recognize and reconstruct small classes of graphs.

8.3 The role of node identifiers

In the definition of the Broadcast Congested Clique model, it is assumed that the nodes have unique identifiers in $[n]$, where n is the number of vertices on the graph. However, this might not be the case in some real-world applications, for example when the identifiers contain a prefix with the address of the geographical position of the nodes.

In case where the (unique) identifiers are not bounded by the number of nodes, it we would be still interesting to find protocols with costs which are sublinear in the size of the graph (that we still call n) and not necessarily in the maximum size of a node identifier (that we call *maxid*). Note that in that case when $maxid > n$, we look for protocols solving problems on graphs of size n labeled with numbers in $[maxid]$. The nodes receive as input the list of the identifiers of the nodes adjacent to it in the graph G. However, in this scenario the nodes do not know which are the identifiers of their nonadjacent vertices. Therefore, for a node the graph is of size *maxid*, where some vertices (at most n) are real players and the others are isolated vertices that cannot communicate. We assume that nodes know at least a linear bound in the size of the graph n, and all know the maximum identifier *maxid*.

We can easily relax the condition on the labels assuming that *maxid* is upper bounded by $\text{poly}(n)$. In such a case, it would be enough to add one more round of bandwidth $\mathcal{O}(\log n)$ to reduce any protocol to one where the identifiers are bounded by n: every node communicates its identifier (which can be encoded in $\lceil \log(\text{poly}(n)) \rceil = \mathcal{O}(\log n)$ bits), and once the nodes have received the identifiers of the others, they make an ordered list and relabel according to the obtained ordering.

We might relax even more the assumption on the identifiers if we admit public randomness. Indeed, suppose that the node identifiers are upper-bounded by 2^n. We can reduce this case (with high probability) to the case where the node identifiers are bounded by a polynomial poly, using the fingerprints technique. Fix $c > 1$ and a prime number in $p \in [n^{c+2}, 2n^{c+2}]$. Each node x takes its identifier $id(x)$ represented in binary, and compute a new identifier $id'(x)$ from the fingerprint of $id(v)$, i.e., $id'(x) = P(id(x), t)$, where $t \in [p]$ is picked uniformly at random using the public coins and $P(a, t) = \sum_i a_i t^{i-1} \mod p$. From Lemma 19, all the identifiers will be different with probability $1 - 1/n^c$ and bounded by $2n^{c+2}$.

Fortunately, the protocols and lower-bounds presented in this thesis do not require the relabeling round, and can be easily adapted to be valid in the case when the identifiers are bounded by $\text{poly}(n)$, for some polynomial poly. For simplicity, the results will be presented in the first definition of the Broadcast Congested Clique model, i.e identifiers are in $[n]$.

Chapter 9

Separation of deterministic and randomized models

The goal of this chapter is to separate, from the point of view of message size complexity (bandwidth), three different variants of the Broadcast Congested Clique model: deterministic protocols, randomized protocols with private coins and randomized protocols with public coins. For this purpose, we introduce variants of the Boolean function TWINS. This Boolean function returns 1 if and only if there are two nodes with the same neighborhood.

Most of the results of this chapter were published in the 21st International Colloquium, on Structural Information and Communication Complexity (SIROCCO 2016) and were obtained in collaboration with Florent Becker, Ivan Rapaport and Ioan Todinca [15].

9.1 Introduction

In this chapter, we study the power of deterministic, public coins and private coins randomized protocols in the BCLIQUE model. Our goal is to define problems which separate, from the point of view of the communication cost, the different models. Our strategy is to try to translate the results obtained in the simultaneous messages communication model, to the BCLIQUE model. In other words, the motivation of this chapter can be resumed as follows: does there exists a function f whose deterministic cost, private-coin cost and public-coin cost are $\Omega(n)$, $\Omega(\sqrt{n})$ and $\mathcal{O}(\log n)$, respectively? (like the equality function in the case of two players, see Proposition 35).

Very recently, Fischer *et al.* [61] showed that such a gap can be extended for any function to the multi-player simultaneous messages number-in-hand model. Recall that this model is equivalent to the Broadcast Congested Clique model, but the inputs of each player is completely independent (in particular not a graph). In [61] the authors showed an extension of Proposition 35 to the NUMBER-IN-HAND model, showing that the cost of any private coin protocol computing a function f is lower bounded by the deterministic cost of computing f.

To separate the three types of protocols, we tried to find an equivalent of the function EQ in the BCLIQUE mode. The natural candidate is the function TWINS(G), which outputs 1 if and only if G has two vertices u and v with the same neighborhood, i.e., such that $N(u) = N(v)$. We obtained that this function easily separates one-round deterministic and public-coin randomized protocols. However, showing lower bounds for this function and one-round private-coin protocols, or multiround deterministic protocols appears to be more challenging.

To cope with this lack of lower bounds, we defined variants of the function TWINS. First, we define a "pointed" version of previous TWINS function. Let $x \in [n]$, and consider the function TWINS$_x(G)$ which outputs 1 if and only if there is a vertex $y \neq x$ in a graph G of size n such that $N(y) = N(x)$. For this function, we showed that there is a private coins protocol that computes TWINS$_x(G)$ with high probability in the BCLIQUE$[\log n]$ model. This shows that in the BCLIQUE model we do not have in general a result analogous to Proposition 36.

In order to separate the private and public coins models for one-round protocols, we use a Boolean function called TRANSLATED-TWINS. We prove that the cost of this function in the private coins setting

is $\Omega(\sqrt{n})$, while it is $\mathcal{O}(\log n)$ for public coins protocols. The main results of this paper are summarized in Table 9.1.

	TWINS	TWIN$_x$	TRANSLATED-TWINS
Deterministic	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
Randomized private-coins	$\mathcal{O}(\sqrt{n}\log n)$	$\mathcal{O}(\log n)$	$\Omega(\sqrt{n})$, $\mathcal{O}(\sqrt{n}\log n)$
Randomized public-coins	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

Table 9.1 – Main results of this chapter.

With respect to multi-round protocols, we will show first that for multi-round randomized protocols, there is no distinction of private and public coin randomized protocols. Indeed, we extend Proposition 34 to show that any public coin randomized protocol in the BCLIQUE model can be simulated by a private coin protocol, slightly increasing the number of rounds, the bandwidth and the error probability. To separate multi-round deterministic and randomized protocols, we introduce the function GRAPHEQ(G) defined on graphs of size $2n$. This function which equals 1 if and only if the adjacency matrix of $G[\{1,\dots,n\}]$ is equal to the adjacency matrix of $G[\{n+1,\dots,2n\}]$, in other words if for every $i,j \in [n]$, vertex j belongs to $N_G(i)$ implies that vertex $j+n$ belongs to $N_G(i+n)$. We prove that the randomized cost of this function is $\mathcal{O}(\log n)$ (even for one-round protocols) while the deterministic cost is $\Omega(n)$.

There are several natural problems that cannot be solved with one-round randomized protocols with cost $o(n)$ bits. In the last part of this paper (Theorem 5) we sketch how the arguments of [14], for proving negative results on one-round deterministic protocols, can be extended to the randomized setting. More precisely, we prove that the randomized public coin cost of the Boolean functions TRIANGLE(G) (that outputs 1 if and only if G has a triangle) and DIAM$_3$(G) (that outputs 1 if and only if G has diameter at most 3) is $\Omega(n)$.

9.2 Equivalence of public and private coin protocols for multi round protocols

In this section we show that up to a logarithmic factor, private and public coin multi-round protocols are equivalent, in the BCLIQUE model with bandwidth at least $\mathcal{O}(\log n)$

Theorem 14. *Let $\epsilon, \delta > 0$ be such that $\epsilon < 1/3$, and let f be a function $f : \mathcal{G}_n \to \{0,1\}$, where $\mathcal{G}_n$ is the set of all graphs of size n. For every r-round ϵ-error public coin protocol computing f in the* BCLIQUE *model with bandwidth b, there exists a $r+1$-round $\epsilon+\delta$-error private coin protocol computing f in the* BCLIQUE *model with bandwidth $b + \log(n) + \log(\delta^{-1})$.*

To show this theorem we reason following the same arguments as Newman in [117] for the two-player case (see Proposition 34). Intuitively, to construct a private coins protocol from a public coins protocol, we may ask to some arbitrary node to privately generate the number of random bits required to run the public coin protocol, and communicate those bits to the other nodes in a first round. However, in our definition of randomized protocols, we did not bound the number of random bits that a protocol is able to use, so this first round may require a very big bandwidth (or too many rounds). In the next lemma, we show that by slightly increasing the error, we can transform any public coin randomized protocol that uses an arbitrary number of random bits, to another public coin protocol that uses a number of random bits that is logarithmic in the size of the input.

Lemma 21. *Let $\epsilon, \delta > 0$ be such that $\epsilon < 1/3$, and let f be a function $f : \mathcal{G}_n \to \{0,1\}$. Suppose that there exists a r-round ϵ-error public coins protocol $\mathcal{P}$ computing f in the* BCLIQUE *model with bandwidth b. Then there exits a r round $\epsilon+\delta$-error public coins protocol $\mathcal{P}''$ computing f in the* BCLIQUE *model with bandwidth b that uses $\mathcal{O}(\log n + \log \delta^{-1})$ public random bits.*

Proof. Let G be the input graph and denote denote by x_i the input of node i. Let q be a Boolean vector of the dimensions required for the random vector of protocol $\mathcal{P}$. Let $\mathcal{P}(G,q)$ be the output of the protocol $\mathcal{P}$ with input G and public random vector q.

Denote by $Z(G,q)$ the random variable over the bits q picked uniformly at random indicating if $\mathcal{P}(G,q)$ equals $f(G)$, formally:

$$Z(G,q) = \begin{cases} 1 & \text{if } \mathcal{P}(G,q) \neq f(G) \\ 0 & \text{otherwise} \end{cases}$$

Since $\mathcal{P}$ has error probability ϵ, then for every fixed graph G, the expected value of $Z(G,q)$ is at most ϵ. Indeed, $E_q(Z(G,q)) = Pr(Z(G,q)=1) = Pr(\mathcal{P}(G,q) \neq f(G)) \leq \epsilon$.

Let $\vec{q} = q_1, \ldots q_t$ be t random strings, with $t = (2n^2)/(3\delta^2)$, and define the following public coin randomized protocol $\mathcal{P}_{\vec{q}}$ as follows. On input G, each node picks using the public coins an integer $i \in \{1, \ldots, t\}$ uniformly at random, and then run $\mathcal{P}$ on input G with random string q_i. In other words they pick $i \in \{1, \ldots t\}$ and, compute $\mathcal{P}(G, q_i)$.

Suppose we pick $\vec{q} = q_1, \ldots, q_k$ independently and uniformly at random, and we fix a graph G. Let $\eta_i(G)$ be a random variable for each $i \in [t]$ defined as $\eta_i(G) = Z(G,q_i) - \epsilon$. Note that $\{\eta_1(G), \ldots, \eta_t(G)\}$ are independent, $E_{\vec{q}}(\eta_i(G)) = E_{\vec{q}}(Z(G,q_i)) - \epsilon = 0$, and $|\eta_i(G)| \leq 1$ (since $\epsilon < 1/2$). Therefore, we can use a Chernoff bound to obtain:

$$Pr_{\vec{q}}\left(\sum_{i\in[t]} \eta_i(G) > \delta \cdot t\right) \quad \leq \quad e^{-\delta^2 \cdot t/2} = 2^{-\delta^2 \cdot t/2 \cdot \log_2(e)} = 2^{-n^2 \cdot log_2(e)/3} < 2^{-n^2}$$

Then

$$Pr_{\vec{q}}\left(\exists G \in \mathcal{G}_n, \sum_{i\in[t]} \eta_i(G) > \delta \cdot t\right) \leq \sum_{G\in\mathcal{G}_n} Pr_{\vec{q}}\left(\sum_{i\in[t]} \eta_i(G) > \delta \cdot t\right) < \sum_{G\in\mathcal{G}_n} \frac{1}{2^{n^2}} = 2^{\binom{n}{2}} \cdot \frac{1}{2^{n^2}} < 1$$

Thus

$$Pr_{\vec{q}}\left(\forall G \in \mathcal{G}_n, \sum_{i\in[t]} \eta_i(G) \leq \delta \cdot t\right) > 0$$

so there exists a choice $\overline{q} = (q_1, \ldots, q_t)$ such that

$$\frac{1}{t}\sum_{i\in[t]} Z(G,q_i) \leq \epsilon + \delta \quad \forall G \in \mathcal{G}_n$$

We define then $\mathcal{P}' = \mathcal{P}_{\overline{q}}$, and we obtain that $\forall G \in \mathcal{G}_n$:

$$\begin{aligned} Pr_i(\mathcal{P}'(G,i) \neq f(G)) &= \sum_{i\in[t]} Pr([\text{the protocol chooses } i] \wedge [\mathcal{P}(x,q_i) \neq f(x)]) \\ &\leq \frac{1}{t}\sum_{i\in[t]} Z(G,q_i) \leq \epsilon + \delta \end{aligned}$$

Therefore $\mathcal{P}''$ is an r round $\epsilon+\delta$-error public coin randomized protocol for f in the BCLIQUE$[b]$ model, that uses $\lceil \log t \rceil = \mathcal{O}(\log n + \log \delta^{-1})$ random bits. □

We are now ready to show Theorem 14.

Proof of Theorem 14. Let $\mathcal{P}$ a protocol as in the statement of the theorem, and for $\delta > 0$ let $\mathcal{P}''$ be the protocol obtained from $\mathcal{P}$ using Lemma 21, i.e., $\mathcal{P}''$ is a r round $\epsilon+\delta$-error public coin randomized protocol on the BCLIQUE$[b]$ model, that uses $\lceil \log t \rceil = \mathcal{O}(\log n + \log \delta^{-1})$ random bits. We simply construct an $\epsilon+\delta$-error private coin randomized protocol $\mathcal{P}'$ on the BCLIQUE model as follows. First, node 1 produces using its private coins a Boolean random vector q of size $\lceil \log t \rceil$, and communicates it in the first round. Then, all the nodes simulate $\mathcal{P}''$ on their inputs considering q as public random vector. Clearly $\mathcal{P}'$ runs in $r+1$ rounds and has bandwidth $b + \lceil \log t \rceil = \mathcal{O}(b + \log n + \log \delta^{-1})$. □

9.3 Lower bounds on deterministic protocols

Theorem 15. *Any one-round deterministic protocol that computes* Twins, Twin_x *or* Translated-Twins *in the* BClique *model has cost* $\Theta(n)$.

The upper bounds of $\mathcal{O}(n)$ are trivial so we only need to prove the lower bounds. For the first two problems, we use the following reduction.

Lemma 22. *Assume that there is a one-round deterministic protocol solving* Twins *(resp.* Twins_{2n+1}*) on in the* BClique *model on graphs of size* $2n+1$*, with cost* $g(n)$*. Then one can solve* Reconstruction *on graphs of size* n *using messages of size* $2g(n)$.

Proof. Let G be an arbitrary graph of size n, i be an integer between 1 and n and S be a subset of $\{1,\dots,n\}$ not containing i. Denote by $H(i,S)$ the graph on $2n+1$ vertices obtained as follows (see Figure 1):

1. $H[\{1,\dots,n\}] = G$.
2. For each $n+1 \leq j \leq 2n$, its unique neighbor with identifier at most n is $j-n$.
3. Vertex $2n+1$ is adjacent exactly to the nodes of S and to $i+n$.

Claim. We claim that $\text{Twins}(H(i,S))$ (resp. $\text{Twins}_{2n+1}(H(i,S))$ is true if and only if $N_G(i) = S$.

Clearly, if $N_G(i) = S$ then vertex i is a twin of $2n+1$ in graph H. Conversely, we prove that if $H(i,S)$ has two twins u and v then one of them is $2n+1$. This comes from the fact that the edges between $\{1,\dots,n\}$ and $\{n+1,\dots,2n\}$ in $H(i,S)$ form a matching, so no two vertices of $\{1,\dots,2n\}$ may be twins. Now assume that $2n+1$ has a twin u. Since $N_{H(i,S)}(2n+1) \cap \{n+1,\dots,2n\} = \{i+n\}$, the only possibility is that $u = i$. Eventually, i and $2n+1$ are twins if and only if $N_G(i) = S$, which proves our claim.

Now assume that we have a distributed protocol for Twins (or Twins_{2n+1}) on graphs with $2n+1$ vertices (actually it suffices to consider graphs from the family H described above). We construct an algorithm for Reconstruction on an arbitrary graph G of size n.

The nodes construct their messages as follows. Each vertex i sends the message m_i that it would send in the Twins protocol if it had neighborhood $N_G(i) \cup \{i+n\}$ and the message m_i^+ that it would send in the same protocol with neighborhood $N_G(i) \cup \{i+n, 2n+1\}$. That makes messages of size $2g(n)$.

The referee needs to retrieve the neighborhood $N_G(i)$ for each i, from the set of messages. For each i and each subset S of $\{1,\dots,n\}$ not containing i, we simulate the behavior of the referee in the protocol for Twins on graph $H(i,S)$. For this purpose, for each $j \leq n$ we use message m_j if $j \notin S$ and message m_j^+ if $j \in S$. The messages for nodes $k > n$ can be constructed directly by the referee. Note that $\text{Twins}(H(i,S))$ is true iff $N_G(i) = S$, thus we can reconstruct $N_G(i)$. Eventually, this allows to solve Reconstruction on graph G. The same arguments work of we replace the Twins protocol by Twins_{2n+1}. □

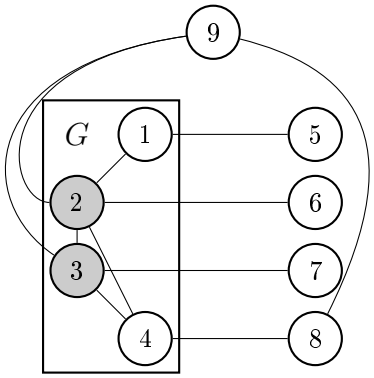
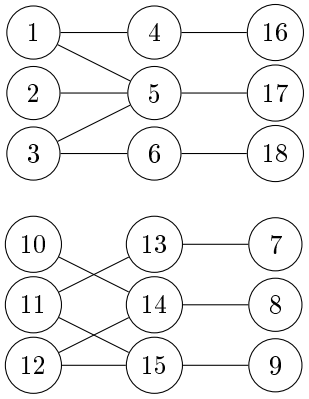


Figure 9.1 – $H(4,S)$, when $S = \{2,3\}$

Since problem RECONSTRUCTION on graphs of size n has cost $\Omega(n)$ (see Section 8.1.1), we conclude that any one-round deterministic protocol for either TWINS or TWINS_{2n+1} also requires messages of size $\Omega(n)$.

For problem TRANSLATED-TWINS, we provide a reduction from OREQ (see Proposition 37 in Section 8.2.1). It will be used both for deterministic and randomized protocols with private coins.



$$x = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \qquad y = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Figure 9.2 – Examples of graphs G_x^1 (top) and G_y^2 (bottom), for a given input (x, y). This is a *yes* instance since $x_3 = y_3$.

Lemma 23. *Assume that there is a one-round protocol solving* TRANSLATED-TWINS *on graphs of size* $6n$ *in the* BCLIQUE *model, with (deterministic, private-coin or public coin) cost* $g(n)$*. Then there is a (deterministic, private-coin or public coin) protocol for function* OREQ *using messages of size* $3ng(n)$*.*

Proof. Let x and y be two $n \times n$ Boolean matrices. We construct a graph $G_{x,y}$ with $6n$ nodes such that $\text{TRANSLATED-TWINS}(G_{x,y}) = \text{OREQ}(x, y)$.

The graph G is formed by two connected components G_x^1 and G_y^2 of $3n$ nodes each, encoding the two matrices as follows (see Figure 9.2 for an example).

G_x^1 has $3n$ nodes numbered from 1 to $2n$ and from $5n+1$ to $6n$. For any $i, j \in \{1, \ldots, n\}$ we put an edge between node i and node $j+n$ if and only if $x_{i,j} = 1$. Then for any $i \in \{1, \ldots, n\}$ we put an edge between node $i+n$ and node $i+4n$. In other words, the node subsets $\{1, \ldots, n\}$ and $\{n+1, \ldots, 2n\}$ induce a bipartite graph representing matrix x, and the node subsets $\{n+1, \ldots, 2n\}$ and $\{5n+1, \ldots, 6n\}$ induce a perfect matching.

The construction of G_y^2, with nodes numbered from $2n+1$ to $5n$ is similar. For any $i, j \in \{1, \ldots, n\}$ we put an edge between node $i+3n$ and node $j+4n$ if and only if $y_{i,j} = 1$. Also, for any $i \in \{1, \ldots, n\}$, we put an edge between node $4n+i$ and node $2n+i$. Thus the node subsets $\{3n+1, \ldots, 4n\}$ and $\{4n+1, \ldots, 5n\}$ form a bipartite graph corresponding to matrix y. The subsets $\{4n+1, \ldots, 5n\}$ and $\{2n+1, \ldots, 3n\}$ induce a matching.

We claim that $\text{TRANSLATED-TWINS}(G_{x,y}) = \text{OREQ}(x, y)$. Assume that $\text{OREQ}(x, y) = 1$. There is an index i such that line number i in x equals line number i in y. Then, by construction, the neighborhood of node $i+3n$ in $G_{x,y}$ is the neighborhood of node i, translated by an additive term $3n$.

Conversely, assume that there is some node $u \in \{1, \ldots, 3n\}$ such that the neighborhood of u is the translated neighborhood of $u+3n$. By construction, the only possibility is that $u \leq n$ (because of the numberings of the matchings the other nodes cannot have translated twins), thus line number u is the same in the two matrices.

To achieve the proof of our lemma, assume that we have a protocol for TRANSLATED-TWINS for graphs with $3n$ nodes, with $g(n)$ bits per message. We design a protocol for OREQ. Recall that for OREQ, each player has a matrix, say x for the first one and y for the second one. The first player constructs graph $G_{x,0} = (G_x^1, G_0^2)$, the second constructs $G_{0,y} = (G_0^1, G_y^2)$ (here 0 denotes the $n \times n$ Boolean matrix whose elements are all 0). The first player sends the $3n$ messages corresponding to the nodes of G_x^1 in the TRANSLATED-TWINS protocol for graph $G_{x,0}$. The second player sends the $3n$ messages corresponding

to the nodes of G_y^2 in protocol Translated-Twins for $G_{0,y}$. The referee collects these $6n$ messages; observe that they are exactly those sent by protocol Translated-Twins for the graph $G_{x,y}$. He applies the same algorithm as the referee of Translated-Twins on these messages. By the claim above, its output is Translated-Twins$(G_{x,y})$, thus OREQ(x, y). Note that the messages used here are of size $\mathcal{O}(3ng(n))$ and that our arguments hold for any type of protocol. □

This achieves the proof of Theorem 15. We finalize this section giving a deterministic lower bound for multi-round protocols solving GraphEq.

Theorem 16. *Any deterministic protocol that computes* GraphEq *in the* BClique *model has cost* $\Omega(n)$.

Proof. Let $\mathcal{P}$ be a deterministic protocol that computes GraphEq in the BClique model, with cost $g(n)$ on input graphs of size $2n$. Call $m = \binom{n}{2}$, and let $x, y \in \{0,1\}^m$ be an input of problem EQ_m. Let G_x and G_y be graphs of size n, where the vertex set is $[n]$ and the edge sets are given by x and y, interpreted as their adjacency matrices. Consider now the graph of size $2n$ called $G_{x,y}$, which is the union of G_x and G_y, after relabel each vertex i of G_y by $i + n$ (see Figure 9.3).

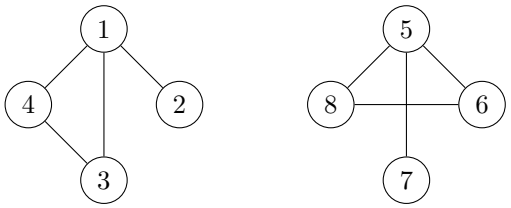


Figure 9.3 – Reduction of EQ_m to GraphEq where $n = 4$, and the inputs of EQ_m are $x = (1,1,1,0,0,1)$ and $y = (1,1,1,0,1,0)$ (the coordinates 1 (resp. 0) in x represent the presence (absence) of edges $(\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\})$ in $G_{x,y}$, while the coordinates of y represent edges $(\{5,6\},\{5,7\},\{5,8\},\{6,7\},\{6,8\},\{7,8\})$.

Note that GraphEq$(G_{x,y})$ equals 1 if and only $G_x = G_y$, i.e., if $x = y$. We define a two-player protocol $\mathcal{P}'$ for EQ_m as follows. At each round, Alice (resp. Bob) simulates protocol $\mathcal{P}$ on each vertex of G_x (resp. G_y), and communicates the obtained messages. Since the cost of $\mathcal{P}$ in $G_{x,y}$ is $g(n)$, the cost of protocol $\mathcal{P}'$ is $n \cdot g(n)$. Recall that the deterministic cost of EQ_m is $\Omega(m)$. We obtain that $n \cdot g(n) = \Omega(\binom{n}{2})$, so $g(n) = \Omega(n)$. □

9.4 Randomized protocols

Theorem 17. *For any constant $c > 0$,* Twins, Twins$_x$, Translated-Twins *and* GraphEq *can be computed with high probability by one-round public coins protocols of cost $O(\log n)$. Problem* Twins$_x$ *can also be computed with high probability by a private coins protocol with cost $O(\log n)$.*

Proof. Let $n^{c+3} < p \leq 2n^{c+3}$ be a prime number. A random $t \in \mathbb{Z}_p$ is chosen uniformly at random using $\mathcal{O}(\log(n))$ public random bits. Let x_i be the input vector of node i, i.e., the i-th row of the adjacency matrix of G. A protocol for Twins consists in each node sending the message $m_i = P(x_i, t)$, i.e., the fingerprint of x (see Section 8.2.2). The referee outputs 1 if and only if $m_i = m_j$ for some pair $i \neq j$. A protocol for Twins$_x$ send the same messages, but this time the referee checks whether $m_x = m_i$ for some $i \neq x$.

In the protocol for GraphEq on graphs of size $2n$, each node $i \in [n]$ calls y_i the first n coordinates of x_i, and each node $i \in [2n] \setminus [n]$, calls y_i the last n coordinates of x_i. Then, each node i communicates $m_i = P(y_i, t)$, the fingerprint of y_i produced using t.

The protocol for Translated-Twins on n-node graphs is slightly different. If a node $i \leq n/2$ has a neighbor $j > n/2$, it sends a special "no" message specifying that it cannot be a candidate for having a translated twin. Otherwise, let y_i^1 be the $n/2$-bits vector formed by the $n/2$ first bits of x_i. Thus y_i^1 is the characteristic vector of $N(i) \cap \{1, \ldots, n/2\}$. Player i sends the message $m_i = P(y_i^1, t)$. Symmetrically, for nodes labeled $i > n/2$, if i has some neighbor $j \leq n/2$ it sends the "no" message. Otherwise, let y_i^2 be the $n/2$-bits vector formed by the last $n/2$ bits of x_i. Hence y_i^2 corresponds to $N(i) \cap \{n/2, \ldots, n\}$, "translated"

by $-n/2$. Player i sends the message $m_i = P(y_i^2, t)$. Then the referee returns 1 if $m_i = m_{i+n/2}$ for some $i \leq n/2$.

Clearly, for protocol TWINS (resp. TWINS$_x$, TRANSLATED-TWINS, GRAPHEQ), if the input graph is a yes-instance then the protocol outputs 1. The probability that TWINS answers 1 on a no-instance is the probability that two fingerprints of two different vectors are equal. For each fixed pair of nodes this probability is at most $1/n^{c+2}$, so altogether the probability of a wrong answer is at most $1/n^c$. With similar arguments for TWINS$_x$, TRANSLATED-TWINS and GRAPHEQ the probability of a wrong answer is at most $1/n^{c+1}$, since the referee makes n tests and each may be a false positive with probability at most $1/n^{c+2}$.

For TWINS$_x$ with private coins, each node i sends a bit stating if it sees x, a number t_i chosen uniformly at random in the interval $n^{c+2} < p \leq 2n^{c+2}$ and also $FP(x_i, t_i)$. The referee retrieves the neighborhood of node x (which was sent bit by bit by all the others) and then, for each $i \neq x$, it constructs $FP(x_x, t_i)$ and compares it to $FP(x_i, t_i)$. If the values are equal for some i, the referee outputs 1, otherwise it outputs 0. Again, any yes-instance will answer 1, and the probability that a no-instance (wrongly) answers 1 is at most $1/n^c$. □

The fact that TRANSLATED-TWINS requires $\Omega(\sqrt{n})$ bits per node for any private coins, ϵ-error randomized protocol follows directly by Lemma 23 and Proposition 37.

Theorem 18. *For any $\epsilon < 1/2$, $C_\epsilon^{\text{priv}}(\text{TRANSLATED-TWINS}) = \Omega(\sqrt{n})$.*

Theorems 17 and 18 show that problem TRANSLATED-TWINS separates the private coins and the public coins protocols.

In order to complete the table of the Introduction, we also observe that problems TWINS and TRANSLATED-TWINS can be solved by randomized private coins protocols using $\mathcal{O}(\sqrt{n}\log n)$ bits.

Theorem 19. *For any $c > 0$, there is a randomized private coins protocol for* TWINS*, and* TRANSLATED-TWINS *using messages of size $\mathcal{O}(\sqrt{n}\log n)$ and having $1/n^c$ one-sided error.*

Proof. Babai and Kimmel in [9] propose a private coins protocol with 1/3 one sided error and $\mathcal{O}(\sqrt{n})$ communication cost for EQ_n, in the *number-in-hand* model with two players (see Proposition 35). Let us call this protocol $\mathcal{P}_0$. As the authors point out, this protocol is symmetrical, in the sense that both players compute the same function on their own input. We define the protocol $\mathcal{P}$ as one obtained by simulating $(c+2)\log_3 n$ calls to protocol $\mathcal{P}_0$. More formally, in $\mathcal{P}$ each player creates $(c+2)\log_3 n$ times the message that it would create in $\mathcal{P}_0$, using at each time independent tosses of private coins. The referee answers 1 if and only if the referee of $\mathcal{P}_0$ would have answered 1 on each of the $(c+2)\log_3 n$ pairs of messages. Therefore $\mathcal{P}$ is a private coin randomized protocol for EQ_n with one sided error smaller than $1/n^{c+2}$, and cost $\mathcal{O}(\sqrt{n}\log n)$.

A one sided private coin randomized protocol $\mathcal{P}'$ for TWINS is one where each node plays the role of Alice in $\mathcal{P}$ taking as an input the characteristic function of its neighborhood, and then the referee simulates the role of the referee in $\mathcal{P}$ for each pair of messages. Similarly, a protocol $\mathcal{P}''$ for TRANSLATED-TWINS works as follows: each node i sends "*no*" in the same cases described in the proof of Theorem 17, and otherwise it simulates the role of Alice on input y_i^1 formed by the first $n/2$ bits of x_i, if $i \leq n/2$ or on input y_i^2 formed by the $n/2$ last bits of x_i if $i > n/2$, where x_i is the characteristic function of $N(i)$. The referee then simulates the referee of $\mathcal{P}$ on the messages of i and $i+n$ every time none of them say "*no*".

Since $\mathcal{P}$ has just one sided error, if TWINS (resp. TRANSLATED-TWINS) is *true*, $\mathcal{P}'$ (resp. $\mathcal{P}''$) will always accept. On the other hand, if TWINS (resp. TRANSLATED-TWINS) is *false*, then the probability that $\mathcal{P}'$ (resp. $\mathcal{P}''$) accepts is the probability that $\mathcal{P}$ accepts for at least one pair of vertices, and then the error of $\mathcal{P}'$ (resp. $\mathcal{P}''$) is at most n^2 times (resp. n times) the error of $\mathcal{P}$. We obtain that $\mathcal{P}'$ and $\mathcal{P}''$ have at most $1/n^c$ one sided error, and communication cost $\mathcal{O}(\sqrt{n}\log n)$. □

Consider the Boolean function TRIANGLE(G) that outputs 1 if and only if G has a triangle, and the function DIAM3(G), that outputs 1 if and only if G has diameter at most 3. In [14] is shown that the deterministic message sizes of these problems are lower-bounded by $\Omega(n)$, using a reduction from RECONSTRUCTION. However, as seen in Theorem 15, a reduction from RECONSTRUCTION does not imply lower-bounds on the message sizes of randomized protocols.

In the following theorem, we extend the techniques in [14] to reduce the problems TRIANGLE(G) and DIAM3(G) from INDEX, showing that the message sizes of randomized protocols for these problems are also of size $\Omega(n)$.

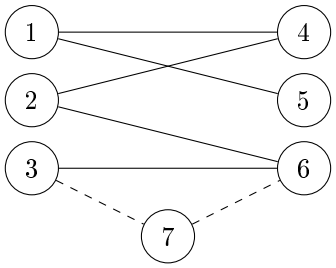
Theorem 20. *For any $\epsilon < 1/3$, any one-round ϵ-error public coins protocol computing* Triangle(G) *(resp.* Diam3(G)*) has cost $\Omega(n)$.*

Proof. Consider the Index function in the model *number-in-hand* with two players: the first player, say Alice, has as input an m-bits Boolean vector x and the second player, Bob, has an integer $q, 1 \leq i \leq m$. Then Index$(x, q) = x_q$, the qth coordinate of Alice's vector. We will use the fact that for any $\epsilon < 1/2$, any public coins randomized protocol for Index requires $\Omega(m)$ bits (see, e.g., [94, 95] for a proof). We may assume w.l.o.g. that $m = n^2$.

In [14], Becker *et al.* show that for the deterministic communication cost for Triangle and Diam3 is $\Theta(n)$, by showing that if there is a protocol $\mathcal{P}$ of cost c for Triangle or Diam3, then there is a protocol for Reconstruction in bipartite graphs of cost $2c$. We slightly modify their proof to obtain a reduction from Index.

Let $\epsilon < 1/2$, and $\mathcal{P}$ be a ϵ-error randomized public coins protocol for Triangles on n-nodes graphs, using $c(n)$ bits. We give a protocol for Index using $2n \cdot c(2n+1)$ bits.

Let x be an $m = n^2$-bits vector. Let H_x be the bipartite graph with vertex set $\{1, \ldots, 2n\}$, such that for any $1 \leq k, l \leq n$, if $x_{(k-1)n+l} = 1$ then H_x has an edge between nodes k and $l + n$. Consider the family of graphs $H_x(i, j)$ obtained from H_x by adding a node $2n + 1$ whose neighbors are nodes i and $j + n$ (for any $1 \leq i, j \leq n$). Observe that $H_x(i, j)$ has a triangle if and only if $x_{(i-1)n+j} = 1$, in which case the triangle is formed by the nodes $\{i, j + n, 2n + 1\}$. To simplify the notation we also define the graph $H_x(0, 0)$ obtained from H_x by adding an isolated node $2n + 1$.



$x = (1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1)$
$q = 9$

Figure 9.4 – An illustration of $H_x(3, 6)$ when $x = (1, 1, 0, 1, 0, 1, 0, 0, 1)$ and $q = 9$.

The protocol for Index is as follows. Bob sends its input q, which only costs $\mathcal{O}(\log n)$ bits. Alice constructs the family of graphs $H_x(i, j)$, for all pairs $1 \leq i, j \leq n$ and for $(i, j) = (0, 0)$. Any node $k \leq 2n$ has exactly two possible of neighborhoods, depending whether it is adjacent to $2n + 1$ or not. For each $k \leq 2n$, Alice creates the message $m^+(k)$ that the protocol for Triangle would send for node k in the graph $H_x(k, 1)$ (if $k \leq n$) or in the graph $H(1, k - n)$ (if $k > n$). It also creates the message $m^-(k)$ that Triangle would construct for node k in the graph $H_x(0, 0)$. In full words, $m^-(k)$ corresponds to the case when the neighborhood of k is the same as in H_x, and $m^+(k)$ to the case when this neighborhood is the neighborhood in H_x, plus node $2n+1$. Then Alice sends, for each k, $1 \leq k \leq 2n$, the pair of messages $(m^-(k), m^+(k))$. Therefore Alice uses $2n \cdot c(2n + 1)$ bits. It remains to explain how the referee retrieves the bit x_q. Let i, j such that $q = (i - 1)n + j$. Observe that $x_q = 1$ if and only if graph $H_x(i, j)$ has a triangle, therefore the referee must simulate the behavior of the referee for Triangle on $H_x(i, j)$. For this purpose, the referee computes the message that node $2n + 1$ would have sent on this graph (it only depends on i and j) and observes that protocol $\mathcal{P}$ on $H_x(i, j)$ would have sent message $m^+(i)$, $m^+(j+n)$ and $m^-(k)$ for any $k \leq 2n$ different from i and j. Therefore the referee can give the same output as $\mathcal{P}$ on $H_x(i, j)$, that is it outputs bit x_q. The protocol for Index will have ϵ error and will use $2n \cdot c(2n + 1)$ bits. Thus $\mathcal{P}$ requires $\Omega(n)$ bits.

The proof for Diam3 is based on a similar reduction. Let $D_x(i, j)$ be the graph obtained from H_x by adding three nodes : node $2n + 1$ seeing all nodes $k \leq 2n$, node $2n + 2$ seeing i and node $2n + 3$ seeing

$j+n$. Graph $D_x(0,0)$ is similar with the difference that nodes $2n+2$ and $2n+3$. Observe (see also [14]) that $D_x(i,j)$ has diameter 3 if and only if $x_{(i-1)n+j}=1$. The rest of the proof follows as before. □

9.5 Discussion

We succeed to separate, with respect to the bandwidth, different types of one-round protocols, say deterministic, private-coin and public-coin protocols. An interesting research perspective consists in identify for which problems the cost of a private-coin protocol is lower bounded by the square root of the cost of the best deterministic protocol.

We conjecture that any graph property that is preserved under isomorphism can not be recognized with high probability by a private-coin protocol with a cost that is asymptotically smaller than the square root of a deterministic protocol.

There exist some other techniques that might allow to simulate public-coin protocols in the private-coin model. For example, using the birthday paradox. Suppose that we can bound the number of possible sequences of random bits that uses a public coin protocol $\mathcal{P}$ to $\mathcal{O}(n)$. Consider now the following private-coin protocol $\mathcal{P}'$. Each node x picks uniformly at random, using its own private coins, $s=\mathcal{O}(\sqrt{n})$ different sequences of random bits from the $\mathcal{O}(n)$ sequences that use protocol $\mathcal{P}$. Then x produces s messages simulating $\mathcal{P}$ in each one of the sequences, and communicates the messages together with the s sequences of random bits. Now take the messages sent by two different nodes. With high probability, these two nodes will coincide in at least one of the s different sequences of random bits. It would be interesting to find if this reasoning can be applied to design a private coin protocol for CONNECTIVITY.

The natural challenge is to determine the cost of function TWINS for randomized protocols with private coins. Using the techniques of Babai and Kimmel [9] for EQ, one can prove that TWINS can be solved by a one-sided, bounded error protocol with private coins and messages of size $\mathcal{O}(\sqrt{n}\log n)$. We believe that the message size complexity of TWINS for private coins protocols is $\Omega(\sqrt{n})$. Another challenge is to determine the cost of TWINS for multi-round deterministic protocols. Using the matrix multiplication protocol of [38], we can obtain a deterministic protocol with cost $\mathcal{O}(n^{1/3}\log n)$ in the UCLIQUE model. This protocol runs in $\mathcal{O}(n^{1/3})$ rounds with bandwidth $\mathcal{O}(\log n)$.

Another interesting question is CONNECTIVITY, that will be discussed in Chapter 12. To the best of our knowledge, the cost of one round protocols computing this function is wide open. Recall that, in the randomized, public coins setting, there exists one-round protocol with cost $\mathcal{O}(\log^4 n)$ bits, due to Ahn, Guha and McGregor [1]. Can this upper bound be improved to $\mathcal{O}(\log n)$? For randomized protocols with private coins and/or for deterministic protocols, can one prove a lower bound of $\Omega(n^c)$ for some constant $c<1$?

Chapter 10

Distributed recognition of graph classes

The goal of this chapter is to study the reconstruction and recognition of small classes of graphs in the BClique model. We start improving a one-round deterministic protocol of Becker *et al.* (Proposition 31) to reconstruct and recognize graphs of bounded degeneracy. Later, we give one-round public-coins protocols for recognizing and reconstructing distance-hereditary graphs and graphs of bounded modular width with high probability. Finally, we give a generic constant-round protocols that recognize and reconstruct any small family of graphs with sublinear cost. We finish extending this last result to the UClique.

10.1 Introduction

The known lower bounds in the BClique model, that use reductions from communication complexity and Reconstruction, are based on the following idea: if there exists a protocol for computing some function (to which we are reducing), then we can use that protocol to communicate too much information (for example, enough information to reconstruct the graph). One may ask then what happens when the input of our problem does not contain much information, i.e., is restricted to some small family. Since a protocol of cost $c(n)$ communicates $n \cdot c(n)$ bits, a class $\mathcal{G}$ of graphs of size n can be potentially be recognized and/or reconstructed with cost $\log |\mathcal{G}|/n$.

We say that a class of graphs $\mathcal{G}$ is *small* if the subset of graphs in $\mathcal{G}$ of size n, called $\mathcal{G}_n$ is in $2^{o(n^2)}$. Examples of such classes are planar graphs, or more generally graphs of bounded genus, line graphs, graphs of bounded treewidth, cliquewidth or modular width, interval graphs, circle graphs, circular arc graphs, k-polygon, etc.

Recall that the problem Recognition of $\mathcal{G}$ consists in determining if the input graph belongs to $\mathcal{G}$. We say that a protocol solves Reconstruction on $\mathcal{G}$ when on input graph $G \in \mathcal{G}$, at the end of the protocol each node knows $E(G)$. Solving Reconstruction on $\mathcal{G}$ with some nontrivial protocol means that we can solve in the same bounds any graph problem restricted to the class $\mathcal{G}$.

We first improve the result of Proposition 31, showing that there is a one-round deterministic protocol that recognizes and reconstruct d-degenerate graphs with cost $\mathcal{O}(d \cdot \log n)$ in the BClique model. Our protocol uses *deterministic linear sketches* (see Section 8.2.4) to find a sequence of vertices $\{v_1, \ldots, v_n\}$ such that v_i has degree at most d in $G[\{v_{i+1}, \ldots, v_n\}]$. The family of d-degenerate graphs is a big family of sparse graphs, containing for example graphs of bounded treewidth, or planar graphs. We show that our protocol for reconstruction is tight, even for one-round public-coin protocols.

Then, we improve extend the protocol that recognizes and reconstruct cographs (Proposition 32), to the class of Distance-Hereditary and Bounded Modular width graphs. We show that such families can be reconstructed and recognized with high probability by randomized protocols with cost $\mathcal{O}(\log n)$ in the BClique model.

Finally, we show a more general result involving any small class of graphs. We show that for any class of graphs $\mathcal{G}$, there exists a one-round deterministic protocol that in the BClique model that solves Reconstruction on $\mathcal{G}$ with cost $\mathcal{O}(\sqrt{\log(|\mathcal{G}_n|) \cdot \log n} + \log n)$. We also show a one-round private-coin protocol and a two-round deterministic protocol solving Reconstruction of $\mathcal{G}$ with the same cost.

This last result implies sublinear recognition and reconstruction protocols for small graph classes

which are not covered by the previous chapters. For example, many graph classes with a few minimal separators, e.g., interval graphs, polygon-circle graphs, d-trapezoid graphs have $2^{\mathcal{O}(n \log n)}$ labeled graphs, so they can be reconstructed and recognized with our protocols with cost $\mathcal{O}(\sqrt{n} \log n)$. Other example is the class of chordal bipartite graphs, which contain $2^{\mathcal{O}(n(\log n)^2)}$ labeled graphs [35], so it can also be reconstructed and recognized with cost $\mathcal{O}(\sqrt{n} \log n)$. In general, if a class of graphs contain $2^{n^{1+\epsilon}}$ graphs, then it can be recognized and reconstructed with our protocol with cost $\mathcal{O}(n^{(1+\epsilon)/2}\sqrt{\log n})$.

We finish this chapter e extend this results to constant-round deterministic protocols in the UCLIQUE model, with cost $\mathcal{O}(\log(|\mathcal{G}_n|)/n + \log n)$.

10.2 Bounded degeneracy

In this section we show an optimal one-round deterministic protocol for reconstruct and recognize d-degenerate graphs with cost $\mathcal{O}(d \cdot \log n)$, for any $0 \leq d \leq n$. In fact, our protocol computes the *d-Core*, which is a maximum set of vertices V' such that every vertex in V' has at least $d+1$ neighbors in $G[V']$. Our protocol computes a maximum sequence of vertices $W = \{w_1, \ldots, w_k\}$, such that w_i has degree at most d in $G[V \setminus \{w_1, \ldots, w_{k-1}\}]$, and outputs all the edges adjacent to nodes in W. The set W is called a *d-prunning*. Clearly, if $W \neq V$, then $G[V \setminus W]$ is a d-Core.

Theorem 21. *There is a one-round, deterministic protocol in the* BCLIQUE *model with cost $\mathcal{O}(d \cdot \log n)$ that reconstructs the input graph G if the graph is d-degenerate, otherwise returns a d-prunning W and all the edges adjacent to nodes in W.*

Proof. In the d-pruning protocol, each player i sends the message $M_i = (M_i^1, M_i^2) = (d_i, f(a_i))$, where d_i is its degree, a_i is the row of the adjacency matrix corresponding to node i, and f is the function of Lemma 20. The number of communicated bits is $\mathcal{O}(d \log n)$. Observe that the second coordinate M_i^2 of message i is seen as an element of $\mathbb{F}_p$, hence all arithmetic operations performed on M_i^2 are modulo p.

Call $M = \{M_j\}_{j \in V(G)}$ the *messages vector* of G. The referee use M to prune the graph in at most n phases (without any communication). Initially we call $G_1 = G$, $M(1) = M$. The input of phase i is the messages vector $M(i)$ of a graph G_i. The idea is to look in G_i for a node u_i of degree at most d, together with all its adjacent edges $E(u_i)$ in G_i. Then update $M(i)$ to obtain the messages vector $M(i+1)$ of the graph G_{i+1}, which corresponds to the deletion of u_i from G_i. The output of each phase is $M(i+1)$ together with u_i and $E(u_i)$.

Formally, let $\mathcal{B} = \{b \in \{0,1\}^n : \sum_{i=1}^n b_i \leq d\}$ be the family of d-sparse Boolean vectors of dimension n. The first step of phase i is to look for a node u_i such that $M(i)^1_{u_i} \leq d$. Then, using $M(i)^2_{u_i}$, we obtain $E(u_i)$. This can be done testing, for each $b \in \mathcal{B}$ if $f(b) = M(i)^2_{u_i}$. Since $M^1_{u_i} \leq d$, we have that the adjacency vector of u_i in G_i is d-sparse, then according to the injectivity property of f, the unique vector of $\mathcal{B}$ that satisfies $f(b) = M^2_{u_i}$ is the adjacency vector of u_i in G_i. Finally, we can use $E(u_i)$ and $M(i)$ to compute $M(i+1) = \{M(i+1)_j\}_{j \in V(G_{i+1})}$ as follows:

$$M(i+1)_j = \begin{cases} M(i)_j & \text{if } \{u_i, j\} \notin E(u_i) \\ (M(i)^1_j - 1, M(i)^2_j - f(e_{u_i})) & \text{if } \{u_i, j\} \in E(u_i) \end{cases}$$

where e_k is the Boolean vector of dimension n with a single one in the k-th coordinate. Notice that if $M(i)$ corresponds to the messages vector of G_i, then $M(i+1)$ corresponds to the messages vector of $G_{i+1} = G[V(G_i) - \{u_i\}]$. Indeed, let $a(i)_j$, $d(i)_j$ and $a(i+1)_j$, $d(i+1)_j$ be the adjacency vectors and degrees of node j in $G(i)$ and $G(i+1)$, respectively. If $\{u_i, j\} \notin E(u_i)$, then $a(i)_j = a(i+1)_j$ and $d(i)_j = d(i+1)_j$, so $M(i)_j = M(i+1)_j$. On the other hand, if $\{u_i, j\} \in E(u_i)$,then $M(i+1)_j = d(i+1)_j = d(i)_j - 1 = M(i)_j - 1$ and $a(i+1)_j = a(i)_j - e_{u_i}$. By the linearity of f, we obtain that $M(i+1)_j = f(a(i)_j - e_{u_i}) = M(i)_j - f(e_{u_i})$. Note that each node communicates their messages using bandwidth $\mathcal{O}(d \cdot \log n)$. □

We remark that for many classes of graphs we can bound the degeneracy. In the following we present some results that relate some graph structures with the degeneracy. We start giving a result that states that graphs of bounded degeneracy have bounded treewidth.

Proposition 44. *[125] For any graph G, the degeneracy of G upper bounded by* $\mathrm{tw}(G)$.

Proof. Let $\mathcal{T} = (\{X_i\}_{i\in[I]}, T = (I, E_k))$ be a tree decomposition of G, such that $\max_{i\in I} |X_i| = \text{tw}(G)$ and $|I|$ is minimum. We claim that exists a vertex $v \in V$ such that there exists only one bag X_i containing it. This vertex must exists, otherwise any leaf node i_1 of T, which is adjacent to a node i_2, must satisfy $X_{i_1} \subseteq X_{i_2}$. Therefore $(\{X_i\}_{i\in[I]\setminus\{i_1\}}, T = (I \setminus \{i_1\}, E_T \setminus \{i_1, i_2\}))$ is also a tree decomposition of G that contradicts the minimality of I. Note that such vertex v must have degree at most $\text{tw}(G)$, since all its neighbors belong to the unique bag containing it.

For $k \in \{0, \ldots, n\}$, pick v_{k+1} as any vertex that is contained in a unique bag of $\mathcal{T}_k$, where $\mathcal{T}_k = (\{X_i\}_{i\in[I_k]}, T_k = (I_k, E_k))$ is a tree decomposition of $G_k = G - \{v_1, \ldots, v_i\}$ (with the convention $G_0 = G$) and which satisfies that $\max_{i\in I_k} |X_i| \leq \text{tw}(G)$ and $|I_k|$ is minimum. Note that the degree of v_{k+1} in G_k is at most $\text{tw}(G)$. Therefore G is of degeneracy at most $\text{tw}(G)$, since $\{v_1, \ldots, v_n\}$ is a sequence of vertices such that v_k has degree at most $\text{tw}(G)$ in $G[v_{k+1}, \ldots, v_n]$, for every $k \in [n]$. □

Let H be some graph, and call $ex(n, H)$ be maximum number of edges over all graphs of size n that not containing H as a subgraph. Druker *et al.* [57] showed that the class of graphs not containing H as a subgraph is a class of degeneracy bounded by $ex(n, H)$ and n.

Proposition 45. *[57] Let H be a graph and G be another graph not containing H as a subgraph. Then G is $2 \cdot ex(n, H)/n$-degenerate.*

Another important family of graph classes that have bounded degeneracy is the class of H-minor free, for some graph H. Recall that the family of H-minor-free graphs is the set of all graphs not containing H minor. From a result due to Kostochka [93] we know that any H-minor-free graph is d-degenerate, where the constant d depends only on the number of vertices of H.

Proposition 46 (Kostochka [93])**.** *There exists a function f such that any H-minor-free graph is $f(|H|)$-degenerate.*

This last result implies in particular that planar graphs, and in general graphs of bounded genus, have bounded degeneracy.

We finish this section showing that the protocol for recognition is tight. We show that any protocol (including randomized public coins or multi-round protocols) that reconstructs graphs of degeneracy d must communicate $\Omega(d \log n)$ bits.

Theorem 22. *Any ϵ-error randomized protocol that with high probability reconstructs a d-degenerate graph in the* BClique *model has cost $\Omega(d \log n)$ bits.*

Proof. Let $c(n)$ be the cost of an optimal protocol that computes the d-pruning of the input graph in the broadcast congested clique model. Since any d-pruning protocol reconstructs d-degenerate graphs, then with $n \cdot c(n)$ bits we can completely encode any graph of degeneracy d. Consider $\mathcal{G}$ the familly of bipartite graphs $G = (V_1 \cup V_2, E)$, with $|V_1| = |V_2| = n/2$, and such that any vertex $v \in V_1$ has exactly d neighbors in V_2. Clearly any graph in $\mathcal{G}$ is d-degenerate, and $|\mathcal{G}| \geq \binom{n/2}{d}^{n/2}$, i.e., $|\mathcal{G}| \geq 2^{\Omega(nd \log n)}$. Then $n \cdot c(n) \geq \log(2^{\Omega(nd \log n)})$, so $c(n) = \Omega(d \log n)$. □

10.3 Distance-Hereditary and bounded modular width graphs

In this section we give a one-round public-coin protocol to recognize the classes of distance-hereditary graphs and graphs of bounded modular width in the BClique model with high probability and cost $cO(\log n)$. Our result is inspired in a protocol of Kari *et al.* [87], used for the recognition and reconstruction of cographs.

Recall that a graph $G = (V, E)$ is distance-hereditary if, the distance between any two nodes of a same connected component is preserved in any induced subgraph that contains them. Moreover, distance-hereditary graphs have a characterization according to the existence of a *twin-pendant vertex decomposition*.

On the other hand, recall that a graph G is of modular width at most k if:

1. G has at most one node (the base case).
2. G is a disjoint union of two graphs of modular width at most k.

3. G is a *join* of two graphs of modular width at most k, i.e., G is obtained from two disjoint graphs of modular width at most k by taking their disjoint union and then adding all possible edges between these graphs.

4. the node set of G can be partitioned into $p \leq k$ modules $V_1, \ldots, V_p$ such that $G[V_i]$ is of modular width at most k, for all i, $1 \leq i \leq p$.

Given a graph of modular-width at most k, one can associate to it a rooted decomposition tree T following the rules above. Each internal node of the tree corresponds to operations 2 to 4, and leaves correspond to single vertices (operation 1). This tree is very similar to the well-known modular decomposition tree of the graph. It easy to observe that, given the decomposition tree of graph G, for any node u of the tree, the set of leaves $V[u]$ of the subtree rooted in u is a module of G (see e.g., [66]). The following observations are crucial for our algorithm.

Lemma 24. *Let G be a graph of modular-width at most k, having strictly more than k vertices. Then G has a non-trivial module of at most k vertices.*

Proof. We prove that G, of modular width at most k, has a module of size at most k. Consider a decomposition tree T of G. Let u be one of the internal nodes such that all sons of u are leaves. Then the corresponding module $V[u]$ is formed by the at most k leaves of the subtree rooted in u, and satisfies the lemma. □

Lemma 25. *Let G be a graph and $\{u_1, \ldots, u_l\}$ be a module of G, with $2 \leq l \leq k$. G is of modular-width at most k if and only if $G - \{u_2, \ldots, u_l\}$ is a of modular width at most k.*

Proof. Let $G = (V, E)$ be of modular width at most k. Actually, any induced subgraph $G[W]$ is of modular width at most k. Indeed, consider a decomposition tree for G, remove all leaves contained in $V \setminus W$, and shortcut all nodes with at most one son. We obtain a decomposition tree of $G[W]$, proving that it is of modular width at most k.

Conversely, assume that $\{u_1, \ldots, u_l\}$ is a module of G, and $G' = G - \{u_2, \ldots, u_l\}$ is a of modular width at most k. Replace, in a decomposition tree of G' the leaf u_1 with a decomposition tree of $G[\{u_1, \ldots, u_l\}]$. We obtain a decomposition tree of G, certifying that its modular width is at most k. □

Let p be a prime number, and let $G = (V, E)$ be a graph. A vector $m = ((a_v, b_v))_{v \in V}$ is *valid* for G at $t \in \mathbb{F}_p$ if there is a linearly independent family of polynomials $\Phi = (\phi_v)_{v \in V}$ in $\mathbb{F}_p[X]$ such that $a_v = \phi_v(t)$ and $b_v = \sum_{w \in N_G(v)} \phi_w(t)$, for each $v \in V$. Note that a if node w in G has degree 1 and neighbor u, then $b_w = a_u$. Moreover, two twin nodes u, w in G satisfy either $b_u = b_w$ (if they are non-adjacent) or $b_u + a_u = b_w + a_w$ (if they are adjacent). In the following lemma, we show how to compute from a valid vector of a graph G, a valid vector of the graph obtained by the deletion of some specific nodes.

Lemma 26. *Let $m = ((a_v, b_v))_{v \in V} \in (\mathbb{F}_p)^{2n}$ be valid for $G = (V, E)$ at t.*

1. *Let u, w be twins in G such that $a_u \neq a_w$. Then, the vector $m' = ((a'_v, b'_v))_{v \in V \setminus \{w\}} \in (\mathbb{F}_p)^{2n-2}$ is valid for $G - w$ at t, where*

$$a'_v = \begin{cases} a_v & \text{if } v \neq u \\ a_u + a_w & \text{if } v = u \end{cases} \qquad b'_v = \begin{cases} b_v & \text{if } v \neq u \\ b_u - a_w \delta_{uw} & \text{if } v = u \end{cases}$$

   where $\delta_{uw} = 1$ if $a_u + b_u = a_w + b_w$ and 0 otherwise.

2. *Let w be a node of degree 1 in G, and let u be its unique neighbor in G. Then, the vector $m' = ((a'_v, b'_v))_{v \in V \setminus \{w\}} \in (\mathbb{Z}_p)^{2n-2}$ is valid for $G - w$ at t, where*

$$a'_v = a_v \text{ for all } v \in V \setminus \{w\}, \qquad b'_v = \begin{cases} b_v & \text{if } v \neq u \\ b_u - a_w & \text{if } v = u \end{cases}$$

3. *Let $M = \{u_1, \ldots, u_k\}$ be a module in G of size k. The vector $m' = ((a'_v, b'_v))_{v \in V \setminus \{u_2, \ldots, u_k\}}$ of $(\mathbb{Z}_p)^{2n-2k}$ is valid for $G - \{u_2, \ldots, u_k\}$ at t, where*

$$a'_v = \begin{cases} a_v & \text{if } v \neq u \\ \sum_{i=1}^{k} a_{u_k} & \text{if } v = u_1 \end{cases} \qquad b'_v = \begin{cases} b_v & \text{if } v \neq u \\ b_{u_1} - \sum_{u \in M \cap N_G(u_1)} a_u & \text{if } v = u_1 \end{cases}$$

Proof. Let $\Phi = (\phi_v)_{v\in V}$ be a linearly independent family of polynomials associated to m.

To prove case (1), consider the family of polynomials $\Phi' = (\phi'_v)_{v\in V\setminus\{w\}}$, given by $\phi'_v = \phi_v$ for each $v \neq u$ and $\phi'_u = \phi_u + \phi_w$, which is trivially linearly independent because Φ is. We will show that $m' = ((a'_v, b'_v))_{v\in V\setminus\{w\}} \in (\mathbb{F}_p)^{2n-2}$ is valid for $G - w$ at t for the family of polynomials Φ', i.e., $a'_v = \phi'_v(t)$ and $b'_v = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t)$. First, if $v \neq u$ then $a'_v = \phi'_v(t) = \phi_v(t) = a_v$. On the other hand, $a'_u = \phi'_u(t) = \phi_u(t) + \phi_w(t) = a_u + a_w$. Now fix $v \neq u$, and note that either both u and w are neighbors of v, or none of them, because u and w are twins. If u and w are neighbors of v, then $b'_v = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) = \phi'_u(t) + \sum_{x\in N_G(v)\setminus\{u,w\}} \phi'_x(t) = \phi_u(t) + \phi_w(t) + \sum_{x\in N_G(v)\setminus\{u,w\}} \phi_x(t) = b_v$. On the other hand, if u and w are not neighbors of v, then $b'_v = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) = \sum_{x\in N_G(v)\setminus\{u,w\}} \phi'_x(t) = \sum_{x\in N_G(v)\setminus\{u,w\}} \phi_x(t) = b_v$. Finally, note that δ_{uw} equals 1 if u and w are adjacent, and 0 otherwise. Therefore $b'_u = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) = \sum_{x\in N_G(v)\setminus\{w\}} \phi_x(t) = \sum_{x\in N_G(v)} \phi_x(t) - \delta_{uw}\phi_w(t) = b_u - \delta_{uw}a_w$.

In case (2), we simply note that m' is a valid vector of $G - w$ in t for the family of polynomials $\Phi' = (\phi'_v)_{v\in V\setminus\{w\}}$ such that $\phi'_v = \phi_v$ for all $v \in V \setminus \{w\}$. Indeed, for all $v \in V \setminus \{w\}$, $a'_v = \phi'_v(t) = \phi_v(t) = a_v$. On the other hand, for all $v \neq u, w$, $b'_v = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) = \sum_{x\in N_G(v)\setminus\{w\}} \phi_x(t) = b_v$. Finally, $b'_u = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) = \sum_{x\in N_G(v)} \phi_x(t) - \phi_w(t) = b_u - a_w$.

With respect to case (3), we show that m' is valid for $G' = G - \{u_2, \ldots, u_n\}$ at t for the family of polynomials $\Phi' = (\phi'_v)_{v\in V\setminus\{u_2,\ldots,u_n\}}$ given by $\phi'_v = \phi_v$ for each $v \neq u_1$ and $\phi'_{u_1} = \sum_{i=1}^k \phi_{u_i}$, which is trivially linearly independent. For $w \neq u_1$, we have that $a'_w = a_w = \phi_w(t) = \phi'_w(t)$. Since M is a module in G any node $w \in V\setminus M$ is either adjacent to $\{u_1, \ldots, u_k\}$ or to none of them. In both cases $b'_w = b_w = \sum_{v\in N(w)} \phi'_v(t)$. By definition, $a'_{u_1} = \phi'_{u_1}(t) = \sum_{i=1}^k \phi_{u_i} = \sum_{i=1}^k a_{u_i}$. Finally, $b'_{u_1} = \sum_{u\in N_{G'}(u_1)} \phi'_u = \sum_{u\in N_G(u_1)\setminus M} \phi_u = b_{u_1} - \sum_{u\in M\cap N_G(u_1)} a_u$. □

Both distance hereditary and modular width recognition and reconstruction protocols will consist two parts: first, using the public random bits the nodes will conjointly produce a valid vector for the input graph, which is written on the whiteboard (or, equivalently, communicated to all nodes). Then, each node will use the information in the valid vector to produce a decomposition (this part does not require any extra communication). In the case of distance hereditary graph recognition, it will look for either a pendant node w or a pair of twins u, w, and then delete node w from the graph updating the valid vector according to Lemma 26. The process is reiterated until the whole graph is deleted (if the graph is distance hereditary), or the process is blocked (so the graph is not distance-hereditary). In the case of graphs of modular width at most k, it *guesses* a non-trivial module by testing every set W of at most k nodes, together with the graph induced by the module. Then, it suppresses all vertices of the module except one, as in Lemma 26, and repeats the process.

Theorem 23. *Let $G = (V, E)$ be the input graph and let $k \in \mathbb{N}$. There exist one-round public-coin protocols in the* BClique *model computing* Reconstruction on $\mathcal{G}$ *and* Recognition of $\mathcal{G}$ *with high probability and cost $c(n)$, where*

- *$\mathcal{G}$ is the class of distance-hereditary graphs and $c(n) = \mathcal{O}(\log n)$,*
- *$\mathcal{G}$ is the class of graphs of modular-width at most k, and $c(n) = \mathcal{O}(k^2 \log n)$.*

Proof. Let $G = (V, E)$ the input graph of size n. Let p be a prime number to be fixed later and $\Phi = (x^v)_{v\in V}$ as a family of polynomials in $\mathbb{Z}_p[X]$. The node protocol for both recognition problems is the following: nodes pick $t \in \mathbb{F}_p$ uniformly at random using the public randomness, and write on the whiteboard the message m_v such that $m = (m_v)_{v\in V}$ is valid for G at t for the family of polynomials $\Phi = (x^i)_{i\in V}$. Using the information on the whiteboard, the referee will decompose the input graph using the properties of distance hereditary graphs and graphs of modular width at most k.

First, if the decision problem is distance hereditary recognition, the referee defines $G^1 = G$ and iterates n times in order to find a twin-pendant vertex decomposition of G. At step i the referee looks for a pair of nodes u, w in G^i that satisfies one of the following: (1) $a_u = b_w$, or (2) $a_u \neq a_w$ and either $b_u = b_w$ or $a_u + b_u = a_w + b_w$. If the referee finds a pair satisfying (1) he deduces that that w has degree 1 and his unique neighbor is u. If he finds a pair that satisfying (2), he deduces that u and w are twins. In either case he computes a valid vector for $G^i - \{w\}$ in t according to the corresponding case of Lemma 26, and iterate in $G^{i+1} = G^i - \{w\}$. The process stops when the remaining graph is empty (and the referee accepts) or any pair satisfy neither (1) nor (2) (and the referee rejects).

Let ϕ^i be the linearly independent family of polynomials corresponding to the valid vector of G^i in the i-th iteration. For each $u, v \in V(G^i)$, $u \neq v$, consider the polynomials $\alpha^i_{u,v} = \phi^i_u - \phi^i_v$, $\beta^i_{u,v} = \sum_{w \in N_{G^i}(u)} \phi^i_w - \sum_{w \in N_{G^i}(v)} \phi^i_w$, $\gamma^i_{u,v} = \sum_{w \in N_{G^i}[u]} \phi^i_w - \sum_{w \in N_{G^i}[v]} \phi^i_w$ and $\sigma^i_{u,v} = \phi^i_u - \sum_{w \in N_{G^i}(v)} \phi^i_w$. The protocol errs when a pair of nodes u, w satisfy (1) or (2) without being twins nor one node pending of the other. The probability of this event is at most the probability that at any iteration t is a root of a nonzero polynomial $\alpha^i_{u,v}, \beta^i_{u,v}, \gamma^i_{u,v}, \sigma^i_{u,v}$ for some pair u, v in any iteration. The union of the families $\alpha^i = (\alpha^i_{u,v})_{u,v \in V}$, $\beta^i = (\beta^i_{u,v})_{u,v \in V}$, $\gamma^i = (\gamma^i_{u,v})_{u,v \in V}$ and $\sigma^i = (\sigma^i_{u,v})_{u,v \in V}$, have at most $4n^2$ polynomials, there are at most n iterations, and each polyinomial has degree at most n. This implies that the protocol fails with probability at most $4n^4/p$. If we pick $p \in [4n^{c+4}, 4n^{c+5}]$ for $c > 1$ we obtain that the protocol fails with probability at most $1/n^c$.

In the case of modular width k graph recognition, the referee looks in G for a set of at most k nodes that induce module M, and then compute the valid vector of the graph obtained from the deletion of the module excepting one node, according to Lemma 26. The referee will test every possible set of $k' \leq k$ nodes $M = \{u_1, \ldots, u_{k'}\}$, and every possible prime graph $H = (M, E')$ if M is a module with $G[M] = H$. Notice that a set of nodes $\{u_1, \ldots, u_{k'}\}$ is a module with induced graph H, then (3) $s_{u_1} = \cdots = s_{u_{k'}}$, where $s_u = \sum_{w \in N_G(v)} a_w - \sum_{w \in N_H(v)} a_w$. Once a module is found, a valid vector of the graph $G - \{u_2, \ldots, u_{k'}\}$ in t is computed according to Lemma 26, and then the referee iterates in $G - \{u_2, \ldots, u_{k'}\}$. The process stops when the remaining graph is empty (and the referee accepts), or any module of size at most k is found in the graph, and the referee rejects.

Let ϕ^i be the linearly independent family of polynomials corresponding to the valid vector of the graph G^i corresponding to the i-th iteration. For each $u, v \in V(G^i)$ and $H = (W, E')$ with $u, v \in W \subset V(G^i)$, consider the polynomials $\rho^i_{H,v} = \sum_{w \in N_{G^i}(v)} \phi^i_w - \sum_{w \in N_H(v)} \phi^i_w$, and $\tau^i_{H,u,v} = \rho^i_{H,u} - \rho^i_{H,v}$. The protocol errs if at some iteration there exists a set of at most k nodes M and a prime graph $H = (M, E')$ satisfying (3) but not being a module. The probability of that event is at most the probability that at any iteration t is a root of a nonzero polynomial $\tau^i_{u,v,H}$ for every possible choice of set M of at most k nodes, prime graph $H(M, E')$, and nodes $u, v \in M$. The union of these families have at most $k^2 \cdot 2^{k^2} \cdot n^{k+2}$ polynomials at any iteration. Since there are at most n iterations, and each polynomial has at most n roots, the protocol fails with probability at most $n^{f(k)}(n/p)$, with $f(k) = \mathcal{O}(k^2)$. Picking $p \in [n^{f(k)+c}, n^{f(k)+c+1}]$ we obtain that the protocol fails with probability at most $1/n^c$. □

10.4 Small classes of graphs

In this section, we show that in the Congested Clique model, we can efficiently reconstruct (therefore compute any function in) any small class of graphs with sub linear cost. Moreover, we show that this reconstruction and recognition can be done by a deterministic protocol with just two rounds, or with high probability by a one-round private-coins protocol.

For to graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$ we define the *vertex-distance* between G_1 and G_2, denoted $d(G_1, G_2)$, as the number of nodes $v \in V$ such that $N_{G_1}(v) \neq N_{G_2}(v)$. In an analogy with error correcting codes, we define *error correcting supergraphs* with parameter d as follows: For a graph $G = (V, E)$ of size n, and a Reed-Solomon code C with parameters n and d, call $C(G)$ the graph on $m = n + d\lceil \log(2(n+d)) \rceil$ vertices such that, $V(C(G)) = V \cup V'$ where $|V'| = m - n$,$V' \times V' \cap E(C(G)) = \emptyset$ and if x_i is the i-th row of the adjacency matrix of G, then the i-th row of the adjacency matrix of $C(G)$ is $C(x_i)$. Note that $C(G)[V] = G$. We call V the *original vertices* and the set V' the *correcting vertices*.

Error correcting supergraphs satisfy that if G and G' are two different graphs on the same vertex set V, then the vertex-distance between $C(G)$ and $C(G')$ is at least d (i.e., there are at least d vertices in $C(G)$ with a different neighborhood than the corresponding vertex in $C(G')$). Indeed, since G and G' are different, there exist at least one vertex v in V with different neighborhoods in G and G'. From the definition of function C, the row of the adjacency matrix of $C(G)$ corresponding to v differ in at least d coordinates with respect to the one in $C(G')$. Since the adjacency matrix is symmetric, there are at least d vertices which have different neighborhoods in $C(G)$ and $C(G')$.

Let $p > n$ be a prime number, $a \in \mathbb{Z}^n$ and $t \in \mathbb{F}_p$. Let $P(a, t) = \sum_{i=1}^{n} a_i t^{i-1} \mod p$ the fingerprint of a and t. For $A = (a_{i,j})_{i,j \in [n]} \in \mathbb{Z}^{n^2}$ and $T \in \mathbb{F}_p^n$, call $P(A, T) = (F(a_i, t_i))_{i \in [n]}$, where $a_i = (a_{i,j})_{j \in [n]}$. Let G be a graph of size n and $T \in \mathbb{F}_p^n$, we call $P(G, T)$ the *fingerprint of* G, which corresponds to the vector $P(A, T)$, where A is the adjacency matrix of G. Let $\mathcal{G}$ be a class of graphs, and let C be a Reed-Solomon

code with parameters n and d, and $m = n + d\lceil \log(2(n+d)) \rceil$. The set,

$$\mathcal{R}_{p,d}(\mathcal{G}_n) = \{R \in (\mathbb{F}_p)^m : \exists G \neq G' \in \mathcal{G}_n \text{ such that } P(C(G), R) = P(C(G'), R)\}$$

is called the set of *roots* of $\mathcal{G}_n$. In the following lemma, we give a bound on the size of $\mathcal{R}_p(\mathcal{G}_n)$ with respect to the size of $\mathcal{G}_n$, n, p and d.

Lemma 27. *Let $\mathcal{G}$ be a class of graphs, $n, d > 0$, and $p > n$ be a prime number, then*

$$|\mathcal{R}_{d,p}(\mathcal{G}_n)| \leq n^d \cdot p^{m-d} \cdot |\mathcal{G}_n|^2$$

Proof. Let G and G' be two different graphs in $\mathcal{G}_n$. Since the distance between $C(G)$ and $C(G')$ is at least d, there exist a set $I \subset [m]$ such that, for all $i \in I$ the i-th row of the adjacency matrices of $C(G)$ and $C(G')$, called $C(x_i)$ and $C(x'_i)$, are different. Since the number of elements $r \in \mathbb{F}_p$ such that $P(C(x_i), r) = P(C(x'_i), r)$ is at most n (since $P(x_i - x'_i, \cdot)$ is a polynomial of degree at most n), we obtain that the number of elements $R \in \mathbb{F}_p^m$ such that $P(G, R) = P(G', R)$ is at most $n^d \cdot p^{m-d}$. Therefore $|\mathcal{R}_{d,p}(\mathcal{G}_n)| \leq n^d \cdot p^{m-d} \cdot |\mathcal{G}_n|^2$. □

This lemma implies that there exists $T \in \mathbb{F}_p^m$ such that, for all $G, G' \in \mathcal{G}$, the fingerprint $F(C(G), T)$ of G is different than the fingerprint $P(C(G'), T)$ of G'. Notice that this T depends only in the class of graphs and n. In our protocol, each node will compute the Reed-Solomon code of its input vector to obtain the input that it would obtain as an original vector of the error correcting supergraph of the input graph G. The problem is that only the original vertices of $C(G)$ are real nodes (players), while the correcting vertices cannot communicate. This issue is solved making each node to send the last $m - n$ coordinates of its neighborhood as original vertex in $C(G)$. Then, the referee uses all messages sent by real vertices to reconstruct the neighborhoods of the correcting vertices.

Theorem 24. *Let $\mathcal{G}$ be a class of graphs. Then for each $0 < d < n$,*

1) *There exist a one-round deterministic protocol in the* BClique *model that computes* Reconstruction on $\mathcal{G}$ *with cost* $\mathcal{O}(\log n + \log |\mathcal{G}_n|/d + d \log(n+d))$.

2) *There exist a two-round deterministic protocol in the* BClique *model that computes* Recognition of $\mathcal{G}$ *with cost* $\mathcal{O}(\log n + \log |\mathcal{G}_n|/d + d \log(n+d))$.

3) *There exist a one-round private-coins protocol in the* BClique *model that computes* Recognition of $\mathcal{G}$ *with high probability and cost* $\mathcal{O}(\log n + \log |\mathcal{G}_n|/d + d \log(n+d))$.

Proof. 1) Let p be the first prime greater that $n \cdot |\mathcal{G}_n|^{2/d}$ (then $p \leq 2n \cdot |\mathcal{G}_n|^{2/d}$). Let $m = n + d\lceil \log(2(n+d)) \rceil$. Each node computes a vector $T = (t_i)_{i \in [m]} \in \mathbb{F}_p^m$ such that $T \notin \mathcal{R}_{p,d}(\mathcal{G}_n)$ (each node computes the same T). This vector exists since using Lemma 27 we know that $|(\mathbb{F}_p)^m| = p^m > p^{m-n} \cdot (n^d \cdot |\mathcal{G}_n|^2) \geq |\mathcal{R}_{p,d}(\mathcal{G}_n)|$. Then, in the first (and unique) round each node i computes and communicates (1) $P(C(x_i), t_i)$, where x_i is the row of the adjacency matrix of the input graph G (i.e $C(x_i)$ is the adjacency row of vertex i in $C(G)$); and (2) the $d \log(n+d)$ last coordinates of $C(x_i)$. Note that cost of the protocol is $d \log(n+d) + \log p = \mathcal{O}(d \log(n+d) + \log n + \log |\mathcal{G}_n|/d)$ bits.

After the communication round, the referee uses the $d \log(n+d)$ last coordinates of each $C(x_i)$, $i \in V(G)$ to compute the rows of the adjacency matrix of $C(G)$ corresponding to vertices $i \in [n + d \log(n+d)] \setminus [n]$. Recall that form definition of $C(G)$, the subgraph of $C(G)$ induced by vertices in $[m] \setminus [n]$ is edgeless. Then the referee computes T and $P(x_i, t_i)$ for $i \in [m] \setminus [n]$, obtaining the $P(C(G), T)$.

Finally the referee searches for a labeled graph G' in $\mathcal{G}_n$ that satisfies $P(C(G'), T) = P(C(G), T)$. The fact that $T \notin \mathcal{R}_{p,d}(\mathcal{G}_n)$, means that for every pair $G, G' \in \mathcal{G}_n$, $G \neq G'$ implies that $P(C(G), T) \neq P(C(G'), T)$. Since the input graph G belongs to $\mathcal{G}_n$ and $P(C(G), T) = P(C(G'), T)$, necessarily $G = G'$.

2) Let us now transform the previous protocol into a two-round protocol deterministic for Recognition of $\mathcal{G}$. The first round is exactly the same than in the previous protocol. In the second round, each node knows $P(C(G), T)$ and uses this information search for a labeled graph $G' \in \mathcal{G}_n$ that satisfy $P(C(G'), T) = P(C(G), T)$ (we call such a graph a *candidate*). If no such a graph exists, the nodes deduce that the input graph does not belong to $\mathcal{G}_n$. Otherwise, each node i checks if its neighborhood in G corresponds with the one of vertex i in G', and communicates the answer. The protocol accepts

if every node communicates *yes* in the second round. From the definition of T, each node necessarily compute the same candidate G', therefore the protocol accepting means that the neighborhood of each node coincides with the one of the candidate.

3) We show now how to transform the last protocol into a one-round private-coins protocol that solves RECOGNITION OF $\mathcal{G}$ with high probability. The idea is to check the identity of the candidate G' communicating also an equality-tester (fingerprint) in the first round. Let $p' \in [n^{c+2}, 2n^{c+2}]$ be a prime number where $c > 2$. In our randomized protocol, each node repeats the deterministic protocol of point (1), but also chooses $s_i \in \mathbb{F}_p$ uniformly random, and communicates the value $P(x_i, s_i)$ together with s_i. In this case each node communicates $d + \log p + 2\log p' = \mathcal{O}(d + \log n + \log|\mathcal{G}_n|/d)$ bits.

After the communication round the referee searches for a candidate $G' \in \mathcal{G}_n$ satisfying $P(C(G), T) = P(C(G'), T)$. As before, if no such graph exists the referee deduces that G does not belong to $\mathcal{G}$. Otherwise, the referee checks if $P(G, S) = P(G', S)$, where $S = (s_1, \ldots, s_n)$. The probability that $P(G, S) = P(G', S)$ when $G \neq G'$ is at most $n/p \leq 1/n^{c+1}$ (indeed is $(n/p)^k \leq 1/n^{k(c+1)}$, where $k = d(G, G')$). Therefore, the referee correctly identify if G' equals the input graph with probability at least $1 - 1/n^c$. □

Note that the optimal $0 < d < n$ for a class of graphs $\mathcal{G}$ is obtained noticing that if $d < n$ then $\log(2(n + d)) < 2\log n$, and picking $d = \sqrt{\frac{\log|\mathcal{G}_n|}{\log n}}$. In that case the cost of the protocols becomes $\mathcal{O}(\log n + \sqrt{\log|\mathcal{G}_n| \log n})$.

Corollary 5. *Let $\mathcal{G}$ be a class of graphs. The following holds*

1) *There exist a one-round deterministic protocol in the* BCLIQUE *model that computes* RECONSTRUCTION ON $\mathcal{G}$ *with cost* $\mathcal{O}(\log n + \sqrt{\log|\mathcal{G}_n| \log n})$.

2) *There exist a two-round deterministic protocol in the* BCLIQUE *model that computes* RECOGNITION OF $\mathcal{G}$ *with cost* $\mathcal{O}(\log n + \sqrt{\log|\mathcal{G}_n| \log n})$.

3) *There exist a one-round private-coins protocol in the* BCLIQUE *model that computes* RECOGNITION OF $\mathcal{G}$ *with high probability and cost* $\mathcal{O}(\log n + \sqrt{\log|\mathcal{G}_n| \log n})$.

We finish this section showing that in the UCLIQUE model, we can the same ideas to build protocols that recognize and reconstruct any class of graphs $\mathcal{G}$ in a constant number of rounds with cost $\mathcal{O}(\log n + \log(|\mathcal{G}_n|)/n)$. The key idea is to use the extra power provided by the UCLIQUE model to communicate the fingerprints of the $\Theta(n)$ correcting vertices. To do this without increasing much the size of the messages we have to slightly modify some definitions.

Recall when we defined Solomon-Reed codes in Section 8.2.5, we initially defined the function C with range in $\mathbb{F}_q^{n+d}$, where $q \in (n+d, 2(n+2))$ is a prime number and n, d are the parameters of the correcting code. Note that the first n coordinates of $C(x)$ are still equal to the input x, but seen as elements of $\mathbb{F}_q$. For a graph G, call $MC(G)$ the squared matrix of dimension $n + d$ on elements of $\mathbb{F}_q$ defined as follows.

- For each $i \in [n]$, the i-th line of $MC(G)$ is $C(x_i) \in \mathbb{F}_q^{n+d}$, where x_i is the i-th row of the adjacency matrix of G.
- For each $i \in [d]$, the $n+i$-th line of $MC(G)$ is the vector $(C(x_1)_{n+i}, \ldots, C(x_n)_{n+i}, \vec{0}) \in \mathbb{F}_q^{n+d}$, where $\vec{0}$ is the zero-vector of $\mathbb{F}^d$, and $C(x)_j \in \mathbb{F}_q$ is the j-th coordinate of $C(x)$.

Note that if $G \neq G'$, then the number of lines where $MC(G)$ and $MC(G')$ differ is at least d.

Let $p > q$ be another prime number. The *fingerprint* of an element of $x \in \mathbb{F}_q^n$ and $t \in \mathbb{F}_p$ is defined in the same way we defined fingerprints for Boolean vectors, i.e., the polynomial in $\mathbb{F}_p[X]$ such that $P(x, t) = \sum_{i \in [n]} x_i t^{i-1}$ (where the coordinates of x are seen as elements of $\mathbb{F}_p$). In a similar way, if A is a squared matrix of dimension n with coordinates in $\mathbb{F}_q$, the fingerprint of A and $T \in \mathbb{F}_p^n$ is defined as a vector $P(A, T) = (P(A_i, T))_{i \in [n]}$ where A_i is the $i - th$ line of A.

Obviously if $x, y \in \mathbb{F}_q^n$, are such that $x \neq y$, then there are at most n elements in $\mathbb{F}_p$ where the fingerprint of x equals to the fingerprint of y. Moreover, for a class of graphs $\mathcal{G}$ if we define for a class of graphs the set of roots

$$\mathcal{S}_{p,d}(\mathcal{G}_n) = \{S \in (\mathbb{F}_p)^{n+d} : \exists G \neq G' \in \mathcal{G}_n \text{ such that } P(MC(G), S) = P(MC(G'), S)\}$$

Following the same arguments than in the proof of Lemma 27, we obtain that for any class $\mathcal{G}$, $|\mathcal{R}_{d,p}(\mathcal{G}_n)| \leq n^d \cdot p^n \cdot |\mathcal{G}_n|^2$. We are now ready to show our result.

Theorem 25. *Let $\mathcal{G}$ be a class of graphs. The following holds*

1) *There exist a two-round deterministic protocol in the* UClique *model that computes* Reconstruction on $\mathcal{G}$ *with cost* $\mathcal{O}(\log n + \log |\mathcal{G}_n|/n)$.

2) *There exist a three-round deterministic protocol in the* UClique *model that computes* Recognition of $\mathcal{G}$ *with cost* $\mathcal{O}(\log n + \log |\mathcal{G}_n|/n)$.

3) *There exist a two-round public-coin protocol in the* UClique *model that computes* Recognition of $\mathcal{G}$ *with high probability and cost* $\mathcal{O}(\log n + \log |\mathcal{G}_n|/n)$.

Proof. Let p be the first prime greater than $n \cdot |\mathcal{G}_n|^{2/n}$ (then $p \leq 2n \cdot |\mathcal{G}_n|^{2/n}$), and let q be the smallest prime number such that $2n < q$ (i.e., $q < 4n$). Before the first communication round, each node i computes $C(x_i) \in \mathbb{F}_q^{2n}$, where C us the Solomon-Reed code with parameters n and $d = n$. Then, for each $j \in [n]$ each player i it communicates $C(x_i)_{j+n}$ to player j. This communication round requires bandwidth $2\lceil \log q \rceil = \mathcal{O}(\log n)$.

After the first communication round, node i possesses $C(x_i)$ and $(C(x_1)_{i+n}, \ldots, C(x_n)_{i+n})$, i.e., it knows lines i and $i+n$ of matrix $MC(G)$. Each node computes a vector $T = (t_i)_{i \in [2n]} \in \mathbb{F}_p^{2n}$ such that $T \notin \mathcal{S}_{p,n}(\mathcal{G}_n)$ (each node computes the same T). This vector exists since $p^{2n} > p^n \cdot (n^n \cdot |\mathcal{G}_n|^2) \geq |\mathcal{S}_{p,n}(\mathcal{G}_n)|$. Then, node i communicates (broadcasts) $P(MC(G)_i, T_i)$ and $P(MC(G)_{i+n}, T_{i+n})$. This communication round requires bandwidth $2\lceil \log p \rceil = \mathcal{O}(\log n + (\log |\mathcal{G}_n|)/n)$.

After the second communication round, each node knows $P(MC(G), T)$, so locally they compute the unique candidate $G' \in \mathcal{G}_n$ such that $P(MC(G'), T) = P(MC(G), T)$. If we are solving Reconstruction on $\mathcal{G}$, then necessarily $G = G'$. If we are solving Recognition of $\mathcal{G}$ then in one more (broadcast) round, each node announces if the candidate corresponds with what they have in their input. For the randomized protocol for Recognition of $\mathcal{G}$, we save one round sending a fingerprint as we did in the private-coin protocol of Theorem 24. In any case, the protocol can be performed in the bounds on the bandwidth required by the second round. □

10.5 Discussion

We presented several protocols for reconstructing and recognizing small families of graphs. Between all the protocols presented in this chapter, if there is the alternative (for example if we know that our input is d-degenerate and of modular width k), it would be preferable to use the protocol for reconstructing d-degenerate graphs. Our protocol runs in one round, it is deterministic, and asks a logarithmic bandwidth, so basically has less possible requirements in terms of the measures that we defined for the broadcast congested clique model.

However, even if our protocol for d-degenerate graphs has a smaller cost compared by the one given by Becker *et al.* (Proposition 31), i.e a reduction from $\mathcal{O}(d^2 \log n)$ to $\mathcal{O}(d \cdot \log n)$, the local computation time of our protocol might be quite large compared to the one of [13]. Indeed, our protocol uses deterministic linear sketches, which involve to compute and test all possible vectors $T \in \mathbb{F}_p^n$, for $p = \mathcal{O}(n^d)$. Even if we assume that the deterministic linear sketches are computed beforehand, our protocol may still need to compute and compare eventually all the possible fingerprints for d-sparse vectors. It is an open question whether this can be optimized, to obtain a protocol with local time that is FPT parameterized by d.

Note that this issue may be more delicate for the protocols given in Section10.4, since they require a local computation that may take an enormous quantity of time. To compute T the nodes may have to construct all the possible fingerprints for all graphs in the class (or the error correcting graphs of the class, to be precise), which is exponential. Since the value of T depends only in the class of graphs, we may assume that it was computed beforehand. However, even in this case the nodes must then compute a candidate graph from the communicated fingerprints, so they may have to test again every graph in the class. An interesting open problem is whether the candidate G' can be computed from the fingerprint of a graph, so these parts of the protocols can be improved to be performed in polynomial time. Possibly some recovery properties of Reed-Solomon graphs may give some clues to this question.

An interesting remark is that our protocol reconstructing d-degenerate graphs and the protocols of Section 10.4 can be implemented as protocols in the *dynamic graph streaming* model, within the same space bounds (the number of rounds equals the number of passes, and bandwidth is the space).

For classes of graphs of bounded degeneracy, bounded modular width, or the class of distance hereditary graphs, the protocols on Sections 10.2 and 10.3 might be preferable than the ones provided by Corollary 5. Indeed, there are at most $2^{\mathcal{O}(n \log n)}$ labeled distance hereditary graphs, and families of graphs with degeneracy or modular width at most k contain at most $2^{f(k)n \log n}$ graphs. Corollary 5 provides protocols with cost $\mathcal{O}(\sqrt{n} \log n)$, which is worse than the cost $\mathcal{O}(\log n)$ required by our protocols in Section 10.3 and 10.2. However, the protocols for bounded modular width and distance hereditary graphs of Section 10.3 require public-coins coins, while the ones of Corollary 5 are deterministic or require only private-coins.

Many classes of graphs with *few* minimal separators can be recognized and reconstructed by our protocols. For example, many graph classes with a few minimal separators, e.g., interval graphs, polygon-circle graphs, d-trapezoid graphs have $2^{\mathcal{O}(n \log n)}$ labeled graphs. Indeed, these classes of graphs can be represented by intersection models of geometric objects, which can be encoded in $\mathcal{O}(\log n)$ bits per node. Therefore, they can be reconstructed and recognized with our protocols with cost $\mathcal{O}(\sqrt{n} \log n)$. Other example is the class of chordal bipartite graphs, which contain $2^{\mathcal{O}(n(\log n)^2)}$ labeled graphs [35], so it can also be reconstructed and recognized with cost $\mathcal{O}(\sqrt{n \log^3 n})$. In general, if a class of graphs contain $2^{n^{1+\epsilon}}$ graphs, then it can be recognized and reconstructed with our protocol with cost $\mathcal{O}(n^{(1+\epsilon)/2}\sqrt{\log n})$.

Two important exceptions are chordal graphs and weakly chordal graphs. These two classes of graphs are not small, so they cannot be reconstructed in the broadcast congested clique model. However, in the next chapter, we will show that these classes can be recognized by two-round randomized protocols with polylogarithmic cost in the BCLIQUE$_2$ and BCLIQUE$_3$ models, respectively. Recall that these models, the initial local knowledge of the nodes is expanded to more than just their immediate neighbors. Further research in this sense is to look for protocols in the BCLIQUE model that recognize chordal and weakly chordal graphs, as well as any class of all graphs with polynomially many minimal separators.

Chapter 11

Detection of short cycles and chordality

In this chapter, we consider several problems: INDUCED-CYCLE$_{=k}$, INDUCED-CYCLE$_{\leq k}$, and the problem INDUCED-CYCLE$_{>k}$, which consists in deciding, respectively, whether the input graph has an induced cycle of length exactly k, at most k, strictly larger than k. Problems CYCLE$_{=k}$, CYCLE$_{\leq k}$, CYCLE$_{>k}$ are defined in a similar way, but here we ask whether the input graph has a cycle (induced or not) of length k, at most k, strictly larger than k. We show first that problems INDUCED-CYCLE$_{\leq k}$, CYCLE$_{=k}$ and CYCLE$_{>k}$ can be solved by using our deterministic one round protocols reconstructing graphs of bounded degeneracy. Later, we show that INDUCED-CYCLE$_{>2k+1}$ can be solved by a two-round randomized protocol in the BCLIQUE$_k$ model with high probability. Note that problem INDUCED-CYCLE$_{>3}$ is equivalent to the problem of recognize the class of chordal graphs. Finally, we give some lower bounds including multi-round, randomized protocols in BCLIQUE$_r$. Recall that BCLIQUE$_r$ is the extension of the broadcast congested clique model where each node x receives as input the set of all edges lying on a path of length at most r, starting in x.

11.1 Introduction

One of the most studied problems in the BCLIQUE model are related to the existence of cycles in the input graph G. The first natural question one can formulate, that is, deciding whether G contains a cycle has been, until now, the only question amenable to a simple protocol. In fact, Becker *et al.* [13] show that a simple set of logarithmic bandwidth is sufficient to recognize, deterministically and in one round, whether the input graph G is a forest.

Natural questions concerning cycles has given strong negative results. Drucker *et al.* [57] showed that, if $\ell \geq 4$, then any protocol that decides whether the ℓ-node cycle C_ℓ is a subgraph (or an induced subgraph) of the input graph G needs $\Omega(ex(n, C_\ell)/nb)$ rounds, where $ex(n, H)$ is the maximum number of edges of an n-node graph which does not contain a subgraph isomorphic to H. Remark that $ex(n, C_\ell)$ is $\Theta(n^2)$ for odd values ℓ, and $\Theta(n^{1+1/\ell})$ for even values [30].

Recall that an induced path (resp. cycle) of graph G is also called a *chordless* path (cycle). A graph is called k-*chordal* if it does not contain any induced cycle of length greater than k. The 3-chordal graphs are simply known as *chordal graphs*. Weakly chordal graphs are a subclass of 4-chordal graphs.

In this chapter, we consider several problems: INDUCED-CYCLE$_{=k}$, INDUCED-CYCLE$_{\leq k}$, and the problem INDUCED-CYCLE$_{>k}$, which consists in deciding, respectively, whether the input graph has an induced cycle of length exactly k, at most k, strictly larger than k. Problems CYCLE$_{=k}$, CYCLE$_{\leq k}$, CYCLE$_{>k}$ are defined in a similar way, but here we ask whether the input graph has a cycle (induced or not) of length k, at most k, strictly larger than k.

Note that the existence of an induced cycle of length at most k is equivalent to the existence of a cycle (not necessarily induced) of length at most k. Therefore problem CYCLE$_{\leq k}$ is identical to INDUCED-CYCLE$_{\leq k}$, so the upper and lower bounds for these problems coincide. On the other hand, as we are going to explain later, finding a protocol for detecting induced cycles of length at least $k+1$ requires much more involved arguments than finding protocols for detecting cycles (not necessarily induced) of length at least $k+1$.

We are going to study positive results and lower bounds in the BCLIQUE$_r$ model, which recall that is the extension of the broadcast congested clique model where each node u receives as input the set of

all edges lying on a path of length at most r, starting in u.Notice that the BCLIQUE$_1$ is the BCLIQUE model.

We start in Section 11.2 giving upper bounds based on degeneracy. We use a result by Druker *et al.* [57], showing that graphs without short cycles have sublinear degeneracy, and a result from Birmelé showing that graphs without long cycles have bounded treewdith. Using Theorem 21, we obtain one-round deterministic protocols for INDUCED-CYCLE$_{\leq k}$, CYCLE$_{=k}$ and CYCLE$_{>k}$ with sublinear costs.

Then, in Section 11.3 we give a useful, "local" characterization of graphs which do not have long induced cycles. Using this, together with linear sketches of [1, 86] (see Section 8.2.2), we show that, if each node is allowed to see at distance $\lfloor k/2 \rfloor + 1$, then a polylogarithmic number of bits is sufficient for detecting in two rounds an induced cycle of length strictly larger than k. More precisely, we prove that for every $k \geq 3$, there exists a two-round protocol in the BCLIQUE$_{\lfloor k/2 \rfloor+1}$ model with cost $\mathcal{O}(\log^4 n)$ that solves INDUCED-CYCLE$_{>k}$ with high probability. The approach is based on the randomized protocol of Ahn *et al.* [1] for computing a spanning forest in the BCLIQUE model. Note that this implies that in particular we can detect chordal graphs in two-rounds with poly-logarithmic cost in the BCLIQUE$_2$ model.

Finally, in Section 11.4 we give study some lower bounds in the mentioned problems. We show that our protocol for short cycles is essentially sharp, even for multi-round protocols and the BCLIQUE$_r$ model with $r > 1$. More precisely, we show that our one-round deterministic protocol for recognizing and reconstructing graphs without induced cycles of length at most k can not be beaten by randomized multiround protocols, even if we allow the nodes to see up to distance $\lfloor k/4 \rfloor$. We found a similar situation for problems INDUCED-CYCLE$_{=k}$, INDUCED-CYCLE$_{\leq k}$, INDUCED-CYCLE$_{>k}$ and CYCLE$_{=k}$, and extend them for the BCLIQUE$_{lfloork/4\rfloor}$ model restricted to one-round protocols, or multi-round protocols conditioned bounds in the local running time.

11.2 Upper bounds based on degeneracy

A very helpful result in the study of graphs without short cycles is the one that relates the nonexistence of even cycles in G with the degeneracy of G. More precisely, for any even value k, graphs with no cycles of length exactly k (as subgraphs) have a relatively small degeneracy.

Proposition 47 ([57])**.** *Let k be even. Graphs with no cycles of length k are of degeneracy $\mathcal{O}(n^{2/k})$.*

With respect to long cycles, by a result of Birmelé [19], graphs with no long cycles of length greater than k have treewidth at most k.

Proposition 48 ([19])**.** *A graph without cycles of length greater than k have treewidth at most k.*

Together with Proposition 44, we obtain that a graph without cycles of length greater than k is k-degenerate.

Theorem 26. *There is a one-round deterministic protocol in the* BCLIQUE *model for:*

1. INDUCED-CYCLE$_{\leq k}$*, that communicates with cost $\mathcal{O}(n^{2/k} \log n)$ bits, if k is even.*
2. INDUCED-CYCLE$_{\leq k}$*, that communicates with cost $\mathcal{O}(n^{2/(k-1)} \log n)$ bits, if k is odd.*
3. CYCLE$_{=k}$*, that communicates with cost $\mathcal{O}(n^{2/k} \log n)$ bits, if k is even.*
4. CYCLE$_{>k}$*, that communicates with cost $\mathcal{O}(k \log n)$ bits.*

Proof. The NO-instances of these problems are of small degeneracy by Propositions 47. We simply apply Theorem 21 on the input graph and the corresponding degeneracy bound ($\mathcal{O}(n^{2/k})$ for INDUCED-CYCLE$_{\leq k}$ and CYCLE$_{=k}$ when k is even, $\mathcal{O}(n^{2/(k-1)})$ for INDUCED-CYCLE$_{\leq k}$ when k is odd, k for CYCLE$_{>k}$). Therefore, in one round each node either (1) fully reconstructs the graph and decides the property, or (2) notices that the degeneracy of the input graph is larger than the bound requested for NO instances of the first three problems and concludes that the graph is a YES instance. □

Previous protocol is rather restrictive. It is deterministic, it works in one-round and the information each node has about the graph is minimal, consisting in the 1-neighborhood. The question we ask here is the following: is it possible, by lifting previous restrictions, to decrease the *total* number of bits broadcasted by each node? In Section 11.4 we give negative answer to this question. In other words, the one-round deterministic protocol based on the degeneracy seems to be the best we can do.

11.3 Chordality

Unlike the case of short cycles, there is a significant difference between detecting long induced cycles and detecting long cycles (induced or not).

Recall that graphs without induced cycles of length greater than k are called k-chordal [35]. 3-chordal graphs, i.e., graphs in which every cycle (not necessarily induced) of 4 or more vertices has a chord, are called chordal graphs. It is known that a graph G is chordal if and only if, for each vertex $u \in V$, and each connected component C in $G - N[u]$, the neighborhood $N(C)$ of this component induces a clique in G. This "local" characterization has been exploited by Chandrasekharan *et al.* [40] for devising a fast parallel algorithm recognizing chordal graphs. We begin this section by extending previous characterization to arbitrary chordalities $k > 3$ in order to take advantage of this in our distributed framework.

Let G be a graph, $u \in V(G)$ and $k > 0$. Let $D_1, \ldots, D_p$ be the p connected components of $G - N^{\lfloor k/2 \rfloor}[u]$ (obtained by removing the vertices at distance at most $\lfloor k/2 \rfloor$ from u). Let H_u^k denote the graph obtained from G by contracting each component D_i into a single node d_i.

Lemma 28. *Let G be a graph. G is k-chordal if and only if, for every $u \in V(G)$, H_u^k is k-chordal.*

Proof. Suppose first that G is not k-chordal. Let $u \in V(G)$ be a vertex contained in some chordless cycle of length greater than k. Among the chordless cycles of length greater than k containing u, let C_l be one that has a minimum size intersection with $N^{\lfloor k/2 \rfloor}[u]$. We call $u_0, \ldots, u_{l-1}$ the nodes of the cycle C_l, where $u_0 = u$ and u_i is adjacent to $u_{i-1 \mod l}$ and $u_{i+1 \mod l}$.

The cycle C_l may *leave* the set $N^{\lfloor k/2 \rfloor}[u]$ and then *get in* again potentially many times. We will show that each time that it does that, the cycle intersects a different connected component of $G - N^{\lfloor k/2 \rfloor}[u]$. Thus, when we contract each such component into a single node, we obtain a chordless cycle of length greater than k. For $t \geq 0$, call $P_1, \ldots, P_t$ the connected components of $G[C_l] - N^{\lfloor k/2 \rfloor}[u]$. Since C_l is chordless, there exists for each $i \in \{1, \ldots, t\}$ a pair $a_i, b_i \in \{0, \ldots, l-1\}$ such that $P_i = \{u_j \in C_l \ : \ a_i < j < b_j\}$, and $|C_l \cap P_i| = \{u_{a_i}, u_{b_i}\}$. Moreover, since $P_i \cap N^{\lfloor k/2 \rfloor}[u] = \emptyset$, u_{a_i} and $u_{b_i}$are at distance exactly $\lfloor k/2 \rfloor$ from u in G.

For $i \in \{1, \ldots, t\}$, let D_i be the component of $G - N^{\lfloor k/2 \rfloor}[u]$ that contains P_i. Notice that $N(D_i) \cap C_l = \{a_i, b_i\}$. Indeed, suppose that $|D_i \cap C_l| \geq 3$, and let $J \subset \{1, \ldots, l\}$ be such that $j \in J$ if $x_j \in N(D_i)$. Let $m = \min(J)$ and $M = \max(J)$, and call P' a shortest path between u_m and x_M contained in D_i. Clearly $u_j \notin N(C_i)$ for any $j \in \{0, \ldots, m-1, M+1, \ldots, l-1\}$. Therefore the cycle $C' = u_0, \ldots, u_{m-1}, u_m, P', u_M, u_{M+1}, \ldots, x_{l-1}, x_0$ is chordless. Since x_m and x_M are at distance $\lfloor k/2 \rfloor$ from x in G, C' is of length greater or equal to $k+1$. Then C' is a chodless cycle of length greater than k that contains u and whose intersection with $N^{\lfloor k/2 \rfloor}[u]$ is strictly smaller than the one of C_l, which contradicts the fact that, by our choice of C_l, the intersection of C_l with $N^{\lfloor k/2 \rfloor}[u]$ is a minimum one. We conclude that each component of $G - N^{\lfloor k/2 \rfloor}[u]$ contains at most one of $\{P_1, \ldots, P_t\}$. Therefore, in the graph H_u^k, each path P_i of C_l will be contracted into a different vertex d_i. The new cycle is still a chordless cycle of length greater than k, in the graph H_u^k. So H_u^k is not k-chordal.

For the converse, suppose that there exists $u \in V(G)$ such that H_u^k contains a chordless cycle of length l greater than k , call $C_l = u_0, \ldots, u_{l-1}, x_0$ such a cycle. Consider the set $I = \{i \in \{0, \ldots, l-1\} \mid u_i \in C_l \backslash N^{\lfloor k/2 \rfloor}[u]\}$, of the indices of nodes in C_l that correspond to contracted components of $G - N^{\lfloor k/2 \rfloor}[u]$. For each $i \in I$, call D_i the connected component of $G - N^{\lfloor k/2 \rfloor}[u]$ corresponding to u_i. Notice that since C_l is chordless $N(D_i) \cap C_l = \{u_{i-1}, u_{i+1}\}$ (where the subindices are taken *modulo* l). Let P_i be a chordless u_{i-1}, u_{i+1}-path in $G[D_i \cup \{u_{i-1}, u_{i+1}\}]$.

Call C the cycle of G corresponding to C_l when we replace for each $i \in I$ the subpath u_{i-1}, u_i, u_{i+1} with P_i. Clearly the length of C is greater or equal than that of C_l. Also C is chordless since for each $i \in I$, the node u_i corresponds to a different component D_i, $|N(P_i) \cap C| = \{u_{i-1}, u_{i+1}\}$, and P_i is a chordless u_{i-1}, u_{i+1}-path. We conclude that G is not k-chordal. □

Lemma 28 provides us with a strategy for deciding k-chordality, i.e., for deciding whether the input graph G is a NO instance of problem Induced-Cycle$_{>k}$. For doing this every node x must compute the graph H_x^k and then decides whether H_x^k is k-chordal. In order to compute H_x^k, each node x needs first to find the connected components of $G - N^{\lfloor k/2 \rfloor}[x]$. Each nodes implements this by computing the connected components of $G - F_x$ outside $N^{\lfloor k/2 \rfloor}[x]$, where F_x is the set of all edges lying on a path of length at most $\lfloor k/2 \rfloor + 1$ starting in x.

11.3.1 Computing the connected components of $G - F_x$

Ahn *et al.* provide a probabilistic, one-round protocol for computing, in the BCLIQUE$_1[\mathcal{O}(\log^3 n)]$ model, a spanning forest of the input graph G with ϵ-error [1]. In their protocol, each node constructs a message based on its neighborhood and on a sequence of public random coins, and broadcasts it to all other nodes. Using all these messages, every node is able to construct a spanning forest of the graph with probability $1 - \epsilon$.

Here, we want each node x to compute the connected components of $G - F_x$. To do that, we place ourselves in the BCLIQUE$_{\lfloor k/2 \rfloor + 1}[\mathcal{O}(\log^4 n)]$ model. We do so because (i) we must amplify the bandwidth by a $\log(n)$ factor to ensure that the spanning tree protocol of [1] succeed with high probability, and (ii) every node needs to know the set of edges F_x. Using the spanning forest protocol of [1], we prove that each node x can construct a spanning forest of $G - F_x$ with high probability. The key observation is that, in the protocol of [1], the message of each vertex is a linear function (w.r.t. to the edges of the graph). Therefore, from the messages of G, each vertex x computes the messages that the protocol *would have constructed* on $G - F_x$.

Let G be a graph. Suppose that a node knows a set of edges $F \subseteq E(G)$, and in a previous round it has received enough information to construct a set of linear sketches of the graph G. We will show that we can use algorithm of Proposition 43 and the linearity of the sketch function to compute the connected components of $G - F$.

Algorithm 3: Spanning forest after removing edges

Input: A set of $t = \lceil \log n \rceil$ random strings $r_1, \ldots, r_t$ picked uniformly at random; a set of t connectivity sketches $S_{r_1}(G), \ldots, S_{r_t}(G)$ of a graph G of size n, produced with $r_1, \ldots, r_t$. A set of edges $F \subseteq E(G)$.

Output: A spanning forest of $G - F$.

```
foreach x ∈ V(G) and r ∈ {r_1, ..., r_t} do S_r^x ← S_r(x) ;
foreach e ∈ F and x ∈ e do
    Compute the vector b^{x,e} as in Lemma 29;
    foreach r ∈ {r_1, ..., r_t} do
        Compute S_r(b^{x,e});
        S_r^x ← S_r^x + S_r(b^{x,e});
Run the spanning tree algorithm of Proposition 43 on input {{S_r^x}_{x∈V(G)}}_{r∈{r_1,...,r_t}}. Call T the
  spanning forest produced by it;
return T.
```

Lemma 29. *There exists a one-round protocol in the* BCLIQUE$_{\lfloor k/2 \rfloor + 1}$ *model which computes, for every node $x \in V$, the connected components of $G - N^{\lfloor k/2 \rfloor}[x]$, with high probability and cost $\mathcal{O}(\log^4 n)$.*

Proof. The protocol works as follows. First, each node x sends $t = \lceil \log n \rceil$ different $1/n^2$-linear sketches (see Section 8.2.3) of its connectivity vector a^x, using t random strings $r_1, \ldots, r_t$. Note that each node knows F_x. Observe that the components of $G - N^{\lfloor k/2 \rfloor}[x]$ are exactly the components of $G - F_x$ without considering the nodes in $N^{\lfloor k/2 \rfloor}[x]$. In the following, we show that after the communication round, each node x can compute a spanning forest of $G - F_x$ with probability at least $1 - 1/n^2$. Therefore, the whole protocol succeeds with probability at least $1 - 1/n$.

Let $S_r(G) = (S_r(a^{x_1}), \ldots, S_r(a^{x_n}))$ be one of the $1/n^2$-connectivity sketches of G, produced with the random string r, received in the communication round. Consider, for each $e \in F_x$ and $u \in e$, the vector $b^{u,e}$ of dimension $\binom{n}{2}$ where,

$$b^{u,e}_{e'} = \begin{cases} -a^u_e & \text{if } e' = e, \\ 0 & \text{otherwise} \end{cases}, \text{ for each } e' \in \binom{n}{2}.$$

Let us call c^u be the connectivity vector of node u in $G - F_x$. Note that, for each $e \in \binom{n}{2}$,

$$c^u_e = a^u_e + \sum_{\{e' \in F : u \in e'\}} b^{u,e'}_e = \begin{cases} a^u_e & \text{if } e \in E(G) \setminus F_x, \\ 0 & \text{otherwise.} \end{cases}$$

If we define $S_r^u = S_r(a^u) + \sum_{\{e \in F : u \in e\}} S_r(b^{u,e})$, we obtain, by linearity of S_r, that $S_r^u = S_r(c^u)$ and then $\{S_r(c^u)\}_{u \in V}$ is a $1/n^2$-connectivity sketch of $G - F_x$ produced with r.

Then, after the communication round, any node x can obtain t different $1/n^2$-connectivity sketches of $G - F_x$ produced with random strings $r_1, \ldots, r_t$ picked uniformly at random. Therefore, by Proposition 43, it can produce a spanning forest of that graph with probability at least $1/n^2$. □

11.3.2 Deciding k-chordality

We are now able to express the distributed algorithm recognizing k-chordal graphs, see Algorithm 4.

Theorem 27. *For every $k \geq 3$ there exists a two-round randomized* BCLIQUE$_{\lfloor k/2 \rfloor + 1}$ *protocol that recognizes k-chordal graphs, and thus solves problem* INDUCED-CYCLE$_{>k}$*, with high probability and cost $\mathcal{O}(\log^4 n)$.*

Proof. In the first round, each node $x \in G$ computes the connected components of $G - N^{\lfloor k/2 \rfloor}[x]$ using the protocol of Lemma 29. After the first round, each node x uses its knowledge of G to locally reconstruct H_x^k by identifying the connected components $D_1, \ldots, D_p$ of $G - N^{\lfloor k/2 \rfloor}[x]$ and contracting each D_i into a unique vertex d_i. Note that x sees the edges between D_i and $N^{\lfloor k/2 \rfloor}[x]$. Finally, x checks whether H_x^k is k-chordal and communicates the answer in the second round. By Lemma 28, the input graph is chordal if and only if each vertex x communicated a YES answer. We emphasis that the second round is needed only because the nodes must all agree on the output.

We remark that the protocol may fail only when some node x tries to compute without success the components of $G - N^{\lfloor k/2 \rfloor}[x]$; this event may occur, from Lemma 29, with probability at most $1/n$. □

Algorithm 4: k-chordality

```
Round 1
    Run the protocol of Lemma 29 to compute the components of G − N^⌊k/2⌋[x];
Round 2
    Each node x builds H_x^k contracting each component of G − N^⌊k/2⌋[x] into a single node;
    Each node x checks whether H_x^k is k-chordal and communicates the answer to the other nodes;
    The graph is k-chordal if all messages communicated during this round are YES messages;
```

11.4 Lower bounds

Recall that BCLIQUE$_r$ is the extension of the broadcast congested clique model where each node u receives as input the set of all edges lying on a path of length at most r, starting in u. Our first result tackles the case where $r \leq \lfloor k/4 \rfloor$.

Theorem 28. *Let $\epsilon \leq 1/3$ and $k/4 < r \leq k/3$. Then, any ϵ-error randomized protocol in the* BCLIQUE$_r$ *model that solves* INDUCED-CYCLE$_{\leq k}$*, (resp.* CYCLE$_{=k}$*) has cost $\Omega(\log(|\mathcal{G}_n|)/(n \log n))$, where $\mathcal{G}_n$ is the family of connected graphs of size n without cycles of length at most k (resp. without cycles of length k).*

Proof. The proof of this theorem is very similar to a proof found in [57] where this result is proved for the BCLIQUE$_1$ model. Let $\mathcal{P}$ be a ϵ-error, one-round protocol in the BCLIQUE$_r$ model that solves INDUCED-CYCLE$_{\leq k}$ (resp. INDUCED-CYCLE$_{\leq k}$, CYCLE$_{=k}$) communicating $b(n)$ bits in $R(n)$ rounds on input graphs is of size n. Let $\{G_n\}_n$ be a family of graphs of size n with the maximum number of edges not containing a cycle of length at most k (resp. without cycles of length k).

We show that we can use $\mathcal{P}$ to solve the communication complexity problem DISJ$_m$. Recall that in this problem two players (Alice and Bob) receive each a subset of $\{1, \ldots, m\}$ and have to decide if the intersection of their sets is nonempty. The communication complexity of DISJ$_m$ is $\Omega(m)$ (see Section 8.2.1 and [97]).

The reduction works as follows: Alice and Bob interpret their inputs as subsets of $E(G_n)$, called E_A and E_B, respectively. Let $\tilde{G}_n$ be the graph composed by two copies of $V(G_n)$, called V_A and V_B, and where each node in V_A is connected by a path of length $2\lfloor k/4 \rfloor$ with its corresponding copy in V_B. The

edges between nodes in V_A (resp. V_B) are given by E_A (resp. E_B). Notice that $\tilde{G}_n$ has an cycle of length at most k (resp. a cycle of length k) if and only if $E_A \cap E_B \neq \emptyset$.

Let u be a vertex in $V(G_n)$ and let u_A and u_B the copies of u in V_A and V_B, respectively. We call P_u the path of length $2r$ from u_A to u_B in $\tilde{G}$. For each $u \in V(G)$, Alice (resp. Bob) can simulate $\mathcal{P}$ on each node u_A (resp. $u_B \in V_B$) and in the first (resp. the last) $\lfloor k/4 \rfloor$ nodes in P_u, since the messages sent by those nodes only depend on E_A (resp. E_B). We say that Alice (resp. Bob) *owns* those nodes.

We obtain a communication protocol for DISJ_m, where at each round Alice and Bob exchange the messages produced simulating protocol $\mathcal{P}$ on all the nodes that they own. This implies that protocol $\mathcal{P}$ on graphs of size $2n\lfloor k/4 \rfloor$ is such that $2n\lfloor k/4 \rfloor \cdot R(2n\lfloor k/4 \rfloor) \cdot b(2n\lfloor k/4 \rfloor) = \Omega(m(n))$. Therefore, protocol $\mathcal{P}$ on n-vertex graphs is such that $R \cdot b = \Omega(n^{1+2/k})$ if k is even and $R \cdot b = \Omega(n^{1+2/(k-1)})$ if k is odd. □

The number of graphs without cycles of length at most 2ℓ is $2^{\Omega(n^{1+1/\ell})}$ [30], and the number of graphs without cycles of length exactly k is $2^{\Omega(n^{1+2/k})}$ [30] for k even and $2^{\Omega(n^2)}$ if k is odd.

Corollary 6. *Let $0 < r < k/4$. Any one-round ϵ-error randomized protocol in the* BClique$_r$ *model for:*

- Induced-Cycle$_{\leq k}$ *has cost $\Omega(n^{2/k})$ if k is even, and cost $\Omega(n^{2/(k-1)})$ if k is odd.*
- Cycle$_{=k}$ *has cost $\Omega(n^{2/k})$ if k is even, and cost $\Omega(n)$ if k is odd.*

In the case where the nodes have more knowledge of the graph, i.e, when $k/4 \leq r \leq k/3$, we can obtain a tight bound for one-round protocols. Our result use a reduction from a classical simultaneous communication problem called Index$_n$ (see Section 8.2.1). Recall that in such problem there are two players, Alice and Bob. Alice receives a vector $a \in \{0,1\}^n$ and Bob an index $b \in \{1, \ldots, n\}$. Alice needs to send a single message to Bob allowing him to compute a_b. It is well known that even with shared randomness, Alice must send a message of size $\Omega(n)$ [97].

Theorem 29. *Let $\epsilon \leq 1/3$ and $k/4 < r \leq k/3$. Then, any ϵ-error, one-round protocol in the* BClique$_r$ *model that solves* Induced-Cycle$_{>k}$ *(resp.* Induced-Cycle$_{=k}$, Induced-Cycle$_{\leq k}$, Cycle$_{=k}$*) has cost $\Omega(\log(|\mathcal{G}_n|)/(n \log n))$, where $\mathcal{G}_n$ is the family of connected graphs of size n without induced cycles greater than k (resp. without induced cycles of length k, without cycles of length at most k, without cycles of length k).*

Proof. Let $\mathcal{P}'$ an ϵ-error one-round randomized protocol in the BClique$_r$ model which solves problem Induced-Cycle$_{>k}$ (resp. Induced-Cycle$_{=k}$, Induced-Cycle$_{\leq k}$, Cycle$_{=k}$) using messages of size $c(n)$. Since $\epsilon \leq 1/3$, we can construct a new one-round protocol $\mathcal{P}$ for the same problem(s), using messages of size $\mathcal{O}(c(n) \log n)$, whose error probability is smaller than $1/n^3$. In protocol $\mathcal{P}$ each vertex constructs t independent messages as in $\mathcal{P}'$, which is equivalent to t independent runs of protocol $\mathcal{P}'$. It suffices to take t such that $\epsilon^t \leq 1/n^3$.

Let $m = \lfloor \log |\mathcal{G}_n| \rfloor$, with $\mathcal{G}_n$ is the family of connected graphs of n nodes without induced cycles greater than k (resp. without induced cycles of length k, without cycles of length at most k, without cycles of length k). Let a, b an input of the Index$_m$ problem. In the following, we will show that $\mathcal{P}$ can be turned into a protocol for Index$_m$ where Alice sends a message of size $4rn \cdot c(k(n+1))$.

Let $G_1, \ldots, G_{|\mathcal{G}_n|}$ be the graphs in $\mathcal{G}_n$. Alice interprets her input $a \in \{0,1\}^m$ as a number in $1, \ldots, 2^m$, and then picks $G_a \in \mathcal{G}_n$. Then, she attaches to each node $v \in V(G_a)$ a disjoint path $P_v = v, v_2, v_3, \ldots, v_r$ of length r, obtaining a graph $\tilde{G}_a$. For $u, v \in V(G_a)$ call $\tilde{G}_a(u,v)$ the graph obtained from $\tilde{G}_a$ by adding a new path $P_{u,v}$ of length $k - 2r$ from u_r to v_r. Notice that $P_u \cup P_{u,v} \cup P_v$ is a chordless u,v-path of length k.

Alice then uses $\mathcal{P}$ to produce three messages for each node w in $V(\tilde{G}_a)$, such that $w \in P_u$ for some $u \in G_a$ (possibly $u = w$). In the first message $M_1(w)$, Alice communicates the message that w would output simulating $\mathcal{P}$ on input graph $\tilde{G}_a$. The second and third messages $M_2(w)$ and $M_3(w)$ are the ones that w would produce, for some $v \in V(G_a)$, $v \neq u$, on input $\tilde{G}_a(u,v)$ and $\tilde{G}_a(v,u)$, respectively. Notice that if $w \notin P_u \cup P_v$ then the message of w on $\tilde{G}_a$ equals the one on $\tilde{G}_a(u,v)$. Moreover, the message that w sends in $\tilde{G}_a(u,v)$ (respectively $\tilde{G}_a(v,u)$) in protocol $\mathcal{P}$ is the same for all vertices $v \neq u$, $v \in V(G_a)$.

For each pair $u, v \in V(G_a)$, Bob uses the messages received form Alice to build the set of messages that one would receive running $\mathcal{P}$ on input $\tilde{G}_a(u,v)$. Let $M(\tilde{G}_a(u,v)) = (M(w) : w \in V(G_a(u,v)))$ be such set of messages, where

- If w is a vertex of $\tilde{G}_a$ then

$$M(w) = \begin{cases} M_1(w) & \text{if } w \notin P_u \cup P_v \\ M_2(w) & \text{if } w \in P_u \\ M_3(w) & \text{if } w \in P_v \end{cases}$$

- If w is a vertex of $\tilde{G}_a(u,v)$ but not of $\tilde{G}_a$, then the message can be constructed directly by Bob. Indeed in this case w is internal to some path $P_{u,v}$, so all the edges seen by w in the model are on $P_{u,v}$, on P_u or on P_v. In particular, they do not depend on graph G_a.

Then, simulating again $\mathcal{P}$, but this time in $M(\tilde{G}_a(u,v))$, Bob obtains the answer of the problem Induced-Cycle$_{>k}$ (resp. Induced-Cycle$_{=k}$, Induced-Cycle$_{\leq k}$, Cycle$_{=k}$) on input $G_a(u,v)$ which has $r(n-2)+k$ vertices.

For the problem Induced-Cycle$_{>k}$, we claim that $\tilde{G}_a(u,v)$ has an induced cycle of length at most k if and only if u,v are not adjacent in G_a. Indeed, since $\tilde{G}_a(u,v)$ is connected, if u,v are not adjacent in G_a, then $P_u \cup P_{u,v} \cup P_v$ plus any shortest u,v-path in G_a form a cycle of length greater than k. By the other hand since G_a is k-chordal, so is $\tilde{G}_a$, so any chordless cycle of length greater than k in $\tilde{G}_a(u,v)$ must intersect $P_{u,v}$. The only possibility is when u and v are not adjacent.

For the other problems, we claim that $\tilde{G}_a(u,v)$ is a YES instance if and only if u,v are adjacent in G_a. Indeed, if u,v are adjacent then $P_u \cup P_{u,v} \cup P_v$ is an (induced) cycle of length k. Conversely, by definition of $\mathcal{G}_n$ for each problem, if G_a contains an induced cycle of length exactly k (resp. a cycle of length at most k, a cycle of length exactly k) then it must contain the path $P_u \cup P_{u,v} \cup P_v$, which already has k nodes, so necessarily $\{u,v\}$ is an edge of G_a.

We deduce that using the messages of Alice, Bob can check if any pair $u,v \in V(G_a)$ are adjacent with error probability smaller than $1/n^3$. Then, by querying all pairs u,v, Bob obtains the whole graph G_a with probability at least $1-1/n$. In particular, Bob obtained a_b. We conclude that $3rn \cdot c(r(n-2)+k) \cdot \log(r(n-2)+k) = \Omega(m)$. Therefore, $c(n) = \Omega(\log(\mathcal{G}_n)/n\log n)$. □

Recall again that the number of graphs without cycles of length at most 2ℓ is $2^{\Omega(n^{1+1/\ell})}$ [30]. Recall also that the number of graphs without cycles of length exactly k is $2^{\Omega(n^{1+2/k})}$ [30] for k even and $2^{\Omega(n^2)}$ if k is odd. On the other hand, the number of chordal graphs (hence, without induced cycles of length at least 4) is trivially $2^{\Omega(n^2)}$ (consider for instance the well-known class of split graphs [35]). From these bounds, we obtain:

Corollary 7. *Let $0 < r < k/3$. Any one-round ϵ-error randomized protocol in the* BClique$_r$ *model for:*

- Induced-Cycle$_{\leq k}$ *has cost $\Omega(n^{2/k}/\log n)$ if k is even, and cost $\Omega(n^{2/(k-1)}/\log n)$ if k is odd.*
- Cycle$_{=k}$ *has cost $\Omega(n^{2/k}/\log n)$ if k is even, and cost $\Omega(n/\log n)$ if k is odd.*
- Induced-Cycle$_{=k}$, *has cost $\Omega(n/\log n)$.*
- Induced-Cycle$_{>k}$, *has cost $\Omega(n/\log n)$.*

11.4.1 A Conditional multi-round lower bound for odd cycles

We give here a reduction based on the disjointness problem DISJ$_m$ in the number-on-the-forehead model. Consider three players Alice, Bob and Charlie, each one has a subset X_A (resp. X_B, X_C) of $\{1,\dots m\}$, and each player sees the vector of the two others. Their common goal is to decide whether the three sets have a non-empty intersection.

It is known that this problem requires a communication of $\Omega(m)$ in the deterministic case, and $\Omega(\sqrt{m})$ for any ϵ-error probabilistic protocol with $\epsilon < 1/3$. Moreover, the linear lower bound extends to the probabilistic case under the Strong Exponential Time Hypothesis (stating that, for any $\delta < 1$, there is no algorithm solving n-variable instances of Sat in time $\mathcal{O}(2^{\delta n})$. It is shown that, under this hypothesis and the assumption that local computations must be performed in time $2^{o(n)}$, there is no probabilistic algorithm for 3-party number-on-the-forehead disjointness communicating only $o(m)$ bits (see Section sec:comcom and [57] for discussions on these problems).

Proposition 49 ([57])**.** *There is a family of graphs $\{G_n\}_{n>0}$ with the following properties:*

- *G_n is a tripartite graph $G_n = (X \cup Y \cup Z, E)$ where $|X| = |Y| = n$, $|Z| = n/3$.*
- *G_n contains $m = n^2/e^{\mathcal{O}(\sqrt{\log n})}$ triangles, and each edge belongs to exactly one triangle.*

Drucker *et al.* used these tools providing lower bounds for triangle detection in BCLIQUE$_1[b]$. We extend their bounds in the following sense.

Theorem 30. *Let k be an odd number and let $0 < r \leq k/3$. Any ϵ-error randomized protocol in the* BCLIQUE$_r$ *model for problems* CYCLE$_{=k}$ *and* INDUCED-CYCLE$_{=k}$ *satisfies has cost $R_\epsilon^{3-NOF}(DISJ_m)/\mathcal{O}(n)$, where $R_\epsilon^{3-NOF}(DISJ_m)$ is the communication complexity of the 3-party number-on-the-forehead version of* DISJ$_m$.

Proof. Let $k \geq 3d$ odd and $G = (X \cup Y \cup Z, E)$ be a tripartite graph with m triangles, and where $|X| = n_1$, $|Y| = n_2$, $|Z| = n_3$. Consider the graph $\tilde{G}$, called *the expansion of* G, defined as follows:

- Replace each node $x \in X$, and each $y \in Y$ by paths of length d called $P_x = x_1, \ldots, x_r$ and $P_y = y_1, \ldots, y_r$.
- Replace each node $z \in Z$ by a path of length $k - 2r$ called $P_z = z_1, \ldots, z_{k-2r}$.
- For each $\{x, y\} \in (X \times Y) \cap E(G)$, $\{y, z\} \in (Y \times Z) \cap E(G)$ and $\{x, z\} \in (X \times Z) \cap E(G)$, add to $\tilde{G}$ the edges $\{x_r, y_1\}$, $\{y_r, z_1\}$ and $\{x_1, z_{k-2r}\}$, respectively.

Call $X_j = \{x_j\}_{x \in X}$, $Y_j = \{y_j\}_{y \in Y}$, $Z_j = \{z_j\}_{z \in Z}$.

We obtain that $\tilde{G}$ is a graph of size $N = r(n_1 + n_2) + (k - 2r)n_3$, that contains m cycles of length k, all induced. Indeed, if $\{x, y, z\} \in X \times Y \times Z$ induce a triangle in G, then $x_1, \ldots, x_r, y_1, \ldots, y_r, z_1, \ldots, z_{k-2r}$ induce an induced cycle of length k in $\tilde{G}$. By the other hand, if $\mathcal{C}$ is a cycle in $\tilde{G}$ that intersects each X_j, Y_j and Z_l in only one node, with $2 \leq j \leq r - 1$ and $2 \leq l \leq k - 2r - 1$, then $\mathcal{C}$ must contain $x_1, \ldots, x_r, y_1, \ldots, y_r, z_1, \ldots, z_{k-2r}$, for some $x, y, z \in X \times Y \times Z$, so $\mathcal{C}$ corresponds to a triangle in G.

We show that a cycle $\mathcal{C}$ of odd length smaller or equal than k in $\tilde{G}$ can not intersect X^j, Y^j or Z^l in more than one node, for any $2 \leq j \leq r - 1$ and $2 \leq l \leq k - 2r - 1$. Indeed, suppose w.l.g. that there exists $x^1, x^2 \in X$ such that x_j^1 and x_j^2 are in $\mathcal{C}$ for some $j \in \{2, \ldots, r - 1\}$. Then P_{x^1} and P_{x^2} must be contained in $\mathcal{C}$. Moreover, since the length of $\mathcal{C}$ is smaller than k, besides P_{x^1} and P_{x^2}, the cycle can only contain edges from $X_d \times Y_1$ and $X_1 \times Z_{k-2r}$. Let P_1, P_2 the parts of $\mathcal{C}$ in $X_r \cup Y_1$ and $X_1 \cup Z_{k-2r}$ respectively. Notice that P_1 and P_2 must be paths of even length, and P_{x^1} and P_{x^2} have both the same length. Therefore, the length of $\mathcal{C}$ must be even, which contradicts the fact that k is odd.

The rest of the proof is very similar to the one of [57] for triangle detection in BCLIQUE$_1$. Fix n and let $G_n = (X \cup Y \cup Z, E)$ be a graph of the family described in Proposition 49. Let $m = n^2/e^{\mathcal{O}(\sqrt{\log n})}$ be the number of triangles in G_n and let $\mathcal{T} = \{t_1, \ldots, t_m\}$, be the set of triangles of G_n. Call $\tilde{G}_n$ the expansion of G_n, and $\mathcal{C} = \{C_1, \ldots, C_m\}$ the set of cycles of length k in $\tilde{G}_n$.

For each $t_i \in \mathcal{T}$ let x_i, y_i, z_i be such that $t_i = \{x^i, y^i, z^i\} \in X \times Y \times Z$. From the arguments above, we know that for each triangle $t_i \in \mathcal{T}$ there exists an induced cycle $C_i \in \mathcal{C}$, such that $V(\mathcal{C}) = P_{x^i} \cup P_{y^i} \cup P_{z^i}$. Then, we can map each edge $e \in E(G_n)$ into an edge e' in $(X_r \times Y_1) \cup (Y_r \times Z_1) \cup (X_1 \times Z_{k-2r})$, and we identify such edge e' an index $i(e') \in \{1, \ldots, m\}$ such that $e \in \mathcal{C}_i$.

Let $\mathcal{A}$ be a protocol in BCLIQUE$_r$ that solves CYCLE$_{=k,N}$ in R rounds with bandwidth b. We will use $\mathcal{A}$ to build a protocol for $\mathcal{P}$ for $DISJ_m$ in the 3-party number-in-the-forehead model. Let $A, B, C \subset \{1, \ldots, m\}$ the input of $DISJ_m$. The three parties construct a subgraph H of $\tilde{G}_n$ in which:

- All nodes $V(\tilde{G}_n)$ are present in H.
- All edges in paths P_x, P_y and P_z are present in H, for each $x \in X$, $y \in Y$ and $z \in Z$
- An edge $e \in X_r \times Y_1$ is present in H if $i(e) \in C$, an edge $e \in Y_r \times Z_1$ is present if $i(e) \in A$, and an edge $e \in X_1 \times Z_{k-2r}$ is present if $i(e) \in B$.

Notice that in the BCLIQUE$_r$ model, nodes in $\cup_{x \in X} P_x$ do not see any edges in $B_r \times C_1$, nodes in $\cup_{y \in Y} P_y$ do not see any edges in $X_1 \times Z_{k-2r}$, and nodes in $\cup_{z \in Z} P_z$ do not see any edges in $A_r \times B_1$. Since we are working in the number-in-the-forehead model, Alice knows B and C, Bob A and C and Charlie A and B. Hence Alice can simulate $\mathcal{A}$ in all nodes in $\cup_{x \in X} P_x$, Bob in all nodes in $\cup_{y \in Y} P_y$, and Charlie in all nodes $\cup_{z \in Z} P_z$, and send in each round the corresponding messages.

The subgraph H contains a cycle of odd length smaller or equal than k if and only if there is a (induced) cycle $C_i \in \mathcal{C}$ whose edges appear in H, and this occurs iff $A \cap B \cap C \neq \emptyset$. When $\mathcal{A}$ terminates, with probability $1-\epsilon$ the players can know if H contains a triangle, and then if A, B, C are disjoint. The total cost of $\mathcal{P}$ is $N \cdot R \cdot b$ bits, and since $\mathcal{P}$ solves $DISJ_m$ with error at most ϵ, $2nk \cdot R \cdot b \geq R_\epsilon^{3-NOF}(DISJ_m)$. □

We can easily replace the randomized protocols of the previous theorem with deterministic ones. Therefore, we have:

Corollary 8. *Let $0 < r \leq \lfloor k/3 \rfloor$, with k odd.*

- *Any R-round deterministic protocol in the* BCLIQUE$_r$ *model for* CYCLE$_{=k}$ *or* INDUCED-CYCLE$_{=k}$ *has cost $\Omega(n/e^{\mathcal{O}\sqrt{\log n}}) = \Omega(n^{1-o(1)})$.*
- *Any R-round $\epsilon \leq 1/3$-error randomized protocol in the* BCLIQUE$_r$ *model, where the node's local computation is in time $2^{o(n)}$, has cost $\Omega(n/e^{\mathcal{O}(\sqrt{\log n})}) = \Omega(n^{1-o(1)})$, unless the exponential time hypothesis fails.*

11.5 Discussion

The results of this chapter are summarized in Table 11.1, the main ones are presented in bold characters.

One important open problem is the question whether INDUCED-CYCLE$_{>k}$ can be solved by a one-round protocol in the BCLIQUE$_{\lfloor k/2 \rfloor+1}$ model with polylogarithmic (or even sublinear) cost, as well as the question of multi-round lower bounds for this problem in the BCLIQUE$_r$ model for $r < k/2$.

Another interesting further research is lack of lower bounds in the BCLIQUE$_r$ model for problems INDUCED-CYCLE$\leq k$, CYCLE$_{=k}$, INDUCED-CYCLE$= k$, when $2r < k < 3r$, either for one or multi-round protocols.

We would like to point out that even in the very powerful UCLIQUE model, the protocols for cycle detection are rather slow. In fact, the best protocols for CYCLE$_{=k}$ has cost $\mathcal{O}(n^\rho \log n)$, for every $k \geq 3$, where $\rho < 0.15715$ [38]. One exception is the detection of squares C_4, for which an extremely elegant $\mathcal{O}(1)$-round protocol has been devised [38].

Note that our deterministic protocol given in Theorem 25 beats the protocol of [38] when $k \geq 14 > 2/\rho$ and k is even. Indeed, the class of graphs not containing a cycle of length k as a subgraph is of size $2^{\mathcal{O}(n^{1+2/k})}$, if k is even. Therefore, our protocol reconstructs and recognizes graphs without cycles of length even length $k > 14$ with cost $\mathcal{O}(n^{2/k} + \log n)$.

	$k \leq 2r$	$k \geq 3r$	$k \geq 4r$
CYCLE$_{\leq k}$, & INDUCED-CYCLE$_{\leq k}$	**There is a one-round deterministic BCLIQUE$_1[b]$ protocol,** $b = \begin{cases} \mathcal{O}(n^{2/k}\log n), & \textbf{if } k \textbf{ is even,} \\ \mathcal{O}(n^{2/(k-1)}\log n), & \textbf{if } k \textbf{ is odd.} \end{cases}$		
	There is a one-round deterministic BCLIQUE$_r[\mathcal{O}(1)]$ protocol.	Any one-round, ϵ-error BCLIQUE$_r[b]$ protocol satisfies $b = \Omega(n^{2/k}/\log n)$, k even, $b = \Omega(n^{2/(k-1)}/\log n)$, k odd.	**Any R-round ϵ-error BCLIQUE$_r[b]$ protocol satisfies $R \cdot b = \Omega(n^{2/k})$, k even, $R \cdot b = \Omega(n^{2/(k-1)})$, k odd.**
CYCLE$_{>k}$	Single-round deterministic BCLIQUE$_1[\mathcal{O}(k \log n)]$ protocol.		

	$k \leq 2r-2$	$k \geq 3r$
INDUCED-CYCLE$_{>k}$	**There is a two-round, randomized BCLIQUE$_r[\mathcal{O}(\log^4 n)]$ protocol.**	**Any one-round ϵ-error BCLIQUE$_r[b]$ protocol requires $b = \Omega(n/\log n)$.**

Table 11.1 – Summary of the results.

Chapter 12

About graph connectivity

In this short chapter, we would briefly discuss the CONNECTIVITY problem, which consists in determining if the input graph is connected. We first give a very simple R-round deterministic protocol in the BCLIQUE model that solves CONNECTIVITY with bandwidth $\mathcal{O}(n^{1/R}\log n)$. Later, we show a one-round deterministic protocol in BCLIQUE$_r$ model that solves CONNECTIVITY with bandwidth $\mathcal{O}(n^{1/r}\log n)$.

The results of this chapter were published in the 2016 ACM Symposium on Principles of Distributed Computing (PODC 2016) and were obtained in collaboration with Ioan Todinca [112].

12.1 Introduction

As we said in Chapter 8 and in the introduction of this thesis, the connectivity problem, or more specifically problem MAXIMUM SPANNING TREE (MST) is in the very beginning and somehow triggers the interest in the Congested Clique model. More precisely, the UCLIQUE model was defined in [107] where the authors studied the maximum spanning tree in the $\mathcal{CONGEST}$model, when the communication graph has constant diameter. The authors showed a protocol in the UCLIQUE model that computes a MST in $\mathcal{O}(\log\log n)$ rounds with bandwidth $\mathcal{O}(\log n)$. More recently, Ghaffari *et al.* [76] showed that a MST can be computed in the UCLIQUE model by a randomized protocol with $\mathcal{O}(\log^* n - \log^* b)$ rounds and bandwidth $\mathcal{O}(b\cdot\log n)$. This last result implies in particular a $\mathcal{O}(\log^* n)$-round protocol with bandwidth $\mathcal{O}(\log n)$, or a constant round-protocol with cost $\mathcal{O}(\log n\cdot\log\ldots\log n)$, where $\log\ldots\log n$ represent d compositions of the logarithmic function ,where d is a constant that depends in the number of rounds.

These two protocols are based on a parallel algorithm that computes a MST published in 1926 and due to Borůvka [115]. Let G be a graph, and denote by $\hat{V}$ the set of *supernodes*, which initially are the n singletons $\{\{u\}|u\in V\}$. The protocol runs int $t=\lceil\log n\rceil$ phases. At phase $0\leq i<t$, each supernode picks an incident edge to each set $\hat{v}\in\hat{V}$. Then, we merge the obtained connected components of supernodes into a single supernode. The procedure finishes when all supernodes are isolated, in which case the supernodes represent the connected components of G. The procedure finishes before $t=\lceil\log n\rceil$ steps since the number of supernodes at least halves at each step. This algorithm can be easily simulated in the Broadcast Congested Clique model. Indeed, at each round a node sends the identifier of an adjacent neighbor on G contained in a different supernode (or a bit indicating that no such node exists). Then every node merge the supernodes and repeat. This way, we obtain a protocol in the BCLIQUE model that computes a MST in $\mathcal{O}(\log n)$ rounds with bandwidth $\mathcal{O}(\log n)$

The super-fast protocols computing a MST in the UCLIQUE are basically Borůvka's algorithms, where multiple phases are simulated in a reduced number of rounds, identifying sets of vertices that will form a supernode, and then using load-balancing techniques (between other sophisticated techniques) to simulate several phases in one communication round. These techniques do not seem to be applicable in the broadcast version of the model. In fact, no protocol exists (randomized or deterministic) computing a MST that beats the cost $\mathcal{O}(\log^2 n)$ obtained simulating Borůvka's algorithm. This is still the case if we try to solve CONNECTIVITY.

As we explained in Section 8.2.3, Ahn, Guha and McGregor provided in [1] the first one-round randomized protocol in the Broadcast Congested Clique model that solves CONNECTIVITY with high probability and with bandwidth $\mathcal{O}(\log^4 n)$.

In this chapter, we observe that Borůvka's algorithm can be used to produce, for each $0 < \epsilon \leq 1$ a $\mathcal{O}(1/\epsilon)$-round protocol that computes a spanning forest in the BCLIQUE model with cost $n^{\epsilon} \cdot \log n$. Later, we propose a one-round connectivity protocol, this time the BCLIQUE$_r$ model with cost $\mathcal{O}(n^{1/r} \log n)$. Recall that BCLIQUE$_r$ corresponds to the broadcast congested clique model, when each node receives as input the graph induced by the nodes at distance at most r (one may think of this model as the one where vertices are allowed to perform r unrestricted *local* communications, and then some congested *global* communications).

12.2 A constant round deterministic protocol

In this section, we show that Borůvka's algorithm can be implemented in a constant number of rounds, increasing the bandwidth of the protocol to a super-logarithmic though sub-linear number of bits. The idea is to not send just one, but instead $n^{1/\epsilon}$ neighbors in pairwise different supernodes at each round, for $\epsilon > 0$. At each round, the number of supernodes will be reduced by a factor of $n^{-1/\epsilon}$, so the protocol will finish after at most ϵ rounds.

Theorem 31. *Let $\epsilon > 0$. There is a ϵ-round deterministic protocol that computes the connected components of the input graph in the* BCLIQUE *model with bandwidth $\mathcal{O}(n^{1/\epsilon} \cdot \log n)$. The protocol returns a spanning forest of the input graph.*

Proof. Let $G = (V, E)$ the input graph. First, denote $\hat{V}$ the set of *supernodes*, which initially are the n singletons $\{\{u\} | u \in V\}$. At each round, every node sends $n^{1/\epsilon}$ arbitrary neighbors in pairwise different supernodes (if it sees less than $n^{1/\epsilon}$ supernodes, then it sends one neighbor for each of them). At the end of each round adjacent supernodes are merged. Note that the number of bits communicated in each round is $\mathcal{O}(n^{1/\epsilon} \log n)$.

A supernode of round $t > 1$ is called *active* if it contains at least $n^{1/\epsilon}$ supernodes of round $t - 1$, and otherwise is called *inactive*. Note that at any round $t > 0$, an inactive supernode corresponds to a connected component of G. The protocol finishes when every supernode is inactive.

Let n_t be the number of active supernodes at round $t \geq 0$. Since the number of active nodes at round $t + 1$ at most $n_t / n^{1/\epsilon}$, the protocol stops in at most ϵ rounds. □

We remark that this result implies a two-round deterministic protocol for CONNECTIVITY in the BCLIQUE model with cost $\mathcal{O}(\sqrt{n} \log n)$.

12.3 A one-round deterministic protocol in the distance r model

We use our protocol reconstructing graphs of bounded degeneracy to produce a one-round deterministic protocol that solves CONNECTIVITY, this time in the BCLIQUE$_r$ model. Our protocol is based on the following idea: if we pick and remove one edge of each cycle of length at most $2r$ in the graph, then the resulting graph has the same connected components than the original graph and is $\mathcal{O}(n^{1/r})$ degenerate. The key is that in the BCLIQUE$_r$ model, every node knows all the cycles of length at most $2r$ containing it without any communication.

Let $G = (V, E)$ be a graph, $r > 0$, and σ some total ordering of E. Call $\tilde{E}$ the set of edges e in E satisfying that there exists a cycle C_l of length $l \leq 2r$ in G, such that e is the maximum edge of C_l with respect to σ. Call now $\tilde{G} = (V, E - \tilde{E})$. Note that $\tilde{G}$ has no cycles of length at most $2r$.

Proposition 50. *[8] G is connected if and only if $\tilde{G}$ is connected, and any spanning forest of $\tilde{G}$ is a spanning forest of G.*

We are ready now to formally show our result.

Theorem 32. *There is a one-round deterministic protocol that computes the connected components of the input graph in the* BCLIQUE$_r$ *model with cost $\mathcal{O}(n^{1/r} \cdot \log n)$. The protocol returns a spanning forest of the input graph.*

Proof. Choose an ordering of the edges of the input graph G, for example if we denote the edges $e = (u, v)$ with $u < v$, then $(u_1, v_1) < (u_2, v_2)$ if either $u_1 < u_2$ or $u_1 = u_2$ and $v_1 < v_2$.

In the protocol, a node v looks for all cycles of length at most $2r$ in G that contain it. Notice that nodes do this without any communication since they see all neighbors at distance at most r. For each such cycle, v picks the maximum edge according to the edge ordering, obtaining the row of the adjacency matrix of $\tilde{G}$ corresponding to v. Recall from Proposition 47 in the last chapter, that a graph without cycles of length $2r$ is of degeneracy $d = \mathcal{O}(n^{1/r}$. Therefore, each node can simulate the protocol recognizing d-degeneracy graphs (Theorem 21) when the input graph is $\tilde{G}$. Each node then reconstructs $\tilde{G}$ and outputs a spanning forest of $\tilde{G}$. From Proposition 50, a spanning forest of $\tilde{G}$ is a spanning forest of G. □

12.4 Discussion

We have shown that, in the BCLIQUE model, $\mathcal{O}(\epsilon)$ rounds are enough to decide connectivity deterministically with bandwidth $\mathcal{O}(n^{1/\epsilon} \cdot \log n)$. On the other hand, if nodes see the graph induced by the nodes at distance at most r, then connectivity can be decided by a deterministic one-round protocol, with messages of size $\mathcal{O}(n^{1/r} \cdot \log n)$. Of course, unless $r > \log n / \log\log n$, this protocol does not beat the cost $\mathcal{O}(\log^n)$ obtained from Borůvka's algorithm.

Like the results of Ahn *et al.* [1], our connectivity protocol can be transformed into protocols to detect if the input graph is bipartite. Indeed, we can use our protocol to find the connected components of graph $D(G)$, defined just before Proposition 33. The protocol asks to each node v to simulate the role of nodes v_1 and v_2 in graph $D(G)$. From Proposition 33 we know that G is bipartite if $D(G)$ has twice the number of connected components of G.

Let us recall that the existence of a one-round deterministic protocol for connectivity, in the broadcast congested clique with sub-linear message size, is still open. We might as well ask whether the problem could be solved by randomized protocols using only *private coins*, or even if there is a one-round public-coin protocol with cost $\mathcal{O}(\log n)$.

One explanation of why lack of lower-bounds for CONNECTIVITY in the Broadcast Congested Clique model, is because all the lower-bounds known for this model are are obtained by reductions to problems in communication complexity. As noticed by Becker *et al.* [14], we can easily solve the problem CONNECTIVITY in the model where each edge of the graph is divided between k different players in one round communicating $\mathcal{O}(k \log n)$ bits. We wonder if the techniques leading to lower-bounds in the communication rounds in the $\mathcal{CONGEST}$ model could be used in this context, since they are not only based on communication complexity but also exploit the structure of the communication network.

Chapter 13

Conclusion and perspectives of Part II

All the second part of this thesis was devoted to the study of the Broadcast Congested Clique model.

We succeed to separate, with respect to the bandwidth, different types of one-round protocols, say deterministic, private-coin and public-coin protocols. This is a clear aspect where the local shared information (i.e., the fact that every edge and non-edge is known by two players) plays an important role. Compared to the Number-in-hand model, we obtained that Translated-Twins have similar gaps as Equality for deterministic, private-coins and public-coins protocols (i.e $\Omega(n)$ for deterministic protocols, $\Omega(\sqrt{n})$ and $\mathcal{O}(\sqrt{n}\cdot\log n)$ for private-coins randomized, and $\Theta(\log n)$ for public-coins protocols). However, the gap on the cost between private-coins and deterministic protocols for problem Twins$_x$ is exponential.

An interesting research perspective consists in identify for which problems the cost of a private-coin protocol is lower bounded by the square root of the cost of the best deterministic protocol. We conjecture that any graph property that is preserved under isomorphism cannot be recognized with high probability by a private-coins protocol with a cost that is asymptotically smaller than the square root of a deterministic protocol.

The Broadcast Congested Clique model might seem a weak model, especially when it is restricted to one-round protocols. However, the results of Chapter 10 show that any small class of graphs can be recognized and reconstructed by one or constant-round protocols with sub-linear cost. Maybe surprisingly, many classes of graphs with *few* minimal separators can be recognized and reconstructed by our protocols. This includes polygon-circle, d-trapezoid, circular arc, chordal and weakly chordal graphs. The protocols for the two last classes run in the broadcast congested clique where the nodes have some extra information about their neighbors at distance more than one. Further research in this sense is to look for protocols in the BClique model that recognize chordal and weakly chordal graphs, as well as any class of all graphs with polynomially many minimal separators.

Another interesting question is whether all graphs can be reconstructed in a non-trivial number of rounds on UClique model with $\mathcal{O}(\log n)$ bandwidth. Unlike the BClique model, the unicast version of the congested clique communicates at least $n^2 \log n$ bits per round, so potentially all the information in the graph could be transmitted in a round. If we were able to somehow confine an arbitrary input graph into a small class combining only local information, then we may reconstruct it using the protocol that reconstruct graphs in a known small class (Theorem 25).

More formally, let $\mathcal{G}_n$ be the set of all labeled graphs of size n. Let $\epsilon \in (0,1)$ and $\mathcal{Q}_{n,\epsilon}$ be the set of all classes of graphs on n vertices (i.e., the subset of $2^{\mathcal{G}_n}$) containing at most $2^{n^{1+\epsilon}}$ graphs. We pose the following open question: Determine if there exist a subset Q of $\mathcal{Q}_{n,\epsilon}$ and function $f : \mathcal{G}_n \to [|Q|]$ such that

1) G belongs to the $f(G)$-th class in Q, and

2) f can be computed in the UClique model with cost $o(n - n^\epsilon \log n)$

If the answer is affirmative, then there would be a protocol solving Reconstruction (i.e., reconstruct an arbitrary graph) with cost $o(n)$. Indeed, the protocol first compute $f(G)$ with cost $o(n - n^\epsilon \log n)$ and deduce that G belongs to the $f(G)$-th class in Q, that we call $\mathcal{C}$. Then, use Theorem 25 to solve Reconstruction on$\mathcal{C}$ on input G, with cost $\mathcal{O}(n^\epsilon \log n)$.

Other interesting open questions involve problems Twins and Connectivity: We leave as an open question whether the one-round private-coin protocol with cost $\mathcal{O}(\sqrt{n}\log n)$ given in Theorem 19 is

optimal (this again is a particular case of the question on the gap between deterministic and private-coin protocols). On the other hand, we do not know any nontrivial protocols in the multi-round deterministic setting. As we mentioned in Chapter 9, using the matrix multiplication protocol of Censor-Hillel *et al.* [38] we can solve TWINS in the UCLIQUE model with cost $\mathcal{O}(n^{0.158})$.

Problem CONNECTIVITY has somehow complementary difficulties: Unlike TWINS, we do not know any nontrivial one-round protocols in the deterministic or private-coins settings. On the other hand, there is the one-round public-coins protocol with polylogarithmic cost due to Ahn *et al.* [1], and constant round deterministic protocols, as we showed in Chapter 12.

Finally, we would like to point out that some of the techniques presented in this thesis, e.g., the deterministic sparse linear sketches, can be implemented in the *dynamic graph streaming* model. In this model, each item of the data stream is a pair of nodes on a graph n and the goal is to compute some property of the graph obtained at the end of the stream. The restriction is that the algorithm can only examine the data stream one item at a time, and must have memory of size $o(n^2)$. The linearity property of the sketches technique allows to handle the edge additions and deletions, so the protocol detecting graphs of bounded degeneracy can be implemented in the *dynamic graph streaming* model with space $\mathcal{O}(d \log n)$. In this context, further research should consider the study of the techniques developed during this thesis in other models of computation.

Appendix A

List of Problems

In this section we give the formal definitions of the problems that will be mentioned in the first part of this thesis. These definitions were taken from the book of Cygan et. al. in Parameterized Complexity [48].

Clique
Input: A graph G and an integer k.
Question: Does there exist a set of k vertices of G that is a clique in G?

Connected Dominating Set
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G[X]$ is connected and $N_G[X] = V(G)$?

Connected Feedback Vertex Set
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G[X]$ is connected and $G - X$ is a forest?

Connected Vertex Cover
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G[X]$ is connected and $G - X$ is edgeless?

Cycle Packing
Input: A graph G and an integer k.
Question: Does there exist in G a family of k pairwise vertex-disjoint cycles?

Dominating Set
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $N_G[X] = V(G)$?

Feedback Vertex Set
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G - X$ is a forest?

Hamiltonian Cycle
Input: A graph G.
Question: Does there exist a chordless cycle C in G such that $V(C) = V(G)$?

Hamiltonian Path
Input: A graph G.
Question: Does there exist a chordless path P in G such that $V(P) = V(G)$?

Hitting Set
Input: A universe U, a family $\mathcal{A}$ of sets over U, and an integer k.
Question: Does there exist a set $X \subseteq U$ of size at most k that has nonempty intersection with every element of $\mathcal{A}$?

Independent Dominating Set
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G[X]$ is edgeless and $N_G[X] = V(G)$?

Independent Feedback Vertex Set
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G[X]$ is edgeless and $G - X$ is a forest?

Independent Set
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G[X]$ is edgeless?

Induced Matching
Input: A graph G and an integer k.
Question: Does there exist a set X of exactly $2k$ vertices of G such that $G[X]$ is a matching consisting of k edges?

Longest Induced Path
Input: A graph G and an integer k.
Question: Does there exist a chordless path P in G consisting of k vertices?

Maximum Leaf Spanning Tree
Input: A graph G and an integer k.
Question: Does there exist a set a spanning tree of G with at least k leaves?

Maximum Matching
Input: A graph G and an integer k.
Question: Does there exist an edge set $S \subseteq E(G)$ of size at least k such that no two edges in S share an endpoint?

Minimum Fill-In
Input: A graph G and an integer k.
Question: Does there exist an edge set F of size at most k such that $H = (V(G), E(G) \cup F)$ is chordal?

Odd Cycle Transversal
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G - X$ is bipartite?

Red-Blue Dominating Set
Input: A graph bipartite graph G with bipartition classes R and B and an integer k.
Question: Does there exist a set $X \subseteq R$ of size at most k such that $N_G[X] = B$?

Scattered Set (d-independent set)
Input: A graph G and integers k and d.
Question: Does there exist a set X of at least k vertices of G such that $\text{dist}_G(x, y) > d$ for every distinct $x, y \in X$?

Traveling salesman Problem (TSP)
Input: A graph G with edge weights $w : E(G) \to \mathbb{R}^+$.
Task: Find a closed walk of minimum possible total weight that visits all vertices of G.

Treedepth
Input: A graph G and an integer k.
Question: Is the treedepth of G at most k?

Treelength
Input: A graph G and an integer k.
Question: Is the treelength of G at most k?

Treewidth
Input: A graph G and an integer k.
Question: Is the treewidth of G at most k?

Vertex Cover
Input: A graph G and an integer k.
Question: Does there exist a set X of at most k vertices of G such that $G - X$ is edgeless?

Bibliography

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proc. of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467, 2012.

[2] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: Sparsification, spanners, and subgraphs. In *Proc. of the 31st Symposium on Principles of Database Systems*, PODS '12, pages 5–14, 2012.

[3] Dana Angluin. Local and global properties in networks of processors. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 82–93. ACM, 1980.

[4] Heger Arfaoui, Pierre Fraigniaud, David Ilcinkas, and Fabien Mathieu. Distributedly Testing Cycle-Freeness. In *Proceedings of the 40th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 8747 of *LNCS*, pages 15 – 28, Nouan-le-Fuzelier, France, June 2014.

[5] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[6] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[7] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[8] Baruch Awerbuch, Oded Goldreich, Ronen Vainish, and David Peleg. A trade-off between information and communication in broadcast protocols. *J. ACM*, 37(2):238–256, April 1990.

[9] László Babai and Peter G. Kimmel. Randomized simultaneous messages: Solution of a problem of Yao in communication complexity. In *Proc. of the 12th Annual IEEE Conference on Computational Complexity*, pages 239–246, 1997.

[10] Hans-Jürgen Bandelt and Henry Martyn Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[11] Michel Bauderon and Bruno Courcelle. Graph expressions and graph rewritings. *Mathematical systems theory*, 20(1):83–127, 1987.

[12] Florent Becker, Antonio Fernández Anta, Ivan Rapaport, and Eric Rémila. Brief announcement: A hierarchy of congested clique models, from broadcast to unicast. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015*, pages 167–169, 2015.

[13] Florent Becker, Adrian Kosowski, Martín Matamala, Nicolas Nisse, Ivan Rapaport, Karol Suchan, and Ioan Todinca. Allowing each node to communicate only once in a distributed system: shared whiteboard models. *Distributed Computing*, 28(3):189–200, 2015.

[14] Florent Becker, Martín Matamala, Nicolas Nisse, Ivan Rapaport, Karol Suchan, and Ioan Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. In *Proc. of the 25th IEEE International Parallel and Distributed Processing Symposium*, IPDPS '11, pages 508–514, 2011.

[15] Florent Becker, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. The simultaneous number-in-hand communication model for networks: Private coins, public coins and determinism. In *Proceedings of the 21st International Colloquium, on Structural Information and Communication Complexity*, SIROCCO 2014, pages 83–95, 2014.

[16] Anne Berry, Jean Paul Bordat, and Olivier Cogis. Generating all the minimal separators of a graph. *Int. J. Found. Comput. Sci.*, 11(3):397–403, 2000.

[17] Anne Berry, Jean Paul Bordat, and Olivier Cogis. Generating all the minimal separators of a graph. *Int. J. Found. Comput. Sci.*, 11(3):397–403, 2000.

[18] Anne Berry, Jean Paul Bordat, and Pinar Heggernes. Recognizing weakly triangulated graphs by edge separability. *Nord. J. Comput.*, 7(3):164–177, 2000.

[19] Etienne Birmelé. Tree-width and circumference of graphs. *Journal of Graph Theory*, 43(1):24–25, 2003.

[20] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Computing*, 25(6):1305–1317, 1996.

[21] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.

[22] Hans L. Bodlaender. Fixed-parameter tractability of treewidth and pathwidth. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 196–227. Springer, 2012.

[23] Hans L. Bodlaender and Fedor V. Fomin. Tree decompositions with small cost. *Discrete Applied Mathematics*, 145(2):143–154, 2005.

[24] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

[25] Hans L. Bodlaender, Ton Kloks, and Dieter Kratsch. Treewidth and pathwidth of permutation graphs. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*, volume 700 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 1993.

[26] Hans L. Bodlaender, Ton Kloks, Dieter Kratsch, and Haiko Müller. Treewidth and minimum fill-in on d-trapezoid graphs. *J. Graph Algorithms Appl.*, 2(2), 1998.

[27] Hans L. Bodlaender, Ton Kloks, Dieter Kratsch, and Haiko Müller. Treewidth and minimum fill-in on d-trapezoid graphs. *J. Graph Algorithms Appl.*, 2(2), 1998.

[28] Hans L. Bodlaender and Udi Rotics. Computing the treewidth and the minimum fill-in with the modular decomposition. *Algorithmica*, 36(4):375–408, 2003.

[29] Hans L. Bodlaender and Dimitrios M. Thilikos. Treewidth for graphs with small chordality. *Discrete Applied Mathematics*, 79(1):45 – 61, 1997.

[30] John A Bondy and Miklós Simonovits. Cycles of even length in graphs. *Journal of Combinatorial Theory, Series B*, 16(2):97–105, 1974.

[31] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(5&6):555–581, 1992.

[32] Vincent Bouchitté and Ioan Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM J. Comput.*, 31(1):212–232, 2001.

[33] Vincent Bouchitté and Ioan Todinca. Listing all potential maximal cliques of a graph. *Theor. Comput. Sci.*, 276(1-2):17–32, 2002.

[34] Nicolas Bousquet, Daniel Gonçalves, George B. Mertzios, Christophe Paul, Ignasi Sau, and Stéphan Thomassé. Parameterized domination in circle graphs. *Theory Comput. Syst.*, 54(1):45–72, 2014.

[35] Andreas Brandstädt, Jeremy P Spinrad, et al. *Graph classes: a survey*, volume 3. Siam, 1999.

[36] Hajo Broersma, Ton Kloks, Dieter Kratsch, and Haiko Müller. Independent sets in asteroidal triple-free graphs. *SIAM Journal on Discrete Mathematics*, 12(2):276–287, 1999.

[37] H.J. Broersma, E. Dahlhaus, and T. Kloks. A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Applied Mathematics*, 99(1–3):367 – 400, 2000.

[38] Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015*, pages 143–152, 2015.

[39] A. Chakrabarti, Yaoyun Shi, A. Wirth, and A. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 270–278, 2001.

[40] N. Chandrasekharan and S. Sitharama Iyengar. NC algorithms for recognizing chordal graphs and k trees. *IEEE Trans. Computers*, 37(10):1178–1183, 1988.

[41] Maw-Shang Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM J. Comput.*, 27(6):1671–1694, 1998.

[42] Mathieu Chapelle, Mathieu Liedloff, Ioan Todinca, and Yngve Villanger. Treewidth and pathwidth parameterized by the vertex cover number. In Frank Dehne, Roberto Solis-Oba, and Jörg-Rüdiger Sack, editors, *WADS*, volume 8037 of *Lecture Notes in Computer Science*, pages 232–243. Springer, 2013.

[43] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computers Science*, 411(40-42):3736–3756, 2010.

[44] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[45] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic*. Cambridge University Press, 2012.

[46] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

[47] Alain Cournier and Michel Habib. *A new linear algorithm for Modular Decomposition*, pages 68–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[48] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Dániel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[49] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. On cutwidth parameterized by vertex cover. *Algorithmica*, 68(4):940–953, 2014.

[50] Peter Damaschke, Haiko Müller, and Dieter Kratsch. Domination in convex and chordal bipartite graphs. *Inf. Process. Lett.*, 36(5):231–236, 1990.

[51] Hubert de Fraysseix. Local complementation and interlacement graphs. *Discrete Mathematics*, 33(1):29–35, 1981.

[52] Jitender S. Deogun, Ton Kloks, Dieter Kratsch, and Haiko Müller. On the vertex ranking problem for trapezoid, circular-arc and other graphs. *Discrete Applied Mathematics*, 98(1-2):39–63, 1999.

[53] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[54] G. A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25(1):71–76, 1961.

[55] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Transactions on Algorithms*, 11(2):13:1–13:20, 2014.

[56] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[57] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *ACM Symposium on Principles of Distributed Computing, PODC 2014*, pages 367–376, 2014.

[58] Bruno Escoffier, Laurent Gourvès, and Jérôme Monnot. Complexity and approximation results for the connected vertex cover problem in graphs and hypergraphs. *J. Discrete Algorithms*, 8(1):36–49, 2010.

[59] Hiroshi Eto, Fengrui Guo, and Eiji Miyano. Distance- d independent set problems for bipartite and chordal graphs. *J. Comb. Optim.*, 27(1):88–99, 2014.

[60] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex cover. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *ISAAC*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008.

[61] Orr Fischer, Rotem Oshman, and Uri Zwick. Public vs. private randomness in simultaneous multiparty communication complexity. In Jukka Suomela, editor, *Structural Information and Communication Complexity - 23rd International Colloquium, SIROCCO 2016, Helsinki, Finland, July 19-21, 2016, Revised Selected Papers*, volume 9988 of *Lecture Notes in Computer Science*, pages 60–74, 2016.

[62] Fedor V. Fomin, Archontia C. Giannopoulou, and Michal Pilipczuk. Computing tree-depth faster than 2^n. *Algorithmica*, 73(1):202–216, 2015.

[63] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

[64] Fedor V. Fomin, Dieter Kratsch, and Ioan Todinca. *Exact (Exponential) Algorithms for Treewidth and Minimum Fill-In*, pages 568–580. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[65] Fedor V. Fomin, Dieter Kratsch, Ioan Todinca, and Yngve Villanger. Exact algorithms for treewidth and minimum fill-in. *SIAM J. Comput.*, 38(3):1058–1079, 2008.

[66] Fedor V. Fomin, Mathieu Liedloff, Pedro Montealegre-Barba, and Ioan Todinca. Algorithms parameterized by vertex cover and modular width, through potential maximal cliques. In R. Ravi and Inge Li Gørtz, editors, *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, volume 8503 of *Lecture Notes in Computer Science*, pages 182–193. Springer, 2014.

[67] Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM J. Comput.*, 44(1):54–87, 2015.

[68] Fedor V. Fomin and Yngve Villanger. Finding induced subgraphs via minimal triangulations. In *STACS 2010*, LIPIcs, pages 383–394. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[69] Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012.

[70] Pierre Fraigniaud, Mika Göös, Amos Korman, Merav Parter, and David Peleg. Randomized distributed decision. *Distributed Computing*, 27(6):419–434, 2014.

[71] Pierre Fraigniaud, Juho Hirvonen, and Jukka Suomela. Node labels in local decision. In Christian Scheideler, editor, *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, volume 9439 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2015.

[72] Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004.

[73] Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In Gregory Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2013.

[74] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(1):25–66, 1967.

[75] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47 – 56, 1974.

[76] Mohsen Ghaffari and Merav Parter. MST in log-star rounds of congested clique. In Giakkoupis [77], pages 19–28.

[77] George Giakkoupis, editor. *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*. ACM, 2016.

[78] Petr A. Golovach, Pinar Heggernes, Dieter Kratsch, and Arash Rafiey. Finding clubs in graph classes. *Discrete Applied Mathematics*, 174:57–65, 2014.

[79] Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems*, pages 241–247. ACM, 2015.

[80] Rob Gysel. Minimal triangulation algorithms for perfect phylogeny problems. In Adrian Horia Dediu, Carlos Martín-Vide, José Luis Sierra-Rodríguez, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, volume 8370 of *Lecture Notes in Computer Science*, pages 421–432. Springer, 2014.

[81] James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and mst. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, pages 91–100, New York, USA, 2015. ACM.

[82] James W. Hegeman and Sriram V. Pemmaraju. Sub-logarithmic distributed algorithms for metric facility location. *Distributed Computing*, 28(5):351–374, 2015.

[83] James W. Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-constant-time distributed algorithms on a congested clique. In *Proceedings of the 28th International Symposium on Distributed Computing, DISC 2014*, pages 514–530, 2014.

[84] Pinar Heggernes, Pim van 't Hof, Bart M. P. Jansen, Stefan Kratsch, and Yngve Villanger. Parameterized complexity of vertex deletion into perfect graph classes. *Theor. Comput. Sci.*, 511:172–180, 2013.

[85] Stephan Holzer and Nathan Pinsker. Approximation of distances and shortest paths in the broadcast congest clique. In Emmanuelle Anceaume, Christian Cachin, and Maria Gradinariu Potop-Butucaru, editors, *19th International Conference on Principles of Distributed Systems, OPODIS 2015, December 14-17, 2015, Rennes, France*, volume 46 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[86] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58. ACM, 2011.

[87] Jarkko Kari, Martin Matamala, Ivan Rapaport, and Ville Salo. Solving the induced subgraph problem in the randomized multiparty simultaneous messages model. In *Proceedings of the 21st International Colloquium, on Structural Information and Communication Complexity*, SIROCCO 2015, pages 370–384, 2015.

[88] T. Kloks and D. Kratsch. Treewidth of chordal bipartite graphs. *Journal of Algorithms*, 19(2):266 – 281, 1995.

[89] Ton Kloks. Treewidth of circle graphs. In *Proceedings of the 4th International Symposium on Algorithms and Computation*, ISAAC '93, pages 108–117, London, UK, UK, 1993. Springer-Verlag.

[90] Ton Kloks, Hans L. Bodlaender, Haiko Müller, and Dieter Kratsch. Computing treewidth and minimum fill-in: All you need are the minimal separators. In *Proceedings of the First Annual European Symposium on Algorithms*, ESA '93, pages 260–271, London, UK, UK, 1993. Springer-Verlag.

[91] Ton Kloks, Hans L. Bodlaender, Haiko Müller, and Dieter Kratsch. Erratum: Computing treewidth and minimum fill-in: All you need are the minimal separators. In Jan van Leeuwen, editor, *Algorithms - ESA '94, Second Annual European Symposium, Utrecht, The Netherlands, September 26-28, 1994, Proceedings*, volume 855 of *Lecture Notes in Computer Science*, page 508. Springer, 1994.

[92] Ton Kloks, Dieter Kratsch, and C. K. Wong. Minimum fill-in on circle and circular-arc graphs. *J. Algorithms*, 28(2):272–289, 1998.

[93] A. V. Kostochka. Lower bound of the hadwiger number of graphs by their average degree. *Combinatorica*, 4(4):307–316, 1984.

[94] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

[95] Ilan Kremer, Noam Nisan, and Dana Ron. Errata for: "on randomized one-round communication complexity". *Computational Complexity*, 10(4):314–315, 2001.

[96] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In Soma Chaudhuri and Shay Kutten, editors, *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 300–309. ACM, 2004.

[97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[98] Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *J. Comb. Theory, Ser. B*, 73(1):7–40, 1998.

[99] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[100] Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In Panagiota Fatourou and Gadi Taubenfeld, editors, *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 42–50. ACM, 2013.

[101] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, New York, NY, USA, 1986.

[102] Mathieu Liedloff, Pedro Montealegre-Barba, and Ioan Todinca. Beyond classes of graphs with "few" minimal separators: FPT results through potential maximal cliques. In *Graph-Theoretic Concepts in Computer Science, 41st International Workshop, Proceedings of WG 2015, Munich*, 2015. To appear.

[103] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[104] Daniel Lokshantov, Martin Vatshelle, and Yngve Villanger. Independent set in P_5-free graphs in polynomial time. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 570–581. SIAM, 2014.

[105] Daniel Lokshtanov. On the complexity of computing treelength. *Discrete Applied Mathematics*, 158(7):820–827, 2010.

[106] Daniel Lokshtanov, Marcin Pilipczuk, and Erik Jan van Leeuwen. Independence and efficient domination on P_6-free graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1784–1803. SIAM, 2016.

[107] Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in O(log log n) communication rounds. *SIAM Journal on Computing*, 35(1):120–131, 2005.

[108] Zvi Lotker, Boaz Patt-Shamir, and David Peleg. Distributed MST for constant diameter graphs. *Distributed Computing*, 18(6):453–460, 2006.

[109] Federico Mancini. Minimum fill-in and treewidth of split+ke and split+kv graphs. *Discrete Applied Mathematics*, 158(7):747–754, 2010.

[110] Dániel Marx. Parameterized coloring problems on chordal graphs. In *Parameterized and Exact Computation, First International Workshop, IWPEC 2004*, volume 3162 of *LNCS*, pages 83–95. Springer, 2004.

[111] Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.

[112] Pedro Montealegre and Ioan Todinca. Brief announcement: Deterministic graph connectivity in the broadcast congested clique. In Giakkoupis [77], pages 245–247.

[113] Pedro Montealegre and Ioan Todinca. *On Distance-d Independent Set and Other Problems in Graphs with "few" Minimal Separators*, pages 183–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[114] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[115] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar borůvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1):3 – 36, 2001.

[116] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[117] Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67 – 71, 1991.

[118] Boaz Patt-Shamir and Marat Teplitsky. The round complexity of distributed sorting: extended abstract. In Cyril Gavoille and Pierre Fraigniaud, editors, *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 249–256. ACM, 2011.

[119] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000.

[120] J.Scott Provan and Roger C Burk. Two-connected augmentation problems in planar graphs. *Journal of Algorithms*, 32(2):87 – 107, 1999.

[121] Mihai Pătraşcu and Ryan Williams. *On the possibility of faster SAT algorithms*, pages 1065–1075.

[122] Vijay Raghavan and Jeremy Spinrad. Robust algorithms for restricted domains. *Journal of Algorithms*, 48(1):160 – 172, 2003.

[123] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[124] Felix Reidl, Peter Rossmanith, Fernando Sanchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014.

[125] Neil Robertson and P.D Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49 – 64, 1984.

[126] Neil Robertson and P.D Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.

[127] Neil Robertson and P.D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325 – 357, 2004. Special Issue Dedicated to Professor W.T. Tutte.

[128] Donald J. Rose, Robert Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976.

[129] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012.

[130] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[131] Konstantin Skodinis. Efficient analysis of graphs with small minimal separators. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, *Graph-Theoretic Concepts in Computer Science, 25th International Workshop, WG '99, Ascona, Switzerland, June 17-19, 1999, Proceedings*, volume 1665 of *Lecture Notes in Computer Science*, pages 155–166. Springer, 1999.

[132] Karol Suchan. Minimal separators in intersection graphs. Master's thesis, Akademia Gorniczo-Hutnicza im. Stanislawa Staszica w Krakowie, 2003.

[133] Ravi Sundaram, Karan Sher Singh, and C. Pandu Rangan. Treewidth of circular-arc graphs. *SIAM J. Discret. Math.*, 7(4):647–655, November 1994.

[134] Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *ICALP (1) Automata, Languages and Programming, 35th International Colloquium*, Lecture Notes in Computer Science, pages 634–645, 2008.

[135] Martin Vatshelle. *New Width Parameters of Graphs*. PhD thesis, University of Bergen, Norway, 2012.

[136] Mingyu Xiao and Hiroshi Nagamochi. *Exact Algorithms for Maximum Independent Set*, pages 328–338. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[137] Mihalis Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):351–358, 1982.

Pedro MONTEALEGRE BARBA

Algorithmes en Graphes Séquentiels et Distribués

Cette thèse porte sur des aspects structuraux et algorithmiques des graphes. Elle est divisée en deux parties, qui comportent deux études différentes : une partie sur des algorithmes centralisés-séquentiels, et une autre sur des algorithmes distribués.

Dans la première partie, on étudie des aspects algorithmiques de deux structures de graphes appelés *séparateurs minimaux* et *cliques maximales potentielles*. Ces deux objets sont au cœur d'un méta-théorème dû à Fomin, Todinca and Villanger (SIAM J. Comput. 2015), qui affirme qu'une grande famille des problèmes d'optimisation peut être résolue en temps polynomial, si le graphe d'entrée contient un nombre polynomial de séparateurs minimaux. La contribution de cette partie consiste à prolonger le méta-théorème de Fomin *et al.* de deux manières : d'un côté, on l'adapte pour qu'il soit valide pour une plus grande famille des problèmes ; de l'autre, on étend ces résultats à des version paramétrées, pour certains paramètres des graphes.

La deuxième partie de la thèse corresponde à une étude du modèle appelé « Diffusion dans une Clique Congestionnée ». Dans ce modèle, les sommets d'un graphe communiquent entre eux dans des rondes synchrones, en diffusant un message de petite taille, visible par tout autre sommet. L'objectif ici est d'élaborer des protocoles qui reconnaissent des classes de graphes, en minimisant la taille des messages et le nombre de rondes. La contribution de cette partie est l'étude du rôle du hasard dans ce modèle, et la conception de protocoles pour la reconnaissance et la reconstruction des certaines classes des graphes.

Mots clés : algorithmes paramétrés, cliques maximales potentielles, séparateurs minimaux, algorithmes distribués, modèle de diffusion dans une clique congestionnée.

Sequential and Distributed Graph Algorithms

This thesis is about structural and algorithmic aspects of graphs. It is divided in two parts, which are about two different studies: one part is about centralized-sequential algorithms, and the other part is about distributed algorithms.

In the first part of the thesis we study algorithmic applications of two graph structures called *minimal separators* and *potential maximal cliques*. These two objects are in the core of a meta-theorem due to Fomin, Todinca and Villanger (SIAM J. Comput. 2015), which states that a large family of graph optimization problems can be solved in polynomial time, when the input is restricted to the family of graphs with polynomially many minimal separators. The contribution of this part of the thesis is to extend the meta-theorem of Fomin et al. in two ways. On one hand, we adapt it to be valid into a larger family of problems. On the other hand, we extend it into a parameterized version, for several graph parameters.

In the second part of this thesis we study the *broadcast congested clique* model. In this model, the nodes of a graph communicate in synchronous rounds, broadcasting a message of small size visible to every other node. The goal is to design protocols that recognize graph classes minimizing the number of rounds and the message sizes. The contribution of this part is to explore the role of randomness on this model, and provide protocols for the recognition and reconstruction of some graph classes.

Keywords : parameterized algorithms, potential maximal cliques, minimal separators, distributed algorithms, broadcast congested clique

Laboratoire d'Informatique Fondamentale d'Orléans
Bâtiment IIIA, Rue Léonard de Vinci
B.P. 6759 F-45067 ORLEANS Cedex 2